

Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут»

**Л. М. Добровська  
І. А. Добровська**

# **Теорія та практика нейронних мереж**

## **Навчальний посібник**

*Рекомендовано Міністерством освіти і науки України  
як навчальний посібник для студентів вищих навчальних закладів,  
які навчаються за освітньо-професійною програмою підготовки  
спеціалістів та магістрів зі спеціальності  
«Інформаційні управляючі системи та технології»*

Київ  
НТУУ «КПІ»  
2015

УДК 004.032.26 (075.8)  
ББК 32.973я73  
Д56

*Рекомендовано Міністерством освіти і науки України  
(Лист № 41/11-12068 від 29.07.2014 р.)*

**Рецензенти:** *А. В. Гусинін*, канд. техн. наук, доц.,  
ТОВ «Тіч Консалтинг Україна»  
  
*В. О. Клименко*, канд. фіз.-мат. наук, доц.,  
Національний авіаційний університет

**Відповідальний редактор** *Ю. В. Антонова-Рафі*, канд. техн. наук,  
Національний технічний університет України  
«Київський політехнічний інститут»

**Добровська Л. М.**  
Д56 Теорія та практика нейронних мереж : навч. посіб. /  
Л. М. Добровська, І. А. Добровська. – К. : НТУУ «КПІ» Вид-во  
«Політехніка», 2015. – 396 с. – Бібліогр. : с. 385–387. – 55 пр.  
ISBN 978-966-622-691-7

Подано теоретичні відомості про основні теорії нейронних мереж. Розглянуто основні проблеми й алгоритми обробки інформації на основі нейромереж різної структури, проаналізовано способи вирішення різних проблем і причин, які обмежують їх можливості. Наведені алгоритми проілюстровано типовими прикладами. Запропоновано задачі для самостійного розв'язання.

Для студентів – майбутніх фахівців з інформаційних технологій, аспірантів та викладачів вищих технічних навчальних закладів усіх рівнів акредитації. Також буде корисним для підвищення та перепідготовки спеціалістів з інформаційних технологій.

**УДК 004.032.26(075.8)**  
**ББК 32.973я73**

ISBN 978-966-622-691-7

© Л. М. Добровська,  
І. А. Добровська, 2015  
© НТУУ «КПІ» (ММІФ), 2015

## ЗМІСТ

<b>Умовні скорочення.....</b>	<b>7</b>
<b>Передмова .....</b>	<b>8</b>
<b>Вступ .....</b>	<b>10</b>
<b>Розділ 1. Поняття про нейронні мережі .....</b>	<b>12</b>
1.1. Біологічний нейрон .....	12
1.2. Модель штучного нейрона .....	15
1.3. Архітектура мережі .....	17
1.4. Перша модель штучної нейронної мережі та її особливості.....	23
1.5. Класифікація нейронних мереж.....	24
1.6. Подання знань засобами нейронних мереж.....	25
<i>Приклади розв'язання задач .....</i>	<i>31</i>
<i>Контрольні запитання .....</i>	<i>32</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>33</i>
<b>Розділ 2. Персептрон .....</b>	<b>34</b>
2.1. Архітектура нейрона типу персептрон .....	34
2.2. Принципи функціонування одонейронного персептрона.....	35
2.3. Процедура навчання одношарового персептрона.....	37
2.4. Одношаровий персептрон Ф. Розенблатта .....	43
<i>Приклади розв'язання задач .....</i>	<i>44</i>
<i>Контрольні запитання .....</i>	<i>52</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>52</i>
<b>Розділ 3. Лінійні перетворення нейронних мереж .....</b>	<b>54</b>
3.1. Лінійні перетворення та їх матричне зображення .....	54
3.2. Матричне подання лінійних перетворень.....	55
3.3. Зміна базису .....	57
3.4. Ключові поняття, пов'язані з лінійними перетвореннями .....	60
3.5. Зведення лінійних перетворень до діагонального вигляду .....	62
<i>Приклади розв'язання задач .....</i>	<i>63</i>
<i>Контрольні запитання .....</i>	<i>75</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>75</i>
<b>Розділ 4. Асоціативне контрольоване навчання Хебба .....</b>	<b>78</b>
4.1. Постулат навчання Хебба.....	78
4.2. Лінійний асоціатор і контрольоване правило навчання Хебба.....	79
4.3. Аналіз ефективності навчання Хебба .....	81
4.4. Псевдообернене правило.....	83

4.5. Застосування правила Хебба на практиці.....	85
4.6. Правило дельти як різновид навчання Хебба.....	86
<i>Приклади розв'язання задач</i> .....	87
<i>Контрольні запитання</i> .....	95
<i>Задачі для самостійного розв'язання</i> .....	95
<b>Розділ 5. Оптимізація характеристик функціонування мережі.....</b>	<b>97</b>
5.1. Сутність навчання на основі корекції похибок .....	97
5.2. Квадратичні функції та їх особливості.....	98
5.3. Визначення точок, які відповідають оптимальним значенням параметрів заданої мережі.....	102
<i>Приклади розв'язання задач</i> .....	114
<i>Контрольні запитання</i> .....	129
<i>Задачі для самостійного розв'язання</i> .....	130
<b>Розділ 6. Лінійна одношарова мережа.....</b>	<b>132</b>
6.1. Однонейронна мережа АДАЛІН .....	132
6.2. Середньоквадратична похибка мережі АДАЛІН .....	133
6.3. Алгоритм LMS .....	135
6.4. Аналіз збіжності алгоритму LMS для одного нейрона.....	136
6.5. Адаптивне фільтрування.....	139
<i>Приклади розв'язання задач</i> .....	144
<i>Контрольні запитання</i> .....	153
<i>Задачі для самостійного розв'язання</i> .....	153
<b>Розділ 7. Багатошаровий персептрон.....</b>	<b>156</b>
7.1. Можливості багатошарового персептрона.....	156
7.2. Алгоритм зворотного поширення похибки.....	158
7.3. Матриця Якобі. Чутливість мереж зі зворотним зв'язком.....	160
7.4. Багатошаровий персептрон з одним прихованим шаром як універсальний апроксиматор .....	165
<i>Приклади розв'язання задач</i> .....	167
<i>Контрольні запитання</i> .....	174
<i>Задачі для самостійного розв'язання</i> .....	174
<b>Розділ 8. Модифікації методу зворотного поширення .....</b>	<b>177</b>
8.1. Недоліки методу зворотного поширення .....	177
8.2. Евристичні модифікації методу SDBP .....	178
8.3. Методи, основані на алгоритмах оптимізації .....	180
<i>Приклади розв'язання задач</i> .....	188
<i>Контрольні запитання</i> .....	197
<i>Задачі для самостійного розв'язання</i> .....	197
<b>Розділ 9. Нейронні мережі на основі радіальних базисних функцій .....</b>	<b>199</b>
9.1. Задача інтерполяції.....	199

9.2. Мережа з використанням радіальних базисних функцій Гаусса.....	202
9.3. Мережа з використанням радіальних базисних функцій Гаусса як універсальний апроксиматор.....	206
9.4. Алгоритми навчання радіальних мереж на основі функцій Гаусса .....	206
9.5. Методи підбирання кількості базисних функцій .....	210
<i>Приклади розв'язання задач .....</i>	<i>214</i>
<i>Контрольні запитання .....</i>	<i>217</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>218</i>
<b>Розділ 10. Асоціативне неконтрольоване навчання.....</b>	<b>219</b>
10.1. Прості асоціативні мережі .....	219
10.2. Неконтрольоване правило навчання Хебба .....	221
10.3. Правило навчання Хебба зі зменшенням (затримкою) ваги.....	223
10.4. Одношарова мережа Інстар. Правило навчання нейрона Інстар.....	225
10.5. Правило Кохонена .....	229
10.6. Одношарова мережа Оутстар. Правило навчання Оутстар .....	229
<i>Приклади розв'язання задач .....</i>	<i>232</i>
<i>Контрольні запитання .....</i>	<i>239</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>239</i>
<b>Розділ 11. Конкурентні нейронні мережі .....</b>	<b>243</b>
11.1. Нейронна мережа Хеммінга .....	243
11.2. Конкурентна одношарова мережа .....	247
11.3. Самоорганізовані мережі: шар і мапа Кохонена .....	250
11.4. Мережа квантування векторів LVQ.....	258
<i>Приклади розв'язання задач .....</i>	<i>263</i>
<i>Контрольні запитання .....</i>	<i>271</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>271</i>
<b>Розділ 12. Мережа Гроссберга.....</b>	<b>275</b>
12.1. Система зору людини .....	275
12.2. Фундаментальна нелінійна шунтувальна модель .....	278
12.3. Двошарова конкурентна мережа Гроссберга.....	280
<i>Приклади розв'язання задач .....</i>	<i>288</i>
<i>Контрольні запитання .....</i>	<i>296</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>296</i>
<b>Розділ 13. Теорія адаптивних резонансів. Мережа ART1.....</b>	<b>298</b>
13.1. Мережа ART1: структура .....	298
13.2. Мережа ART1: перший шар .....	300
13.3. Мережа ART1: другий шар .....	304

13.4. Мережа ART1: підсистема орієнтації .....	306
13.5. Мережа ART1: алгоритм навчання.....	312
<i>Приклади розв'язання задач .....</i>	<i>314</i>
<i>Контрольні запитання .....</i>	<i>321</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>321</i>
<b>Розділ 14. Мережа Хопфілда.....</b>	<b>324</b>
14.1. Нелінійні динамічні системи у вигляді рекурентних мереж .....	324
14.2. Мережа Хопфілда дискретного часу .....	326
14.3. Модель Хопфілда неперервного часу.....	327
14.4. Функція Ляпунова та її особливості .....	329
14.5. Модель Хопфілда як асоціативна пам'ять .....	336
<i>Приклади розв'язання задач .....</i>	<i>341</i>
<i>Контрольні запитання .....</i>	<i>350</i>
<i>Задачі для самостійного розв'язання.....</i>	<i>350</i>
<b>Додатки.....</b>	<b>353</b>
<b>Додаток А.</b> Приклади функцій активації пакету прикладних програм Neural Network Toolbox системи MatLab.....	353
<b>Додаток Б.</b> Приклади реалізацій у середовищі MatLab.....	359
<b>Додаток В.</b> Деякі положення векторної алгебри у контексті нейронних мереж .....	366
<b>Додаток Г.</b> Деякі положення функціонального аналізу в контексті нейронних мереж.....	374
<b>Додаток Д.</b> Основи теорії стійкості нелінійних динамічних систем у контексті нейронних мереж.....	382
<b>Список літератури.....</b>	<b>393</b>
<b>Предметний покажчик .....</b>	<b>391</b>

## УМОВНІ СКОРОЧЕННЯ

АДАЛІН	– АДАптивний ЛІнійний Нейрон
БП	– багатошаровий персептрон
ГРБФ	– гаусівська радіальна базисна функція
ДНВ	– дискретизація навчальних векторів
ІТ	– інформаційні технології
КФ	– квадратична функція
ЛА	– лінійний асоціатор
ЛП	– лінійне перетворення
НМ	– нейронні мережі
РБФ	– радіальні базисні функції
ФА	– функції активації
ФЛ	– функція Ляпунова
СGBP	– Conjugate Gradient Backpropagation
LMBP	– Levenberg-Marquardt Backpropagation
LMS	– Least Mean Square
LTM	– Long-term memory
LVQ	– Learning Vector Quantization
МОБП	– Momentum Modification to Backpropagation
SDBP	– Steepest Descent Backpropagation
STM	– Short-term memory
SVD	– Singular Value Decomposition
VLBP	– Variable Learning Rate Backpropagation Algorithm

## ПЕРЕДМОВА

Основні тенденції розвитку кібернетики на початку третього тисячоліття можна виразити двома поняттями: «біологізація» і «гібридизація». Під *біологізацією* найчастіше розуміють побудову та дослідження моделей поведінки складних об'єктів і способів керування ними на основі імітації механізмів, які реалізуються природою у живих істотах. Такий підхід зумовлено тим, що у наш час «класичні» методи оброблення інформації сприймають як найпростіші реалізації універсальних способів функціонування біологічних об'єктів. Стрімке зростання обчислювальних потужностей і розвиток математичного апарату дозволили розв'язувати складні задачі. *Гібридизація*, у свою чергу, полягає у поєднанні різних методів і (або) моделей оброблення інформації про один і той самий об'єкт. Парадигма такого підходу ґрунтується на припущенні, що будь-яка складна штучна модель реального об'єкта завжди буде простішою за оригінал, тільки багатоаспектне його дослідження з подальшою інтеграцією отриманих результатів дозволить набути потрібних знань або наблизитися до оптимального розв'язку.

Сучасна теорія інтелектуальних обчислень пов'язана з побудовою та використанням штучних нейронних мереж (НМ) та гібридних систем, до складу яких входять елементи штучних НМ (наприклад, НМ з використанням генетичних алгоритмів для визначення топології мережі, нейронечіткі мережі тощо). Під *нейронними мережами* розуміють обчислювальні структури, які моделюють прості біологічні процеси, що зазвичай асоціюються із процесами людського мозку. Ці мережі набувають вигляду систем, здатних до навчання.

Актуальність досліджень у галузі прикладної математики, що спеціалізується на штучних НМ, підтверджується різними їх застосуваннями – це автоматизація процесів розпізнавання образів, адаптивне керування, апроксимація функціоналів, прогнозування, створення експертних систем, організація асоціативної пам'яті й багато інших додатків. Широке коло завдань, які вирішують НМ, не дозволяє нині створювати універсальні, потужні мережі, натомість розробляються спеціалізовані мережі, що функціонують за різними алгоритмами.

Незважаючи на істотні відмінності, НМ мають такі загальні ознаки: 1) основу кожної мережі складають порівняно прості (здебільшого однотипні) елементи, що імітують роботу нейронів мозку; 2) для них характерний принцип паралельного оброблення сигналів, якого досягають з'єднанням великої кількості нейронів у шари і різних шарів між собою (при цьому всі нейрони взаємодіють пошарово).



Цей навчальний посібник створено за матеріалами курсу «Нейронні мережі», який викладають у Національному технічному університеті України «Київський політехнічний інститут» під час підготовки майбутніх фахівців з інформаційних технологій. *Предметом вивчення навчальної дисципліни є штучні нейронні мережі. Метою навчальної дисципліни є формування у студентів таких здатностей:* 1) використовувати методи моделювання на основі НМ в інженерних розрахунках; 2) проектувати та розробляти програмне забезпечення у вигляді НМ (в тому числі й у середовищі MatLab з використанням Neural Networks Toolbox); 3) використовувати спроектовану НМ для розв'язання прикладних задач.

Після засвоєння навчальної дисципліни студенти мають продемонструвати такі результати навчання:

*знання:* 1) з теоретичних, методичних і алгоритмічних основ інформаційних технологій у вигляді НМ для їх використання під час розв'язання прикладних і наукових задач у сфері інформаційних систем і технологій; 2) моделей у вигляді НМ та інструментального засобу їх розробки – пакету Neural Networks Toolbox системи MatLab, уміння його застосовувати на всіх етапах життєвого циклу програмної реалізації НМ; 3) з теоретичних і практичних основ методики моделювання на основі НМ, проектування та експлуатації відповідної моделі;

*вміння:* 1) досліджувати залежність моделі у вигляді НМ від її параметрів; 2) реалізовувати алгоритми моделювання на основі НМ для дослідження характеристик і стану складних об'єктів; 3) реалізовувати та тестувати програмне забезпечення у вигляді НМ; 4) застосовувати мову специфікації НМ, інструментальний засіб Neural Networks Toolbox під час проектування та розробки НМ;

*досвід:* професійне володіння інформаційними технологіями у вигляді НМ.

Посібник є узагальненням відомого матеріалу з теорії НМ. Публікація книги є важливою тому, що нині бракує видань, у яких навчальний матеріал з теорії НМ викладено в доступній і зрозумілій формі.

У кінці кожного розділу посібника розміщено вправи для перевірки засвоєння навчального матеріалу.

Автори щиро вдячні рецензентам та колективу кафедри автоматизованих систем обробки інформації та управління НТУУ «КПІ» за змістовні поради і зауваження, які були враховані під час написання посібника та значно його покращили.

## ВСТУП

У наш час найпоширенішими є послідовні обчислення, але вони вичерпали свої технічні можливості, тому надзвичайно гостро постає проблема розвитку методів паралельного програмування і створення паралельних комп'ютерів. Штучні НМ як найсучасніший метод розвитку цього напрямку виникли на основі знань про функціонування нервової системи живих істот і є спробою використати процеси, які відбуваються в нервовій системі, для розроблення нових технологічних рішень. У центрі уваги теорії НМ перебувають моделі паралельно розподіленої обробки, в яких інформація обробляється за рахунок взаємодії великої кількості нейронів, кожен з яких передає сигнали збудження і гальмування іншим нейронам мережі.

Серед завдань, які можуть вирішити інтелектуальні системи на основі штучних НМ, можна виокремити такі:

1. *Класифікація образів або віднесення об'єкта до певного класу* (наприклад, розпізнавання мовного сигналу чи рукописного символу, класифікація сигналів електрокардіограми або клітин крові). Завдання полягає у визначенні одного з наперед заданих класів, до якого належить вхідний образ, зображений у вигляді вектора ознак.

Система встановлює подібність образів досліджуваних об'єктів і об'єднує схожі образи в один клас.

2. *Кластерний аналіз (або класифікація без учителя, оскільки, на відміну від завдань типу 1, класи апріорно не задано)* із поділом заданого набору об'єктів на класи схожих між собою об'єктів за визначеними критеріями.

3. *Апроксимація функцій* (використовується під час розв'язання задач моделювання): задано навчальну вибірку у вигляді пар даних вхід – вихід, що генерується невідомою функцією  $F$ , яку й треба визначити.

4. *Організація пам'яті*. У традиційних комп'ютерах звернення до пам'яті відбувається лише завдяки адресації комірок пам'яті, яка не залежить від змісту інформації, розміщеної в них. Під час обчислення адреси може бути допущена помилка і використана інша інформація. Асоціативна пам'ять (чи пам'ять, адресована за змістом) доступна за посиланням до заданого змісту, при цьому інформацію можна одержати навіть якщо частково відома вхідна інформація (або спотворений зміст).

5. *Прогнозування дискретних послідовностей* (наприклад, кількості захворювань певного типу на рівні міста, області, регіону). Нетривіальність прогнозування дискретних послідовностей (сукупності значень у фіксовані моменти часу) зумовлена тим, що, на відміну від добре

алгоритмізованих процедур інтерполяції, прогнозування вимагає екстраполяції даних про минуле на майбутнє, при цьому потрібно враховувати невідому інформацію про явище, яке лежить в основі процесу, що генерує дискретні послідовності.

6. *Оптимізація*. Завдання оптимізації полягає у визначенні розв'язку, який задовольняє систему обмежень і максимізує (або мінімізує) цільову функцію.

7. *Фільтрація* – виокремлення корисного сигналу за наявності фоновому шуму.

8. *Керування динамічною системою*. Поведінку динамічної системи задано у вигляді множини пар  $\{u(t), y(t)\}$ , де  $u(t)$  – вхідний керувальний вплив;  $y(t)$  – вихід системи у момент часу  $t$ . У системах керування з еталонною моделлю метою керування є визначення такого вхідного впливу  $u(t)$ , за якого система діє відповідно до бажаної траєкторії, яку описує еталонна модель.

# Розділ 1. ПОНЯТТЯ ПРО НЕЙРОННІ МЕРЕЖІ

## 1.1. Біологічний нейрон

Концепція обробки інформації за принципом паралелізму надає НМ особливу форму робастності (нечутливості до різних відхилень, неоднорідностей). Зображаючи кожну властивість групою нейронів, можна підвищити робастність мережі. Якщо обчислення розподілені між багатьма нейронами, то зашумлений або неповний вхідний сигнал все одно можна розпізнати.

Мозок людини складається з нервових клітин (*нейронів*), сполучених між собою синапсами. Кожен нейрон отримує інформацію через дендрити і передає її далі через аксон, кінець якого розгалужується на тисячі синапсів (рис. 1.1). Нейрони взаємодіють за допомогою серії імпульсів, які тривають декілька мілісекунд. Кожний імпульс – це сигнал, який передається із частотою від декількох до сотень герців. Цей процес повільніший, ніж у сучасних комп'ютерах, але водночас людський мозок швидше може обробляти інформацію (наприклад, розпізнавати звуки). Вивчення механізмів функціонування окремих нейронів та їх взаємодії важливе для пізнання процесів, які відбуваються у нервовій системі (пошуку, передавання та оброблення інформації).

Як і в будь-якої клітини, у нейрона є тіло, яке називають сомою, з ядром і стандартним набором органел (складових частин клітини).

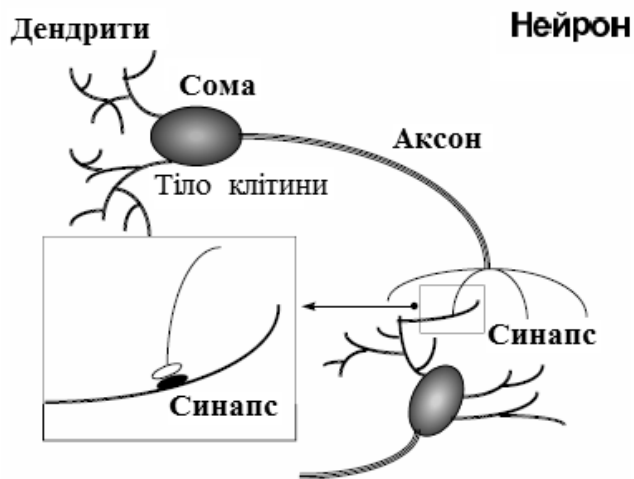


Рис. 1.1. Спрощена структура біологічної нервової клітини (нейрона)

З нейрона виходять численні відростки, які відіграють ключову роль у його взаємодії з іншими нервовими клітинами. Розрізняють такі два типи відростків (рис. 1.1): 1) тонкі, гіллясті дендрити; 2) товстіший аксон, кінець якого має вигляд гілля. Вхідні сигнали надходять у клітину через синапси, а вихідний сигнал передається аксоном через його численні нервові закінчення (колатерали), які контактують із сомою і дендритами інших нейронів, утворюючи чергові синапси.

Синапси приєднують до клітини виходи інших нейронів і можуть розміщуватись як на дендритах, так і безпосередньо на тілі клітини.

Отже, серед складових біологічного нейрона виділяють:

1. *Дендрити* – вхідні волокна, які забезпечують нейрон інформацією.  
2. *Аксон* – довге вихідне волокно, яке передає інформацію м'язовим волокнам та іншим нейронам (у нейронів не може бути більше одного аксона, іноді його немає зовсім).

3. *Синапс* – місце контакту нервових волокон – передає імпульси від клітини до клітини (ці імпульси майже завжди однонаправлені, тому за напрямом передавання імпульсу розрізняють постсинаптичні та передсинаптичні нервові процеси).

4. *Колатерали* – нервові закінчення аксонів, які контактують із собою і дендритами інших нейронів, утворюючи чергові синапси.

Передавання сигналів усередині нервової системи – дуже складний електрохімічний процес [15]. Спрощено можна вважати, що передавання нервового імпульсу між двома клітинами пов'язане з виділенням особливих хімічних субстанцій – нейромедіаторів, які формуються під впливом подразників, що надходять від синапсів. Ці субстанції впливають на клітинну мембрану, викликаючи зміну її енергетичного потенціалу пропорційно до кількості нейромедіатора, який потрапляє на мембрану. Отже, синапс перетворює передсинаптичний електричний сигнал у хімічний, а після цього у постсинаптичний. Синапси різняться один від одного розмірами й можливостями концентрації нейромедіатора поблизу своєї оболонки, тому імпульси однакового розміру, які надходять на входи нервової клітини через різні синапси, можуть збуджувати її більшою чи меншою мірою. Мірою збудження клітини вважають рівень поляризації її мембрани, який залежить від загальної кількості нейромедіатора, виділеного на всіх синапсах.

Кожному входу клітини можуть відповідати числові значення вагових коефіцієнтів, пропорційні кількості нейромедіатора, який одноразово виділяється на відповідному синапсі. У математичній моделі нейрона вхідні сигнали треба множити на ці коефіцієнти для того, щоб коректно враховувати вплив кожного сигналу на стан нервової клітини. Синаптичну вагу зазвичай зображують за допомогою цілих чисел: якщо значення синаптичної ваги додатне, синапс має збуджувальну дію, а якщо від'ємне – гальмівну, що перешкоджає збудженню клітини іншими сигналами. У результаті надходження вхідних імпульсів на конкретні синапси та вивільнення деякої кількості нейромедіатора відбувається електричне збудження нервової клітини (рис. 1.2).

Якщо відхилення від стану електричної рівноваги невелике або баланс збуджень і гальмувань від'ємний, то клітина самостійно переходить у початковий стан і на її виході не реєструють ніяких змін. У цьому разі вважають, що рівень збудження клітини був нижчим за поріг її активації. Якщо ж сума

збуджень і гальмувань перевищила поріг активації клітини, то значення вихідного сигналу починає швидко зростати, набуваючи характерного вигляду нервового імпульсу, який передається аксоном на інші нейрони. Величина цього сигналу не залежить від рівня перевищення цього порогу. Після виконання своєї функції нейромедіатор видаляється: або всмоктується клітиною, або розкладається, або переміщується за межі синапса.

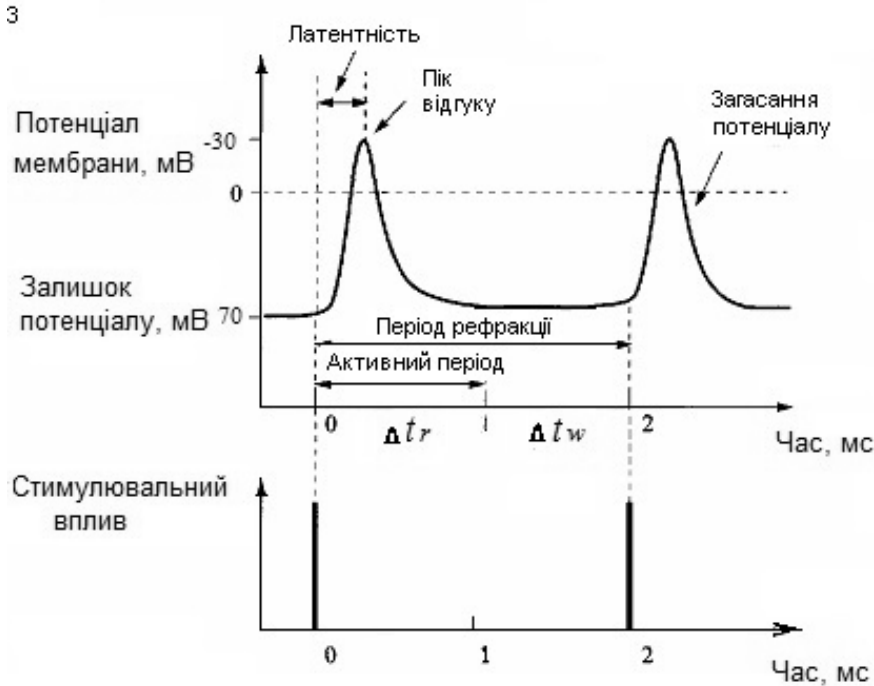


Рис. 1.2. Ідеалізована версія дії двох послідовних потенціалів (латентність – затримка між стимулом і реакцією)

Одночасно з генерацією нервового імпульсу в клітині запускається процес рефракції, який полягає у стрімкому зростанні порогу її активації, у результаті чого нейрон втрачає здатність формувати черговий сигнал навіть у разі сильного збудження. Такий стан зберігається протягом деякого часу  $\Delta t_r$ , який називають періодом абсолютної рефракції. Після закінчення цього періоду настає відносна рефракція  $\Delta t_w$  і поріг активації набуває початкового значення. Протягом цього часу клітину можна активувати лише завдяки більш інтенсивному збудженню. У природних процесах виконується співвідношення  $\Delta t_w \gg \Delta t_r$ .

Кількість нервових клітин, які взаємодіють одна з одною, надзвичайно велика. Вважають, що людський мозок містить близько  $10^{11}$  нейронів, кожен з яких виконує порівняно примітивні функції: підсумовування вагових коефіцієнтів вхідних сигналів і порівняння отриманої суми з пороговим значенням. Кожний нейрон має свої вагу та значення порогу, що визначаються його розміщенням та виконуваним завданням. Через величезну кількість нейронів і міжнейронних зв'язків (до 1000 входів

у кожному нейроні) похибка під час спрацювання окремого нейрона залишається непомітною в загальній масі взаємодійних клітин.

*Головна відмінність НМ (як систем на основі паралельних обчислень) від звичайних електронних систем, створених людиною, полягає у тому, що мережа проявляє високу стійкість до перебоїв: це «стабільна» система, в якій окремі перебої істотно не впливають на результати її функціонування. Жодна сучасна інформаційна технологія (ІТ) не дозволяє побудувати штучну НМ, яка б моделювала мозок людини. Наразі вивчення та копіювання біологічних нервових систем дозволяє сподіватися на створення нового покоління електронних пристроїв, які мають аналогічні характеристики.*

## 1.2. Модель штучного нейрона

Розглянемо спрощену математичну модель штучного нейрона та пояснимо, яким чином штучні нейрони можуть формувати мережі різної архітектури, проілюструємо роботу таких мереж на прикладах. У наш час немає стандартної математичної форми запису архітектури НМ.

**Нейрон з одним входом (вхідним параметром)** [14]. Вхідними параметрами моделі нейрона (рис. 1.3) є скалярні значення входу  $p$  з коефіцієнтом ваги  $w$ , які формують значення  $wp$ , та число 1, яке множиться на значення зсуву  $b$ , вони передаються на суматор. Вихід суматора (вхід мережі)  $u$  передається функції активації (ФА)  $f$ , яка формує вихід нейрона  $y$  у вигляді скалярного значення  $y$  й може бути як лінійною, так і нелінійною.

Вихід нейрона обчислюють за формулою  $y = f(wp + b)$ . Скалярні параметри  $w$  і  $b$  зазвичай налаштовуються на дані задачі за визначеним правилом навчання так, щоб нейрон, який відображає відношення вхід–вихід, досягнув заданої мети. Відповідно до біологічного нейрона вага  $w$  відповідає довжині синапсу, суматор разом із функцією активації зображує клітинне тіло нейрона, а вихід нейрона  $y$  зображує сигнал, який надходить на аксон.

Розглянемо означення найбільш поширених ФА відповідно до системи MatLab. У дод. А наведено приклади ФА, які використовує пакет прикладних програм Neural Network Toolbox системи MatLab [5].

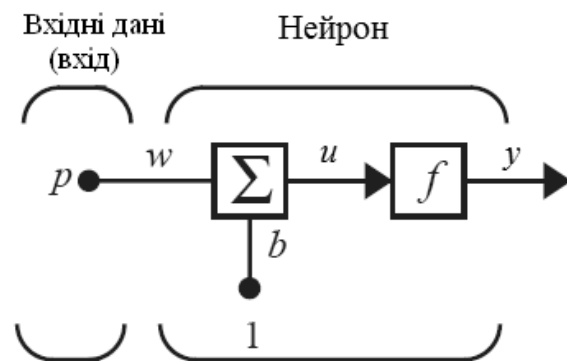


Рис. 1.3. Модель нейрона з одним входом:  $y = f(wp + b)$

**Означення 1.1.** Графік ФА з різким порогом (або одинарним стрибком)  $y = f(u) = \begin{cases} 1, & \text{якщо } u \geq 0; \\ 0, & \text{якщо } u < 0, \end{cases}$  а також вплив на цей графік значень вагового коефіцієнта  $w$  та зсуву  $b$  зображено на рис. 1.4.

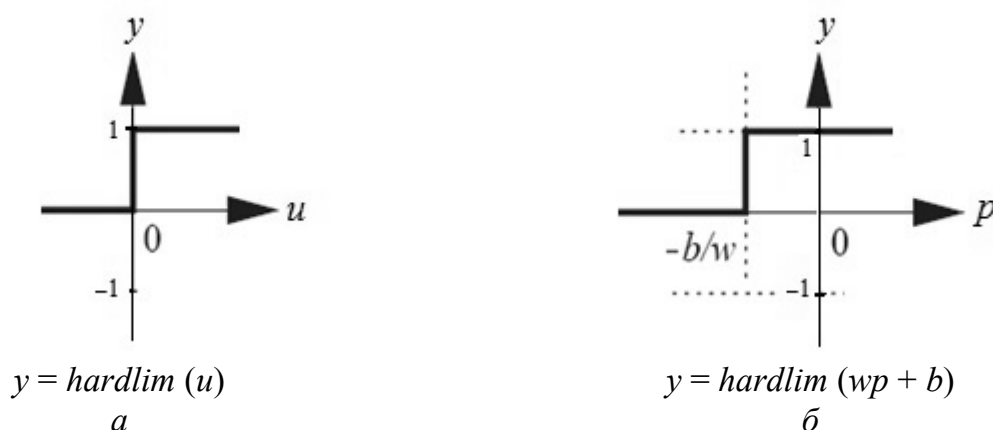


Рис. 1.4. Графіки ФА з різким порогом для нейронів з одним входом

**Означення 1.2.** Графік лінійної ФА  $y = f(u) = u = wp$  зображено на рис. 1.5.

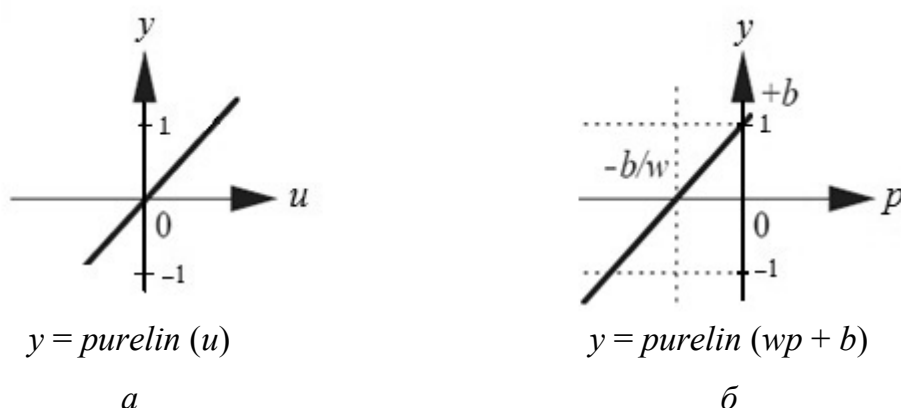


Рис. 1.5. Графіки лінійної ФА для нейронів з одним входом

**Означення 1.3.** Графік сигмоїдної ФА  $f(u) = \frac{1}{1 + e^{-u}}$ , аргументом якої є будь-яке дійсне число з інтервалу  $(-\infty; +\infty)$ , а її значення належать інтервалу  $(0; 1)$ , зображено на рис. 1.6.

Сигмоїдну ФА найчастіше використовують у багатошарових мережах, які реалізують режим навчання на основі алгоритму зворотного поширення похибки.

**Нейрон із декількома входами [14].** У більшості випадків нейрон має більше одного входу. Модель нейрона із  $R$  входами зображено на рис. 1.7. Користувач мережі задає вектор входу  $\mathbf{p} = [p_1 \dots p_R]^T$ , якому



відповідає вектор вагових коефіцієнтів  $\mathbf{w} = [w_{11} \dots w_{1R}]^T$ . Зсув нейрона  $b$  підсумовується зі зваженими входними елементами та формує вхід мережі  $u$ :

$$u = w_{11}p_1 + \dots + w_{1R}p_R + b. \quad (1.1)$$

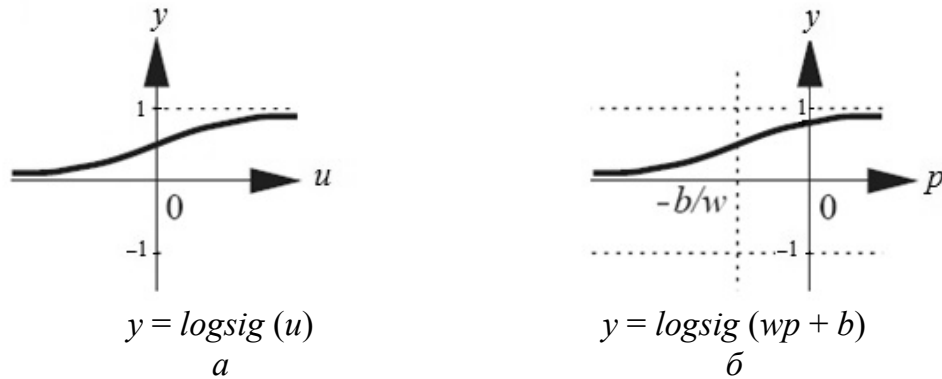


Рис. 1.6. Графіки сигмоїдної ФА для нейронів з одним входом

Вираз (1.1) можна записати у матричній формі:  $u = \mathbf{w}^T \mathbf{p} + b$ , а вихід нейрона – у вигляді  $y = f(\mathbf{w}^T \mathbf{p} + b)$ .

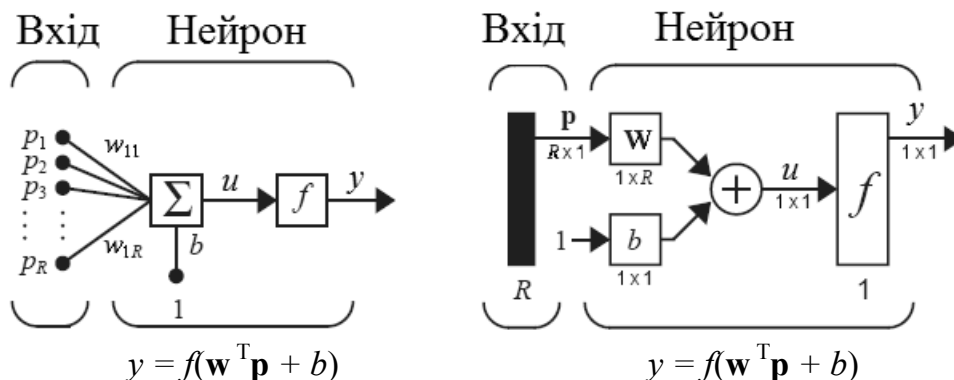


Рис. 1.7. Модель нейрона із  $R$  входами: повна та скорочена форми зображення (відповідно до системи MatLab)

### 1.3. Архітектура мережі

За кількістю шарів розрізняють такі види мереж [14]: які складаються з одного шару нейронів та з декількох шарів нейронів.

Мережі, які складаються з одного шару нейронів. Одношарову НМ, яка складається із  $S$  нейронів, зображено на рис. 1.8. У  $i$ -го нейрона є зсув  $b_i$ , суматор, функція активації  $f$  та вихід  $y_i$  ( $i = 1, 2, \dots, S$ ). Усі нейрони разом формують вектори входу  $\mathbf{p} = [p_1 \dots p_R]^T$ , зсуву  $\mathbf{b} = [b_1 \dots b_S]^T$  та виходу  $\mathbf{y} = [y_1 \dots y_S]^T$ . Одношарова мережа включає вагову матрицю  $\mathbf{W}$ , суматор, вектор зсуву  $\mathbf{b}$ , ФА та вектор виходу  $\mathbf{y}$ . Зазначимо, що кожен із  $R$  елементів входного вектора  $\mathbf{p}$  пов'язаний з окремим нейроном і вагові коефіцієнти тепер утворюють матрицю  $\mathbf{W}$ , яка має  $S$  рядків:

$\mathbf{W} = \begin{bmatrix} w_{11} & \dots & w_{1R} \\ \dots & \dots & \dots \\ w_{S1} & \dots & w_{SR} \end{bmatrix}$ . Індекси елементів вагової матриці  $\mathbf{W}$  відповідають

такій умові: перший індекс вказує на номер нейрона, з яким встановлюється зв'язок; другий індекс – на джерело сигналу (вхід), який отримує нейрон. Наприклад, елемент  $w_{32}$  означає ваговий коефіцієнт, який зображує зв'язок третього нейрона з другим входом. Комбінуючи дві мережі, які розміщені паралельно і мають однакові входи, можна визначити шар нейронів, які мають різні ФА.

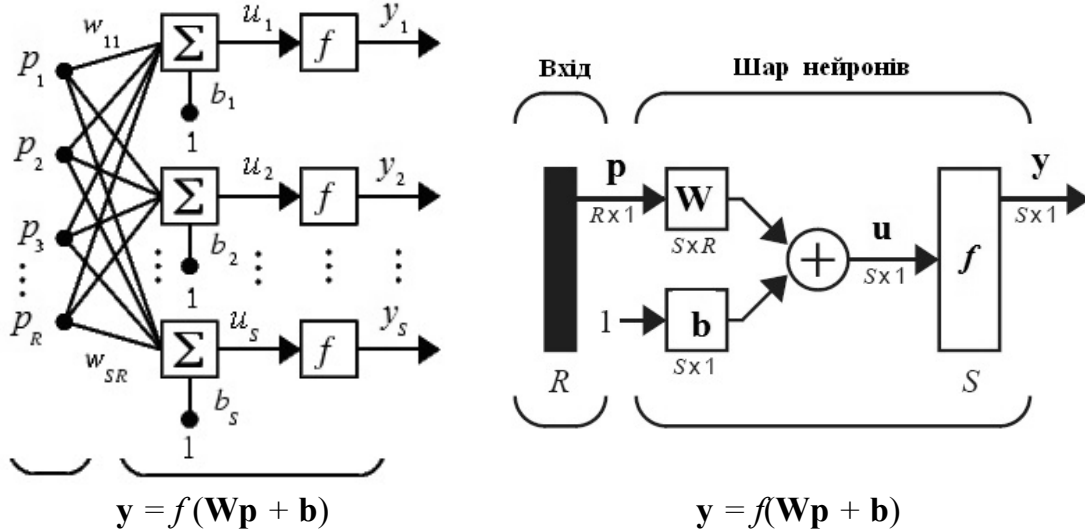


Рис. 1.8. Одношарова мережа, яка складається із  $S$  нейронів (повна та скорочена форми зображення відповідно до системи MatLab)

Мережі, які складаються з декількох шарів нейронів. Розглянемо мережу, яка складається з декількох шарів нейронів. Кожен шар має власну вагову матрицю, вектори зсуву, входу і виходу мережі. Щоб розрізняти шари нейронів, введемо такі позначення:  $\mathbf{W}^i$  – вагова матриця  $i$ -го шару;  $R$  – довжина вектора входу  $\mathbf{p}$ ;  $S^i$  – кількість нейронів у  $i$ -му шарі (різні шари можуть мати різну кількість нейронів); виходи першого шару є входом для другого й т. д. Таким чином, другий шар можна інтерпретувати як одношарову мережу, в якій є  $R = S^1$  входів,  $S = S^2$  нейронів, а розмірність вагової матриці  $\mathbf{W}^2$  становить  $[S^1 \times S^2]$ . Входом другого шару є вектор  $\mathbf{y}^1$ , а виходом –  $\mathbf{y}^2$ . Шар, який формує вихід мережі, називають *вихідним*, інші шари називають *прихованими*. У мережі з трьома шарами, зображеній на рис. 1.9, є вхідний шар (вхід), вихідний шар (третій) і два приховані шари (перший і другий).

Архітектура багатошарової мережі визначається умовами завдання, яке розв'язує мережа: кількість входів відповідає кількості вхідних змінних, кількість виходів – кількості нейронів у вихідному шарі. Зазначимо,

що мережі зі зсувом більш ефективні порівняно з мережами без зсуву. Багатошарові НМ (які, зазвичай, використовують два шари) можуть вирішувати набагато ширший клас завдань, ніж мережі з одним шаром.

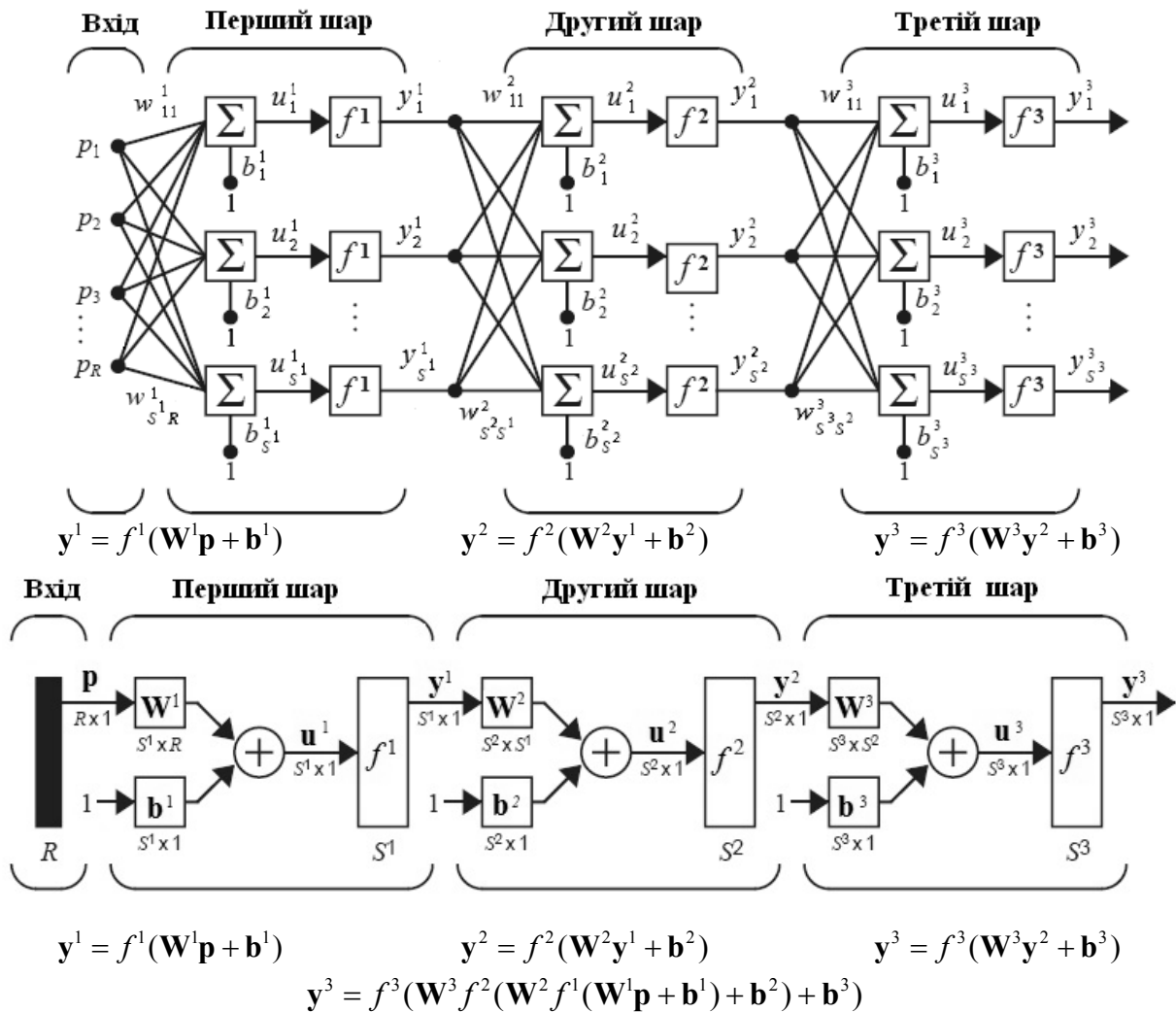


Рис. 1.9. Мережа з трьома шарами (повна та скорочена форми зображення відповідно до системи MatLab)

Найбільш часто використовують багатошаровий перцептрон (БП) і НМ з радіальними базисними функціями, або RBF-мережі. Мережі на основі радіальних базисних функцій і БП є універсальними апроксиматорами в тому сенсі, що за великої кількості елементів у прихованому шарі вони здатні досить добре апроксимувати будь-яку неперервну функцію на компактній підмножині з  $R$ -вимірного евклідового простору  $\mathcal{R}^R$ .

Важливим фактором описання нейрона є вибір *стратегії* (правила або алгоритму) *навчання*. Під правилом навчання розуміють ітераційну процедуру зміни значень ваги та зсуву мережі з метою навчити мережу вирішувати поставлене перед нею завдання. Оскільки найбільш часто використовуються структури, що містять два шари, один з яких є прихованим,

то завдання визначення структури фактично зводиться до визначення кількості нейронів  $S$  у прихованому шарі та взаємозв'язків між нейронами окремих шарів. Мета налаштування (навчання) мережі полягає в отриманні такої поверхні апроксимації відображення вхід–вихід модельованої системи, яка відповідає даним вимірювань про систему і дає інформацію про характер її функціонування. При цьому має залишатися справедливим таке загальне твердження про структуру мережі: чим складнішою є поверхня виконуваного системою відображення  $P \rightarrow Y$ , тим більша кількість нейронів прихованого шару потрібна для її моделювання (рис. 1.10).

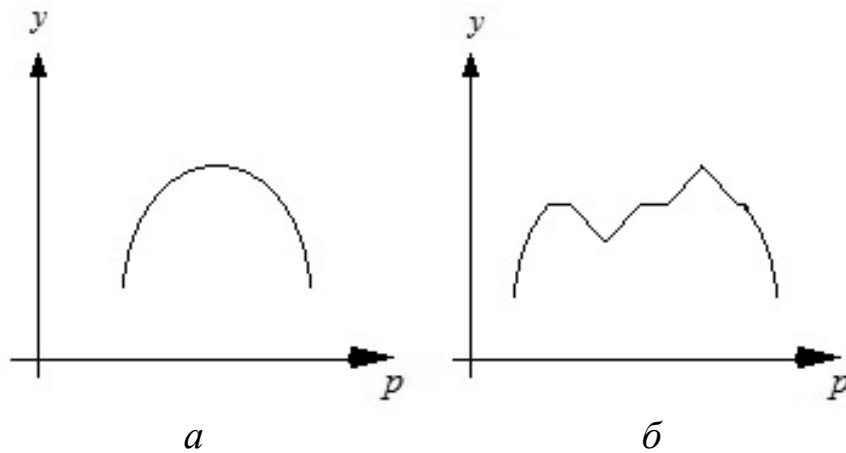


Рис. 1.10. Залежність між складністю поверхні відображення  $P \rightarrow Y$ , яку реалізує модельована система, і кількістю нейронів НМ:  
 а – проста поверхня (прихований шар містить невелику кількість нейронів);  
 б – складна поверхня (прихований шар містить більшу кількість нейронів)

На жаль, під час моделювання реальних систем інформація про рівень складності поверхні найчастіше виявляється неповною, особливо для систем з багатьма входами. Зазвичай наявна лише інформація про результати вимірювання вхідних значень і вихідних, які відповідають вхідним (при цьому ця інформація містить численні помилки вимірювання і зазнає впливу вхідних параметрів, які не були враховані у процесі моделювання). Для систем, які мають більше двох входів, модель неможливо уявити у графічній формі (що дозволяє оцінити складність поверхні), і тому використовується *метод проб*.

На початку процесу моделювання кількість нейронів можна покласти рівною середньому геометричному значенню кількості входів  $r$  і виходів  $l$ :  $S \approx \sqrt{rl}$ . З урахуванням точності спроектованої мережі кількість нейронів можна відповідним чином скоригувати. Перед початком процесу налаштування мережі множину  $U_m$ , що містить вимірювання даних вхід–вихід про систему, слід розділити на дві підмножини – множину  $U_{tr}$  даних навчання і множину  $U_{ts}$  даних тестування (множини  $U_{tr}$  і  $U_{ts}$

не повинні перетинатися):  $U_m = U_{tr} + U_{ts}$ ,  $U_{tr} \neq U_{ts}$ . Множину  $U_{ts}$  використовують для тестування мережі, навченої на основі множини  $U_{tr}$ , вона повинна містити дані, відібрані з множини вимірювань  $U_m$  відповідно до принципу рівномірного розподілу даних у просторі вхідних даних  $P$  (рис. 1.11).

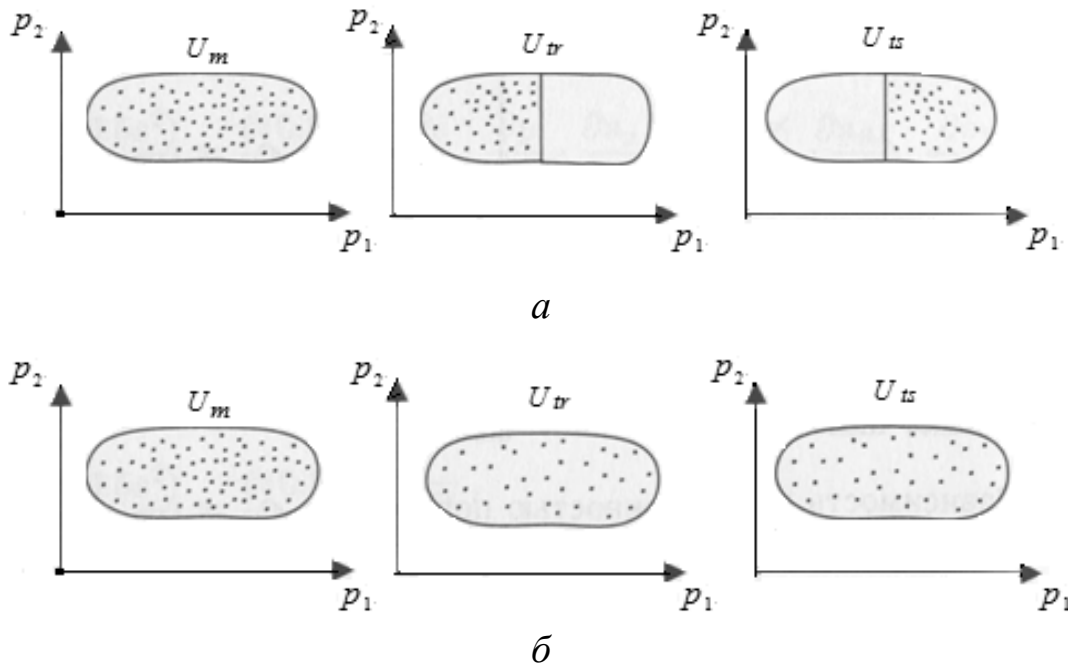


Рис. 1.11. Ілюстрація проблеми розбиття множини вимірювань  $U_m$  на  $U_{tr}$  і  $U_{ts}$ :  
 а – неправильне розбиття; б – правильне розбиття

У випадку, зображеному на рис. 1.11, а, після налаштування мережа виконуватиме точне моделювання системи лише на даних у лівій частині вхідного простору, а використання для тестування даних із правої частини цього простору спричинятиме значну похибку (у розглянутому випадку мережа не мала можливості «ознайомитися» з модельованою системою на всьому просторі її функціонування).

У випадку, зображеному на рис. 1.11, б, як для навчання, так і для тестування мережі використовуються дані, розподілені на всьому просторі функціонування системи. У цьому випадку множина даних тестування буде забезпечувати об'єктивне, коректне оцінювання точності мережі. За результатами аналізу початкової структури необхідно коригувати кількість нейронів  $S$  у прихованому шарі доти, поки не буде зафіксовано покращення результатів тестування, тобто поки середня похибка мережі на множині тестових даних  $U_{ts}$  або на загальній множині даних  $U_m$  не зменшиться.

В окремих випадках навчання виконується без використання тестових даних, особливо якщо множина даних невелика. Процес навчання

і коригування структури триває до досягнення функцією похибки мінімального значення (за умови, що мережа не перенавчилася).

Зміна структури НМ може виконуватися з використанням двох методів: конструктивного і деструктивного. *Конструктивний метод* передбачає, що процес навчання починається з невеликої кількості нейронів, яка поступово збільшується, поки не буде досягнуто оптимального результату. У разі використання *деструктивного методу* навчання розпочинається з великої кількості нейронів, яка потім поступово зменшується.

Основне завдання, пов'язане з навчанням мережі, полягає в тому, щоб забезпечити коректність узагальнення мережею даних й уникнути її перенавчання. Недостатньо навчена мережа демонструє надмірно високий ступінь узагальнення даних. Похибка мережі є високою як для множини даних навчання  $U_{tr}$ , так і для множини тестових даних  $U_{ts}$ . Причиною цього можуть бути, наприклад, занадто мала кількість нейронів або недостатність навчання. Для відповідної моделі НМ зазвичай характерний простий вигляд відображення поверхні  $P \rightarrow Y$ . Середня похибка перенавчання мережі є надто малою, а її поверхня відображення дуже точно проходить через точки, що відповідають даним вимірювань, однак між цими точками поверхня може докорінно відрізнятись від реальної поверхні системи моделювання. Розпізнати випадок, пов'язаний з перенавчанням мережі, легко, оскільки при цьому мережа демонструє дуже низьку похибку на навчальній множині і високу – на тестовій множині. Причиною перенавчання може бути як занадто велика для певного рівня складності моделюваної системи кількість нейронів, так і занадто малий обсяг навчальних даних. Обсяг даних не має бути меншим від кількості параметрів, які підлягають налаштуванню. Якщо даних недостатньо, налаштування деяких параметрів здійснюється випадковим чином і поверхня моделі між навчальними елементами має непередбачуваний характер.

Отже, мережа, яка виконує апроксимацію множини навчальних даних  $U_{tr}$ , не обов'язково є оптимальною мережею. «Оптимальною» є мережа, яка навчена на множині навчальних даних і забезпечує мінімальну похибку на всій множині результатів вимірювань  $U_m$ .

## 1.4. Перша модель штучної нейронної мережі та її особливості

У 1943 р. У. Маккаллок та У. Піттс запропонували одну з перших моделей штучних нейронів [12]. На вхід певного нейрона подається вектор  $\mathbf{p} = [p_1 \dots p_R]^T$ , де вхідні сигнали  $p_j$  ( $j = 1, 2, \dots, R$ ), які набувають значення «1» або «0», підсумовуються з урахуванням відповідних вагових коефіцієнтів  $w_{1j}$ , при цьому вихідний сигнал нейрона  $y$  визначають за формулою  $y = f\left(\sum_{j=1}^R w_{1j} p_j + b\right)$ , де  $b$  – зсув нейрона;  $f$  – ФА вигляду

$$y = f(u) = \begin{cases} 1, & \text{якщо } u \geq 0; \\ -1, & \text{якщо } u < 0. \end{cases}$$

У. Маккаллок та У. Піттс показали, що на основі запропонованих ними нейронів можна побудувати будь-яку логічну функцію, а мережі цих нейронів дозволяють обчислити будь-яку арифметичну функцію. Отже, система таких нейронів реалізує повну обчислювальну модель.

Приклади обчислення логічних функцій «І» та «АБО» за допомогою нейронів Маккаллока–Піттса [6] зображено на рис. 1.12. Кожний із цих нейронів має вхід  $\mathbf{p} = [p_1 \ p_2 \ 1]^T$  з трьома елементами:  $p_1$  і  $p_2$  задають аргументи функцій, а третій завжди дорівнює одиниці. Вагові коефіцієнти становлять відповідно «1», «1» та «-2», тому для будь-яких значень  $p_1$  і  $p_2$  нейрон обчислює значення:  $p_1 + p_2 - 2$  (рис. 1.12, а). Якщо це значення менше нуля, вихідним значенням нейрона є «-1», в іншому випадку – одиниця. Такий нейрон обчислює значення функції  $p_1 \wedge p_2$  (табл. 1.1). Аналогічно можна побачити, що другий нейрон (рис. 1.12, б) обчислює значення функції  $p_1 \vee p_2$ .

Попри те, що У. Маккаллок та У. Піттс продемонстрували можливості нейронів, реально ними зацікавилися лише після розробки алгоритмів навчання, які можна було використовувати на практиці.

З часом модель нейрона Маккаллока–Піттса була розширена за рахунок формування шарів нейронів і розробки алгоритмів їхньої взаємодії. Першу версію такої нейроподібної структури назвали *персептроном*.

Таблиця 1.1

Модель Маккаллока–Піттса для обчислення логічної функції «І»:  $p_1 \wedge p_2$

$p_1$	$p_2$	$(p_1 + p_2 - 2)$	Вихід
1	1	0	1
1	0	-1	-1
0	1	-1	-1
0	0	-2	-1

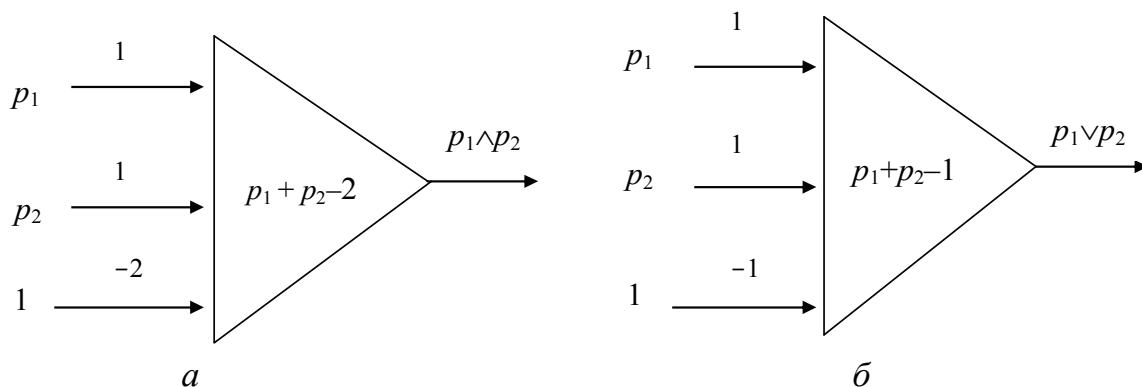


Рис. 1.12. Обчислення логічних функцій «І» (а) та «АБО» (б) за допомогою нейронів Маккаллока–Піттса

## 1.5. Класифікація нейронних мереж

Різноманіття структур НМ дозволяє їх класифікувати з урахуванням різних критеріїв. Окрім поділу на одношарові та багатошарові відомі інші види класифікації. Серед них виділимо такі.

1. Мережі, які реалізують *контрольоване навчання* (з учителем) та *неконтрольоване навчання* (без учителя).

*Під час навчання з учителем* задається множина прикладів необхідної поведінки мережі, яка називається множиною навчання:  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , де  $\mathbf{p}_1, \dots, \mathbf{p}_Q$  – входи нейрона;  $\mathbf{t}_1, \dots, \mathbf{t}_Q$  – бажані (цільові) виходи. Правило навчання використовують для налагодження ваги та зсуву нейрона таким чином, щоб наблизити значення виходу до цільового значення. Навчання з учителем розглядають як задачу апроксимації, яка полягає у визначенні функції  $F(\mathbf{p}, \mathbf{w})$ , що найкраще наближає бажану (цільову) функцію  $f(\mathbf{p})$  і реалізується у вигляді НМ.

*Під час навчання без учителя* зміна ваги та зсуву пов'язана тільки зі зміною входів мережі. У цьому випадку цільові виходи в явному вигляді не задаються. Головна позитивна риса навчання без учителя – це його самоорганізація, обумовлена, найчастіше, використанням зворотних зв'язків.

Відомі різні алгоритми навчання, які, однак, поділяють на два великі класи: *детерміністичні* та *стохастичні*. У першому з них налаштування ваги має вигляд чіткої послідовності дій, у другому воно виконується на основі дій, що підкоряються деякому випадковому процесу (наприклад, правило навчання Больцмана). Серед детерміністичних методів навчання можна виділити [12] *навчання на основі корекції похибок*, *навчання на основі пам'яті*, *навчання Хебба*, *конкурентне навчання*.

2. *Статичні і динамічні мережі*. У статичних НМ зміна параметрів системи відбувається за певним алгоритмом у процесі навчання. Після навчання параметри мережі не змінюються. У динамічних НМ відобра-



ження зовнішньої інформації та її обробка здійснюється у вигляді деякого динамічного процесу (тобто процесу, що залежить від часу).

3. *Бінарні та аналогові НМ.* Перші з них оперують двійковими сигналами і вихід кожного нейрона може набувати тільки двох значень: логічний нуль («загальмований» стан) і логічна одиниця («збуджений» стан). До цього класу мереж належить і персептрон, тому що виходи його нейронів, що формуються ФА з різким порогом, дорівнюють або нулю, або одиниці. В аналогових мережах вихідні значення нейронів здатні набувати неперервних значень, що пов'язано з використанням сигмоїдної функції активації нейронів персептрона.

4. *Асинхронні та синхронні НМ.* У першому випадку у визначений момент часу свій стан змінює лише один нейрон, у другому стан змінюється відразу у цілої групи нейронів, зазвичай у всього шару, на основі ітераційного алгоритму виконання однотипних дій над нейронами.

## **1.6. Подання знань засобами нейронних мереж**

Успіх розв'язання будь-якої задачі «інтелектуальними» системами залежить від форми подання знань. Це стосується і НМ, які утворюють окремий клас інтелектуальних систем.

Нейронна мережа – одна із форм, в якій за допомогою процесу навчання можна закодувати емпіричні знання про фізичні явища або навколишнє середовище (у вигляді набору вимірів, які характеризують визначене явище). Основним завданням НМ є найкраща з усіх можливих форм навчання для формування моделі навколишнього світу і подальшого вирішення поставлених завдань. Знання про світ поєднують два типи інформації: 1) відомий стан навколишнього світу, підтверджений наявними достовірними фактами (таку інформацію називають *апріорною*); 2) спостереження за навколишнім світом (зазвичай у вигляді вимірювань), отримані за допомогою сенсорів, адаптованих під конкретні умови, в яких має функціонувати певна НМ. Зазвичай останні вимірювання значно зашумлені, що може стати джерелом помилок. Попри це отримані вимірювання формують інформацію, яка використовується для побудови прикладів, необхідних для навчання НМ. Приклади можуть бути маркованими й немаркованими: у маркованих прикладах вхідному сигналу відповідає бажаний відгук, немарковані приклади складаються з декількох різних реалізацій одного вхідного сигналу. Множину пар сигналів вхід–вихід, що складається із вхідного сигналу і бажаного виходу, який відповідає йому, називають *даними навчання*.

Розглянемо приклад завдання, яке полягає у розпізнаванні цифр. Нехай вхідний сигнал (зображення) має вигляд матриці, утвореної з чорних і білих

пікселів, яка відображує на білому фоні одну з десяти рукописних цифр. Бажаним виходом мережі є конкретна цифра, зображення якої подається як вхідний сигнал. Зазвичай навчальна вибірка складається з великої кількості рукописних цифр на білому фоні. Наведемо алгоритм створення НМ за умови наявності такого набору прикладів.

1. *Етап навчання.* Вибирають архітектуру НМ, в якій розмір вхідного шару відповідає кількості пікселів на малюнку, а вихідний шар містить десять нейронів, які відповідають цифрам. Після цього налаштовують вагові коефіцієнти мережі на основі навчальної множини.

2. *Етап узагальнення.* Ефективність навчання мережі перевіряють (тестують) на множині прикладів, які мають відрізнятися від тих, що були використані для навчання. При цьому на вхід мережі подають зображення, для якого відомий цільовий вихід мережі. Ефективність навчання мережі перевіряють за допомогою порівняння результатів розпізнавання з реальними даними.

Створення НМ, як і розроблення класичних методів обробки інформації (наприклад, для вирішення завдань класифікації), ґрунтується безпосередньо на реальних даних і полягає в першочерговому формулюванні математичної моделі досліджуваного середовища, наступній її верифікації на реальних даних та в розробленні на основі цієї моделі класифікатора. Таким чином, НМ не тільки реалізують повноцінну модель середовища, але й забезпечують обробку даних.

Набір даних, використовуваний для навчання НМ, повинен містити як позитивні, так і негативні приклади. Приміром, для задачі пасивної ехолокації позитивні приклади включають сигнали, відбиті від об'єкта, який цікавить дослідника (приміром, підводного човна). Однак у реальному середовищі на показання радара впливають і морські об'єкти, які випадково потрапили в зону сигналу (негативні приклади).

У мережі заданої архітектури знання про довкілля подаються у вигляді множини вільних параметрів мережі (тобто значень синаптичної ваги і зсувів). Подання знань на основі НМ – дуже складна проблема, наразі С. Хайкін виділяє такі *три загальні правила* [12].

**Правило 1.** Подібні вхідні сигнали від схожих класів мають формувати єдине подання в НМ та належати до одного класу.

Відомо декілька методів визначення ступеня подібності вхідних сигналів, розглянемо два з них.

*Перший метод.* Ступінь подібності визначають на основі відстані Евкліда. Наприклад, довільний дійсний вектор  $\mathbf{x}_i$  складеться з  $m$  елементів:  $\mathbf{x}_i = [x_{i1} \dots x_{im}]^T$ . Вектор  $\mathbf{x}_i$  визначає деяку точку в  $m$ -вимірному Евклідовому просторі  $\mathcal{R}^m$ .

Відстань Евкліда між парою  $m$ -вимірних векторів  $\mathbf{x}_i$  і  $\mathbf{x}_j$  обчислюють за формулою

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \left( \sum_{k=1}^m (x_{ik} - x_{jk})^2 \right)^{\frac{1}{2}}, \quad (1.2)$$

де  $x_{ik}$  і  $x_{jk}$  –  $k$ -ті елементи векторів  $\mathbf{x}_i$  і  $\mathbf{x}_j$  відповідно. Ступінь подібності між вхідними сигналами, які зображують вектори  $\mathbf{x}_i$  і  $\mathbf{x}_j$ , є величиною, оберненою до відстані Евкліда між ними. Чим ближчі один до одного відповідні елементи векторів  $\mathbf{x}_i$  і  $\mathbf{x}_j$ , тим менша відстань Евкліда  $d(\mathbf{x}_i, \mathbf{x}_j)$  і тим більшою є подібність між цими векторами.

Другий метод визначення ступеня подібності ґрунтується на використанні поняття «скалярний добуток». Якщо вектори  $\mathbf{x}_i$  і  $\mathbf{x}_j$  мають однакову розмірність, то їхній скалярний добуток  $(\mathbf{x}_i, \mathbf{x}_j)$  визначають формулою  $(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j = \sum_{k=1}^m x_{ik} x_{jk}$ . Результат ділення скалярного добутку  $(\mathbf{x}_i, \mathbf{x}_j)$  на значення виразу  $\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|$  дорівнює косинусу внутрішнього кута між векторами  $\mathbf{x}_i$  і  $\mathbf{x}_j$ .

Наведені два методи визначення ступеня подібності пов'язані один з одним: чим менша відстань Евкліда між векторами  $\mathbf{x}_i$  і  $\mathbf{x}_j$ , тим більшим є їхній скалярний добуток. Щоб формалізувати цей зв'язок, пронормуємо вектори  $\mathbf{x}_i$  і  $\mathbf{x}_j$ . При цьому їхня довжина буде дорівнювати одиниці:  $\|\mathbf{x}_i\| = \|\mathbf{x}_j\| = 1$ . Використовуючи формулу (1.2), отримаємо:  $d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) = 2 - 2(\mathbf{x}_i, \mathbf{x}_j)$ , звідки можна зробити висновок, що мінімізація відстані Евкліда  $d(\mathbf{x}_i, \mathbf{x}_j)$  і максимізація скалярного добутку  $(\mathbf{x}_i, \mathbf{x}_j)$  зумовлюють збільшення ступеня подібності векторів  $\mathbf{x}_i$  і  $\mathbf{x}_j$ . Зазначимо, що відстань Евкліда і скалярний добуток описані в умовах повної визначеності.

Розглянемо випадок, коли вектори  $\mathbf{x}_i$  і  $\mathbf{x}_j$  належать до різних множин даних. Припустимо, що різниця між двома множинами даних полягає в різниці між математичними сподіваннями (середніми) їхніх елементів у вигляді векторів.

Нехай вектори  $\boldsymbol{\mu}_i$  і  $\boldsymbol{\mu}_j$  – середні значення векторів  $\mathbf{x}_i$  і  $\mathbf{x}_j$  відповідно, тобто  $\boldsymbol{\mu}_i = E[\mathbf{x}_i]$ ,  $\boldsymbol{\mu}_j = E[\mathbf{x}_j]$ , де  $E$  – статистичний оператор математичного сподівання. Для вимірювання відстані між двома множинами використовують відстань Махаланобіса  $d_{ij}$ , квадрат якої визначають за формулою

$$d_{ij}^2 = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \boldsymbol{\sigma}^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_j),$$

де  $\boldsymbol{\sigma}^{-1}$  – обернена матриця до матриці коваріації  $\boldsymbol{\sigma}$ ; припускається, що матриці коваріації заданих множин однакові:

$$\boldsymbol{\sigma} = E[(\mathbf{x}_i - \boldsymbol{\mu}_i)^T (\mathbf{x}_i - \boldsymbol{\mu}_i)] = E[(\mathbf{x}_j - \boldsymbol{\mu}_j)^T (\mathbf{x}_j - \boldsymbol{\mu}_j)].$$

Якщо  $\mathbf{x}_i = \mathbf{x}_j$ ,  $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j = \boldsymbol{\mu}$ ,  $\boldsymbol{\sigma} = \mathbf{I}$  (де  $\mathbf{I}$  – одинична матриця), то відстань Махаланобіса перетворюється на відстань Евкліда між вектором  $\mathbf{x}_i$  і вектором математичного сподівання  $\boldsymbol{\mu}$ .

**Правило 2.** Для подання в НМ важливої властивості необхідно використовувати велику кількість нейронів.

Як приклад розглянемо задачу виявлення радаром деякого об'єкта (наприклад, літака) за наявності завад, які викликані відбиттям сигналу від сторонніх об'єктів (будинків, дерев тощо). Ефективність такої радарної системи визначають дві ймовірнісні величини: 1) ймовірність того, що за наявності об'єкта система його виявить; 2) ймовірність того, що система визначить наявність об'єкта за його відсутності.

За критерієм Неймана–Пірсона ймовірність виявлення об'єкта має бути максимальною, а ймовірність помилкової тривоги не повинна перевищувати деякої заданої величини. Слід зауважити, що для розв'язання схожих задач дуже важливо, щоб вхідний сигнал містив інформацію про об'єкт. У другому правилі йдеться, що для прийняття рішення мережею про наявність (або відсутність) шуканого об'єкта потрібно залучити велику кількість нейронів. У будь-якому випадку збільшення кількості нейронів підвищує достовірність прийнятого рішення і стійкість до завад.

**Правило 3.** Структура НМ має містити апіорну інформацію та інваріанти – це спрощує архітектуру мережі та процес її навчання. Правильна конфігурація мережі забезпечує її спеціалізацію (спеціалізовану структуру), що дуже важливо, оскільки, по-перше, біологічні НМ, що забезпечують обробку зорової та слухової інформації, є сильно спеціалізованими; по-друге, НМ зі спеціалізованою структурою зазвичай включає значно меншу кількість вільних параметрів, які потрібно налаштовувати, ніж повнозв'язна мережа (мережа з повним набором можливих з'єднань); у зв'язку з цим для навчання спеціалізованої мережі потрібно менше даних і часу; по-третє, спеціалізовані мережі мають більшу пропускну здатність, а вартість їхнього створення менша, ніж вартість створення повнозв'язних мереж.

**Включення апіорної інформації в структуру НМ.** Для виконання третього правила необхідно розробити спеціалізовану структуру НМ, в якій би містилась апіорна інформація. У наш час немає чіткого вирішення цього завдання, однак можна запропонувати деякі процедури, які дають хороші результати. Зокрема, можна використовувати комбінацію двох прийомів, які значно скорочують кількість вільних параметрів мережі:

1. Обмеження архітектури мережі за допомогою локальних зв'язків, які називають рецепторними полями. У контексті систем обробки зорової інформації рецепторне поле нейрона – це обмежена ділянка сітківки, яка виконує функцію передавання цьому нейрону сигналів від світлочутливих елементів.

2. Обмеження вибору синаптичних вагових коефіцієнтів за рахунок спільного використання ваги.

Як приклад розглянемо неповнозв'язну мережу прямого поширення (рис. 1.13), яка має обмежену архітектуру. Перші шість входів утворюють рецепторне поле прихованого нейрона з номером 1 і т. д. для всіх інших прихованих нейронів мережі. Усі чотири приховані нейрони для реалізації синаптичних зв'язків спільно використовують одну множину вагових коефіцієнтів. Спільне використання ваги полягає в застосуванні однакового набору синаптичних вагових коефіцієнтів для кожного з нейронів прихованого шару мережі. Враховуючи, що кожний з чотирьох прихованих нейронів має по шість локальних зв'язків (рис. 1.13), індуковане локальне поле прихованого  $j$ -го нейрона можна описати за формулою

$$u_j = \sum_{i=1}^6 w_i p_{i+j-1}, \quad j = 1, 2, 3, 4, \quad (1.3)$$

де  $\{w_i\}_{i=1}^6$  – набір вагових коефіцієнтів, який спільно використовується всіма (чотирма) нейронами прихованого шару;  $p_k$  – сигнал, одержаний від джерела з номером  $k = i + j - 1$ . Вираз (1.3) записано у формі суми згортки, саме тому мережу прямого поширення з локальними зв'язками і ваговими коефіцієнтами, які використовуються спільно, називають мережею згортки.

**Включення інваріантів у структуру НМ.** Для того, щоб створити систему розпізнавання об'єктів, мови або ехолокації, необхідно врахувати діапазон можливих трансформацій вхідного сигналу, що надходить від об'єкта спостереження. У зв'язку з цим основною вимогою для розпізнавання образів є створення такого класифікатора, який був би інваріантним (незмінним) щодо вказаних трансформацій. Розглянемо найпоширеніші *прийоми забезпечення інваріантності щодо трансформацій вхідного сигналу на основі НМ.*

1. *Структурна інваріантність.* Інваріантність може бути забезпечена за допомогою структуризації. Зокрема, синаптичні зв'язки між окремими нейронами мережі будуються таким чином, щоб трансформовані версії одного сигналу викликали однаковий вихідний сигнал. Розглянемо, наприклад, класифікацію вхідного сигналу за допомогою НМ, яка має бути

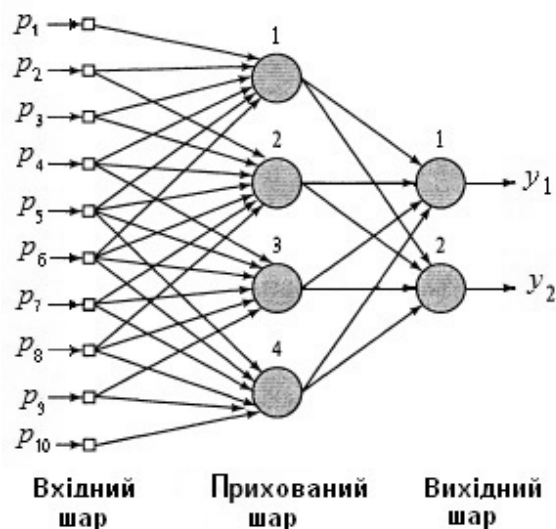


Рис. 1.13. Приклад мережі з рецепторними полями

інваріантною щодо плоского обертання зображення відносно свого центру.

Структурну інваріантність мережі щодо обертання можна виразити таким чином. Нехай  $w_{ji}$  – синаптична вага  $j$ -го нейрона, який пов'язаний з  $i$ -м пікселем вхідного зображення. Якщо умова  $w_{ji} = w_{jk}$  виконується для всіх пікселів  $i$  та  $k$ , що перебувають на однаковій відстані від центру зображення, то НМ буде інваріантна щодо обертання. Однак для того, щоб забезпечити таку інваріантність, потрібно продублювати синаптичні вагові коефіцієнти  $w_{ji}$  усіх пікселів, рівновіддалених від центру зображення. Недоліком структурної інваріантності є те, що кількість синаптичних зв'язків навіть для зображення середнього розміру є надзвичайно великою.

2. *Інваріантність у результаті навчання.* Здатність НМ класифікувати образи можна використовувати для забезпечення інваріантності мережі щодо трансформацій. Мережа навчається на множині прикладів одного й того самого об'єкта, при цьому в кожному прикладі об'єкт подається в дещо зміненому вигляді (наприклад, знімки з різних ракурсів). Якщо таких прикладів багато і НМ навчена розрізняти різні ракурси об'єкта, то можна очікувати, що ці дані будуть узагальнені й мережа зможе розпізнавати ракурси об'єкта, які не використовувалися під час навчання. Проте з технічного погляду інваріантність у результаті навчання має два недоліки: 1) якщо НМ була навчена розпізнавати трансформації об'єктів деякого класу, то вона не обов'язково буде інваріантною по відношенню до трансформацій об'єктів інших класів; 2) таке навчання потребує дуже багато ресурсів, особливо якщо розмірність простору ознак велика.

3. *Використання інваріантних ознак.* Третій метод створення інваріантного класифікатора на основі НМ ґрунтується на такому припущенні: із вхідного сигналу можна виділити інформативні ознаки, що описують найважливішу інформацію у вигляді множини даних, які є інваріантними щодо трансформацій вхідного сигналу. Для використання інваріантних ознак у НМ не потрібно зберігати зайвої інформації, що описує трансформації об'єкта, а відмінності між різними екземплярами одного об'єкта можуть спричинити тільки випадкові фактори (наприклад, шум). Використання простору інваріантних ознак має такі переваги: 1) зменшується кількість ознак, що надходять до НМ; 2) послаблюються вимоги до структури мережі; 3) гарантується інваріантність усіх об'єктів щодо відомих трансформацій.

Отже, використання інваріантних ознак є найкращим методом забезпечення інваріантності класифікаторів на основі НМ.

*Подання знань у мережі пов'язане з визначенням архітектури НМ.* На жаль, у наш час не існує ані формалізованої теорії оптимізації структури НМ, ані оцінки впливу архітектури мережі на подання знань у ній. Ці проблеми, зазвичай, вирішують експериментально. Незалежно від обраної архітектури мережі в процесі її навчання знання про предметну область зображують у вигляді вагових коефіцієнтів синаптичних зв'язків мережі. Така форма подання знань дозволяє НМ адаптуватися і виконувати узагальнення, але вона не забезпечує повноцінного описання процесу обчислення, який використовується для прийняття рішень або формування вихідного сигналу. Це може значно обмежити використання НМ, особливо коли вирішальним є принцип безпеки (наприклад, у сфері медичної діагностики), а також необхідно забезпечити можливість пояснення прийнятого рішення. Одним зі способів забезпечення такої можливості є інтеграція НМ і моделей штучного інтелекту в єдину гібридну систему.

### **Приклади розв'язання задач\***

**Приклад 1.1.** На єдиний вхід нейрона подається значення  $p = 2,0$ , його вага дорівнює  $w = 2,3$ , зсув  $b = -3$ . Встановити, яке значення одержимо на виході нейрона, якщо він має такі ФА:

- 1) одинарного стрибка:  $y = \text{hardlim}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 1, & \text{якщо } u \geq 0; \end{cases}$
- 2) лінійну:  $y = \text{purelin}(u) = u$ ;
- 3) сигмоїдну (логістичного типу):  $y = \text{logsig}(u) = \frac{1}{1 + e^{-u}}$ .

*Розв'язання.* На ФА подається значення

$$u = wp + b = 2,3 \cdot 2 + (-3) = 1,6.$$

На виході нейрона одержимо:

- 1) для функції одинарного стрибка:  $y = \text{hardlim}(1,6) = 1,0$ ;
- 2) для лінійної ФА:  $y = \text{purelin}(1,6) = 1,6$ ;
- 3) для сигмоїдної ФА:  $y = \text{logsig}(1,6) = \frac{1}{1 + e^{-1,6}} = 0,832$ .

**Приклад 1.2.** Маємо нейрон із двома входами та параметрами:  $b = 1,2$ ;  $\mathbf{w} = [3 \ 2]^T$ ;  $\mathbf{p} = [-5 \ 6]^T$ . Розрахувати значення на виході нейрона для таких ФА:

---

\*Задачі взято з посібника [14].

- 1) симетричної з різким порогом:  $y = \text{hardlims}(u) = \begin{cases} -1, & \text{якщо } u < 0; \\ 1, & \text{якщо } u \geq 0; \end{cases}$
- 2) кусково-лінійної з насиченням:  $y = \text{satlin}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ u, & \text{якщо } 0 \leq u \leq 1; \\ 1, & \text{якщо } u > 1; \end{cases}$
- 3) сигмоїдної функції гіперболічного тангенса:  $y = \text{tansig}(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}.$

*Розв'язання.* Розрахуємо значення входу  $u$  ФА:

$$u = \mathbf{w}^T \mathbf{p} + b = [3 \quad 2] \cdot \begin{bmatrix} -5 \\ 6 \end{bmatrix} + 1,2 = -1,8.$$

Знаходимо значення на виході для кожної ФА:  $y = \text{hardlims}(-1,8) = -1$ ;  $y = \text{satlin}(-1,8) = 0$ ;  $y = \text{tansig}(-1,8) = -0,9468$ .

**Приклад 1.3.** В одношаровій НМ маємо шість входів і два виходи. Значення виходів обмежені на інтервалі  $[0; 1]$ . Скільки нейронів має архітектура мережі, який розмір має вагова матриця, які можуть бути використані ФА?

*Розв'язання.* Умова завдання дозволяє сказати про НМ таке: вона має два нейрони (по одному на кожний вихід); вагова матриця  $\mathbf{W}$  має два рядки, які відповідають двом нейронам, і шість стовпців, які відпові-

дають шести входам:  $\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & w_{16} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & w_{26} \end{bmatrix}$ ; добуток

$\mathbf{Wp}$  є двохелементним вектором; на роль ФА найбільше підходить сигмоїдна функція *logsig*, а лінійну ФА не можна використати, оскільки значення виходів мають бути обмежені на інтервалі  $[0; 1]$ .

### **Контрольні запитання**

1. Нарисуйте спрощену структуру біологічної нервової клітини (нейрона). Як відбувається передавання сигналів усередині нервової системи (взаємодія з іншими нейронами)?

2. Опишіть модель штучного нейрона та її складові.

3. Опишіть першу модель штучної НМ (модель Маккаллока–Піттса) та її особливості.

4. Назвіть параметри, якими визначається архітектура багатошарової мережі.

5. Опишіть сутність конструктивного та деструктивного методів навчання НМ.



6. Як розрізняють (класифікують) НМ з урахуванням різних критеріїв?
7. Опишіть найпоширеніші прийоми забезпечення інваріантності щодо змін вхідного сигналу на основі НМ.

### **Задачі для самостійного розв'язання**

**Задача 1.1.** На єдиний вхід нейрона подається значення  $p = 2,0$ , його вага  $w = 1,3$ , а зсув  $b = 3$ . Які ФА можуть бути у цього нейрона, якщо на виході маємо значення 1; 0,9963?

**Задача 1.2.** Маємо нейрон з одним входом і зсувом. Потрібно, щоб на виході було значення «-1» для вхідних значень, які менші за три, та «1» для вхідних значень, які більші трьох або дорівнюють йому. Які необхідні ФА та значення зсуву, що задовольняють умову завдання? Чи пов'язане значення зсуву з вагою та входом? Якщо так, то як саме? Накресліть діаграму мережі.

**Задача 1.3.** Маємо нейрон із двома входами, ваговим вектором  $\mathbf{w} = [3 \ 2]^T$  і вхідним вектором  $\mathbf{p} = [-5 \ 7]^T$ . На виході потрібно одержати значення 0,5. Визначити ФА, які б дозволили одержати цей результат. Чи існує зсув, який задовольняє умову завдання, якщо використовується: а) сигмоїдна ФА (*logsig*); б) лінійна ФА (*purelin*); в) симетрична з різким порогом ФА (*hardlims*)? Якщо так, то яке його значення?

## Розділ 2. ПЕРСЕПТРОН

### 2.1. Архітектура нейрона типу персептрон

Відповідно до принципів функціонування біологічних нейронів існують різні математичні моделі штучних нейронів, які реалізують властивості біологічної нервової клітини. Узагальнена схема, покладена в основу більшості таких моделей, відповідає моделі нейрона Маккаллока–Піттса. Ця модель містить суматор зважених вхідних сигналів і нелінійний блок, який формує вихідний сигнал нейрона, що функціонально залежить від вихідного сигналу суматора. Властивості нелінійної ФА (особливо її неперервність) впливають на спосіб навчання нейрона (вибір вагових коефіцієнтів та зсуву).

Нейрон, який використовується в моделі персептрона (рис. 2.1), має ФА з різким порогом: повертає значення «1», якщо вхід ФА  $u \geq 0$ , і значення «0», якщо  $u < 0$ . Кожному елементу  $p_j$  вектора входу  $\mathbf{p}$  персептрона відповідає вага  $w_{1j}$ . Завдяки ФА персептрон може класифікувати вхідні вектори, розділяючи простір вхідних даних на дві області, як це зображено для персептрона з двома входами та зсувом на рис. 2.2. При цьому простір вхідних даних розділяється на дві області за допомогою прямої  $L$ , яка задається рівнянням  $\mathbf{w}^T \mathbf{p} + b = 0$ . Ця пряма перпендикулярна до вагового вектора  $\mathbf{w}$  і зміщена на величину  $b$ . Вектори входу, розміщені вище прямої  $L$ , відповідають додатному потенціалу нейрона, тому вихід персептрона для цих векторів буде дорівнювати одиниці. Вектори входу, розміщені нижче прямої  $L$ , відповідають виходу персептрона, який дорівнює нулю. Зі зміною значень зсуву та ваги пряма  $L$  змінює своє положення. Коли розмірність вектора входу перевищує «2», межею поділу буде гіперплощина.

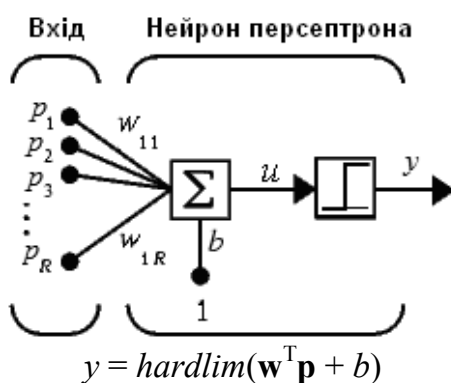


Рис. 2.1. Схема персептрона

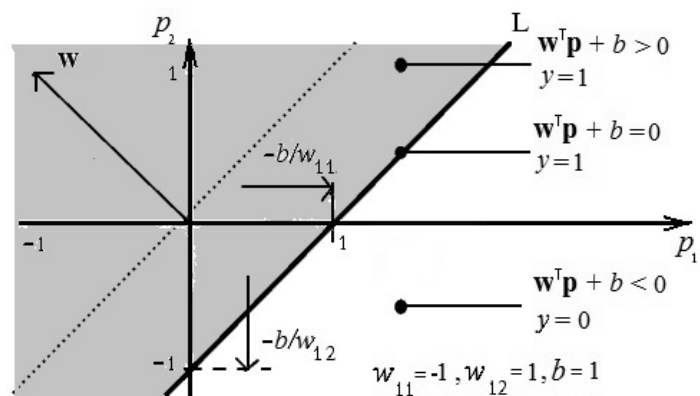


Рис. 2.2. Робота персептрона з двома входами і зсувом

Модель НМ на основі одношарового персептрона, рівняння якої в системі MatLab має вигляд  $y = \text{hardlim}(\mathbf{Wp} + \mathbf{b})$ , зображено на рис. 2.3.

Розглянемо матрицю вагових коефіцієнтів мережі:  $\mathbf{W} = \begin{bmatrix} w_{11} & \dots & w_{1R} \\ \dots & \dots & \dots \\ w_{S1} & \dots & w_{SR} \end{bmatrix}$ .

Зобразимо вагову матрицю у такому вигляді:  $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \dots \\ \mathbf{w}_S^T \end{bmatrix}$ , де  $\mathbf{w}_i = \begin{bmatrix} w_{i1} \\ \dots \\ w_{iR} \end{bmatrix}$ ,  $i = 1, 2, \dots, S$ . Це дозволяє записати  $i$ -й елемент вектора виходу мережі у вигляді  $y_i = \text{hardlim}(u_i) = \text{hardlim}(\mathbf{w}_i^T \mathbf{p} + b_i)$ , де  $\text{hardlim}(u) = \begin{cases} 1, & \text{якщо } u \geq 0; \\ 0, & \text{якщо } u < 0. \end{cases}$

Отже, ключова властивість персептрона, який складається з одного нейрона, полягає в тому, що він може розбити вхідні вектори на дві групи, граничний розв'язок між якими визначається рівнянням  $\mathbf{w}^T \mathbf{p} + b = 0$ . Оскільки границя поділу є лінійною, то одношаровий персептрон можна використовувати тільки для розпізнавання вхідних векторів, які є лінійно подільними.

Отже, ключова властивість персептрона, який складається з одного нейрона, полягає в тому, що він може розбити вхідні вектори на дві групи, граничний розв'язок між якими визначається рівнянням  $\mathbf{w}^T \mathbf{p} + b = 0$ . Оскільки границя поділу є лінійною, то одношаровий персептрон можна використовувати тільки для розпізнавання вхідних векторів, які є лінійно подільними.

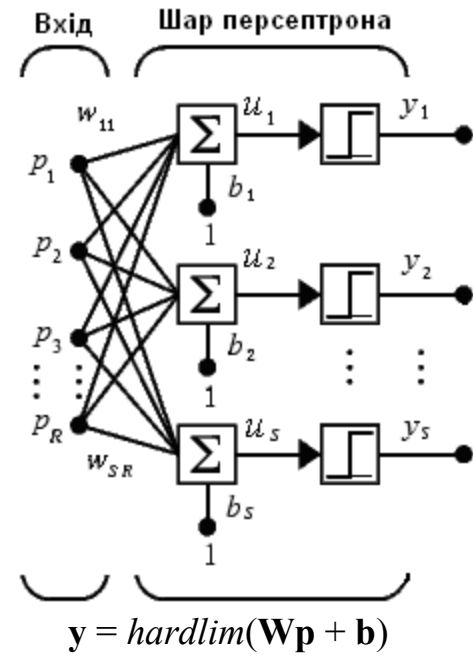


Рис. 2.3. Модель мережі на основі одношарового персептрона

## 2.2. Принципи функціонування однеюнейронного персептрона

Розглянемо однеюнейронний персептрон із двома входами [12, 14]. Вихід цієї мережі в системі MatLab визначається таким чином:

$$y = \text{hardlim}(u) = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b) = \text{hardlim}(w_{11}p_1 + w_{12}p_2 + b).$$

Граничний розв'язок визначається вхідними векторами, для яких виконується умова  $u = \mathbf{w}^T \mathbf{p} + b = w_{11}p_1 + w_{12}p_2 + b = 0$ .

**Приклад 2.1.** Нехай  $w_{11} = 1$ ,  $w_{12} = 1$ ,  $b = -1$ . Тоді граничний розв'язок набуває вигляду  $u = w_{11}p_1 + w_{12}p_2 + b = p_1 + p_2 - 1 = 0$  і визначає пряму в просторі входів (рис. 2.4), з одного боку від якої вихід мережі дорівнює одиниці, а з другого – нулю. Для визначення цієї прямої знайдемо точки перетину з осями  $p_1$  і  $p_2$ :  $p_2 = -\frac{b}{w_{12}} = -\frac{-1}{1} = 1$ , якщо

$p_1 = 0$ ;  $p_1 = -\frac{b}{w_{11}} = -\frac{-1}{1} = 1$ , якщо  $p_2 = 0$ . Щоб визначити, який бік границі відповідає виходу  $y = 1$ , потрібно перевірити хоча б одну точку. Наприклад, для входу  $\mathbf{p} = [2 \ 0]^T$  вихід мережі  $y = \text{hardlim}({}_1\mathbf{w}^T\mathbf{p} + b) = \text{hardlim}\left([1 \ 1] \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} - 1\right) = 1$ . Отже, виходу мережі  $y = 1$  відповідає область, яка міститься справа від граничного розв'язку. Граничний розв'язок можна також знайти графічно, звернувши увагу на те, що границя, яка визначається рівнянням

$${}_1\mathbf{w}^T\mathbf{p} + b = 0, \quad (2.1)$$

завжди ортогональна до вектора  ${}_1\mathbf{w}$  (рис. 2.4). Застосуємо персептрон для реалізації простої логічної функції «І», для якої пари вхід–ціль мають такий вигляд:  $\left\{\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = 0\right\}$ ,  $\left\{\mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 0\right\}$ ,  $\left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 0\right\}$ ,  $\left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 1\right\}$ .

Проблему реалізації функції «І» графічно ілюструє рис. 2.5, а, де входи інтерпретуються таким чином: темні кола  $\bullet$  вказують на те, що ціль дорівнює одиниці; світлі кола  $\circ$  вказують на те, що ціль дорівнює нулю. Перший крок функціонування нейрона полягає у визначенні граничного розв'язку (тобто прямої, яка відокремлює темні кола від світлих). Таких прямих нескінченна кількість. Оберемо серед них пряму, яка проходить посередині між двома видами входів (рис. 2.5, б). Далі виберемо ваговий вектор, який є ортогональним до граничного розв'язку. Оскільки вагові вектори можуть мати будь-яку довжину, то їх існує нескінченна кількість, наприклад (рис. 2.5, б):  ${}_1\mathbf{w} = [2 \ 2]^T$ . Тепер необхідно визначити значення зсуву  $b$ , вибравши точку граничного розв'язку, тобто точку, яка задовольняє рівняння (2.1).

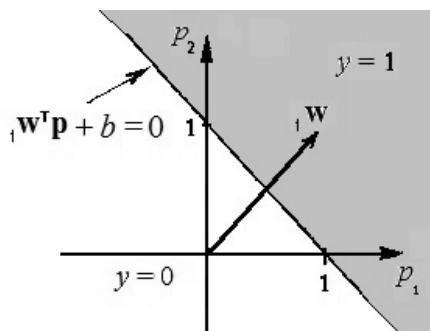


Рис. 2.4. Розв'язок, який відповідає прикладу 2.1

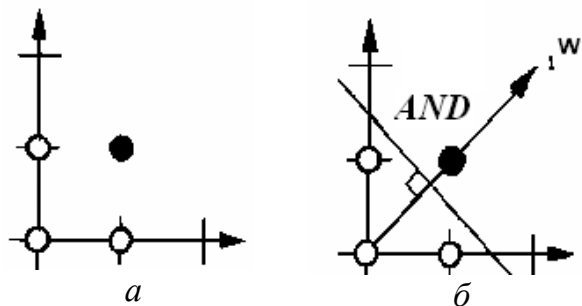


Рис. 2.5. Вхідні дані для реалізації логічної функції «І»

Для вхідного вектора  $\mathbf{p} = [1,5 \ 0]^T$  одержимо:  ${}_1\mathbf{w}^T \mathbf{p} + b = [2 \ 2] \times$

$\times \begin{bmatrix} 1,5 \\ 0 \end{bmatrix} + b = 3 + b = 0$ , звідки  $b = -3$ . Перевіримо тепер мережу на одній

із пар вхід–ціль. Для вхідного вектора  $\mathbf{p}_2$  одержимо вихід вигляду

$$y = \text{hardlim} ({}_1\mathbf{w}^T \mathbf{p}_2 + b) = \text{hardlim} \left( [2 \ 2] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 3 \right) = \text{hardlim} (-1) = 0,$$

який дорівнює  $t_2$ . Можна перевірити, що всі входи класифікуються правильно.

### 2.3. Процедура навчання одношарового персептрона

Для персептрона, який складається із  $S$  нейронів (рис. 2.3), існує один граничний розв'язок для кожного нейрона, при цьому граничний розв'язок для  $i$ -го нейрона має вигляд  ${}_i\mathbf{w}^T \mathbf{p} + b_i = 0$ ,  $i = 1, \dots, S$ . Один нейрон персептрона може розбити вхідні вектори на два класи, оскільки його вихід набув одного із двох значень: «0» або «1». Кожний клас можна зобразити різними векторами виходу. Отже, загальна кількість можливих класів дорівнює  $2^S$ .

Визначимо алгоритм (правило) навчання персептрона, який складається з  $S$  нейронів [14]. Це правило визначається набором прикладів поведінки мережі:  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , де  $\mathbf{p}_i$  – вхід мережі;  $\mathbf{t}_i$  – бажаний вихід ( $i = 1, \dots, Q$ ). Правило змінює значення ваги та зсуву мережі таким чином, щоб наблизити виходи мережі до цілі. Розглянемо приклад алгоритму навчання такого персептрона.

Нехай необхідно визначити мережу типу персептрон, яка здатна класифікувати вхідні вектори для заданих пар вхід–ціль, вигляду (рис. 2.6, а):

1) пара  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\}$  зображена темним колом ●;

2) пари  $\left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\}; \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$  зображені світлими

колами ○.

Для вирішення цього завдання мережа повинна мати два входи та один вихід. Щоб спростити правило навчання нейрона, побудуємо мережу без зсуву, яка матиме два параметри:  $w_{11}$  та  $w_{12}$ . Нагадаємо: якщо зсуву немає, граничний розв'язок проходить через початок координат. Переконаємося, що визначена мережа може вирішити поставлене

завдання. Для цього має існувати граничний розв'язок, який може відокремити вектори  $\mathbf{p}_2$  і  $\mathbf{p}_3$  від вектора  $\mathbf{p}_1$ . Нескінченну кількість таких граничних розв'язків і вагових векторів, які їм відповідають, зображено на рис. 2.6, б. Зазначимо, що ваговий вектор є ортогональним до граничного розв'язку і має значення тільки його напрямку.

Навчання розпочинається з присвоєння деяких початкових значень параметрам мережі. Необхідно навчити мережу з двома входами та одним виходом без зсуву, тому треба ініціалізувати тільки два вагові коефіцієнти мережі. Нехай початковий ваговий вектор  ${}_1\mathbf{w} = [1 \ -0,8]^T$ , тоді для входу  $\mathbf{p}_1$

$$y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_1) = \text{hardlim}\left([1 \ -0,8] \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \text{hardlim}(-0,6) = 0$$

(тобто вихід мережі  $y = 0$ , тоді як має дорівнювати одиниці:  $t_1 = 1$ ).

Результат роботи розглянутої НМ зображено на рис. 2.6, в. Початковому ваговому вектору  ${}_1\mathbf{w}$  відповідає граничний розв'язок, який неправильно класифікує вектор  $\mathbf{p}_1$ , тому слід змінити ваговий вектор таким чином, щоб мати правильну класифікацію. Додавання  $\mathbf{p}_1$  до  ${}_1\mathbf{w}$  наближає  ${}_1\mathbf{w}$  до  $\mathbf{p}_1$ . Повторне додавання  $\mathbf{p}_1$  асимптотично наближає  ${}_1\mathbf{w}$  до  $\mathbf{p}_1$ .

Це правило має такий вигляд: якщо  $t = 1$  і  $y = 0$ , то

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1. \quad (2.2)$$

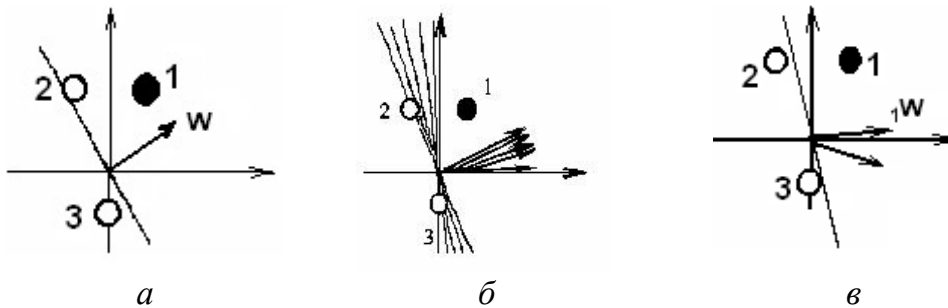


Рис. 2.6. Приклад функціонування одношарового персептрона: а – вхідні вектори; б – нескінченна кількість можливих граничних розв'язків; в – результат роботи мережі

Застосовуючи це правило до розглянутого вище завдання, одержимо нове значення  ${}_1\mathbf{w}$ :

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1,0 \\ -0,8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2,0 \\ 1,2 \end{bmatrix}.$$

Для вхідного вектора  $\mathbf{p}_2$  матимемо:

$$y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_2) = \text{hardlim}\left([2 \ 1,2] \cdot \begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) = \text{hardlim}(0,4) = 1.$$

Ціль  $t_2$ , пов'язана із входом  $\mathbf{p}_2$ , дорівнює нулю, тоді як на виході мережі маємо одиницю. Отже, вектор  $\mathbf{p}_2$  класифіковано неправильно.

Тепер необхідно перемістити ваговий вектор  ${}_1\mathbf{w}$  подалі від вхідного вектора. Для цього можна замінити правило (2.2) на правило такого вигляду: якщо  $t = 0$  і  $y = 1$ , то

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}, \quad (2.3)$$

$$\text{одержимо } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2,0 \\ 1,2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3,0 \\ -0,8 \end{bmatrix}.$$

Для вхідного вектора  $\mathbf{p}_3$  матимемо:

$$y = \text{hardlim} ({}_1\mathbf{w}^T \mathbf{p}_3) = \text{hardlim} \left( \begin{bmatrix} 3,0 & 0,8 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right) = \text{hardlim} (0,8) = 1.$$

Відповідний результат роботи мережі  $y = 1$  є неправильною класифікацією вектора  $\mathbf{p}_3$ , оскільки  $t_3 = 0$ . Цей випадок вже було розглянуто, і йому відповідає правило навчання (2.3), згідно з яким

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3,0 \\ -0,8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3,0 \\ 0,2 \end{bmatrix}.$$

Результат роботи НМ показує, що персептрон нарешті навчився правильно класифікувати вектори  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  і  $\mathbf{p}_3$ . Отже, для трьох заданих векторів остаточне правило навчання має вигляд: якщо  $t = y$ , то  ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$ , і всі можливі комбінації вихідних і цільових змінних охоплюють такі три правила навчання:

$$\left. \begin{aligned} {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old} + \mathbf{p}, \quad t = 1, y = 0; \\ {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old} - \mathbf{p}, \quad t = 0, y = 1; \\ {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old}, \quad t = y. \end{aligned} \right\} \quad (2.4)$$

Три правила навчання із (2.4) можна переписати у вигляді окремого виразу. Для цього визначимо похибку персептрона як  $e = t - y$ . Тоді правила (2.4) можна переписати таким чином:

$$\left. \begin{aligned} {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old} + \mathbf{p}, \quad e = 1; \\ {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old} - \mathbf{p}, \quad e = -1; \\ {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old}, \quad e = 0. \end{aligned} \right\} \quad (2.5)$$

У перших двох правилах навчання (2.5) знак біля  $\mathbf{p}$  збігається зі знаком похибки  $e$ . Крім того, відсутність  $\mathbf{p}$  у третьому правилі навчання відповідає випадку  $e = 0$ . Отже, можна об'єднати три правила навчання

у такий вираз:

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t - y)\mathbf{p}. \quad (2.6)$$

Це правило можна розширити, врахувавши зсув. Нагадаємо, що зсув є ваговим коефіцієнтом, вхід якого завжди дорівнює одиниці. Таким чином, можна замінити вхід  $\mathbf{p}$  у правилі (2.6) на вхід із зсувом, який дорівнює одиниці. У результаті одержимо правило навчання персептрона для зсуву:

$$b^{new} = b^{old} + e. \quad (2.7)$$

Правило навчання персептрона (2.5) уточнює значення вагового вектора окремого нейрона типу персептрон, однак це правило можна узагальнити для персептрона із  $S$  нейронами. Для уточнення  $i$ -го рядка вагової матриці використаємо формулу  ${}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p}$ , а для уточнення  $i$ -го елемента вектора зсуву – формулу  $b_i^{new} = b_i^{old} + e_i$ ,  $i = 1, \dots, S$ . Отже, правило навчання персептрона можна записати у матричній формі:  $\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T$ .

Розглянемо процедуру навчання персептрона для задачі розпізнавання образів (класифікації векторів). Нехай задано дві пари вхід–ціль

$$\text{вигляду } \left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = 0 \right\}; \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = 1 \right\}. \text{ Найчастіше початкові}$$

значення ваги та зсуву задаються випадковим чином визначеними маленькими значеннями. Припустимо, що ці значення мають вигляд  ${}_1\mathbf{w} = [0,5 \ -1 \ -0,5]^T$ ,  $b = 0,5$ .

На першому кроці подамо на вхід мережі вектор  $\mathbf{p}_1$ . Перша ітерація правила навчання персептрона використовує вхідний вектор  $\mathbf{p}_1$ :

$$y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_1 + b) = \text{hardlim} \left( [0,5 \ -1 \ -0,5] \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0,5 \right) = 1.$$

Обчислимо похибку персептрона  $e = t_1 - y = 0 - 1 = -1$  й уточнимо значення ваги та зсуву:

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p}_1 = \begin{bmatrix} 0,5 \\ -1 \\ -0,5 \end{bmatrix} + (-1) \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0,5 \\ 0 \\ 0,5 \end{bmatrix};$$

$$b^{new} = b^{old} + e = 0,5 + (-1) = -0,5.$$



Друга ітерація правила навчання персептрона використовує вхідний вектор  $\mathbf{p}_2$ :

$$y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_2 + b) = \text{hardlim} \left( \begin{bmatrix} -0,5 & 0 & 0,5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (-0,5) \right) =$$

$$= \text{hardlim}(-0,5) = 0;$$

$$e = t_2 - y = 1 - 0 = 1;$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p}_2 = \begin{bmatrix} -0,5 \\ 0 \\ 0,5 \end{bmatrix} + (1) \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 1 \\ -0,5 \end{bmatrix};$$

$$b^{new} = b^{old} + e = -0,5 + 1 = 0,5.$$

Третя ітерація знову використовує вхідний вектор  $\mathbf{p}_1$ :

$$y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_1 + b) = \text{hardlim} \left( \begin{bmatrix} 0,5 & 1 & -0,5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0,5 \right) =$$

$$= \text{hardlim}(0,5) = 1;$$

$$e = t_1 - y = 0 - 1 = -1;$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e \cdot \mathbf{p}_1 = \begin{bmatrix} 0,5 \\ 1 \\ -0,5 \end{bmatrix} + (-1) \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0,5 \\ 2 \\ 0,5 \end{bmatrix};$$

$$b^{new} = b^{old} + e = 0,5 + (-1) = -0,5.$$

Якщо продовжити описаний процес навчання персептрона, то можна побачити, що обидва вхідні вектори класифікуються правильно.

Правило навчання персептрона гарантує збіжність до розв'язку за скінченну кількість кроків за умови, що такий розв'язок існує.

Персептрон може вирішувати різні завдання. Нагадаємо, що персептрон з одним нейроном може розділити простір лінійно подільних вхідних векторів на дві області, границя між якими є лінійною і визначається рівнянням  ${}_1\mathbf{w}^T \mathbf{p} + b = 0$ . Але існує багато задач, для яких простір вхідних даних не є лінійно подільним (рис. 2.7), у такому разі застосовують багатошарові мережі на основі персептрона.

Якщо довжина одного вектора входу значно відрізняється від довжини інших векторів, то для навчання персептрона може знадобитися багато часу. Це зумовлено тим, що алгоритм навчання пов'язаний з додаванням

або відніманням вхідного вектора від поточного вагового вектора. Можна зрівноважити вплив великих або малих елементів вхідного вектора через нормування вхідних даних:

$$\Delta_1 \mathbf{w} = (t - y) \frac{\mathbf{p}}{\|\mathbf{p}\|} = e \frac{\mathbf{p}}{\|\mathbf{p}\|}, \quad (2.8)$$

де  $\Delta_1 \mathbf{w} = \mathbf{w}^{new} - \mathbf{w}^{old}$ .

Нормоване правило навчання персептрона (2.8) значно скорочує кількість циклів навчання у випадку, коли значення векторів входу дуже різняться один від одного.

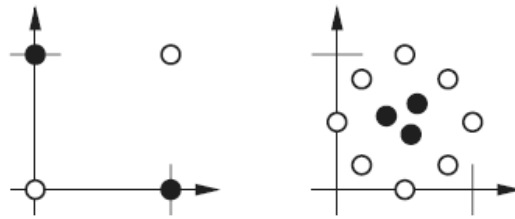


Рис. 2.7. Приклади задач з лінійно неподільним набором вхідних даних

Отже, процес навчання персептрона – це процедура налагодження значень ваги та зсуву через використання правила навчання з метою зменшення різниці між цільовими та вихідними значеннями.

Нейронні мережі на основі персептрона мають такі обмеження:

1) вихід персептрона може набувати тільки одного з двох значень (якщо використано ФА *hardlim* – «1» або «0», функцію *hardlims* – «1» або «-1»);

2) персептрони можуть вирішувати завдання класифікації тільки для лінійно подільних наборів векторів.

Опишемо алгоритм навчання персептрона для завдання класифікації з двома класами вхідних векторів, які відповідають значенням  $t = 0$  і  $t = 1$ .

Нехай задано навчальну вибірку у вигляді пар даних вхід–ціль:  $\{(\mathbf{p}^1, t^1), \dots, (\mathbf{p}^Q, t^Q)\}$ , яка генерується функцією  $t^i = f(\mathbf{p}^i)$ ,  $i = 1, \dots, Q$ , де  $\mathbf{p}^i = [p^i_1 \dots p^i_R]^T$  – вектор вхідних даних з елементами  $p^i_j \in \mathbb{R}$ ;  $t^i$  – бажаний відгук. Функцію  $f(\cdot)$  вважають невідомою, але задано множину її реалізацій:  $T = \{(p^i_1, \dots, p^i_R, t^i), i = 1, 2, \dots, Q, R \geq 1\}$ . Визначити функцію  $F(\mathbf{w}, \mathbf{p}^i)$ , яка найкраще апроксимує бажану функцію  $f(\mathbf{p})$ , якщо вона існує. Функція  $F(\mathbf{w}, \mathbf{p}^i)$  реалізована у вигляді НМ, яка описує перетворення вхідного сигналу у вихідний і задовольняє умову  $\sum_{i=1}^Q |F(\mathbf{w}, \mathbf{p}^i) - t^i| = 0$ .

*Змінні та параметри:* вектор входу  $\mathbf{p}(i) = [1 \ p_1(i) \ \dots \ p_R(i)]^T$ , ваговий вектор  $\mathbf{w}(i) = [b(i) \ w_1(i) \ \dots \ w_R(i)]^T$ , зсув  $b(i)$ , вихід нейрона  $y(i)$ , бажаний вихід нейрона  $t(i)$ , параметр швидкості навчання  $0 < \alpha \leq 1$ .

*Алгоритм навчання одного нейрона* типу персептрон (приклад 1 дод. Б):

*Крок 1. Ініціалізація:*  $\mathbf{w}(0) = \mathbf{0}$ .

*Крок 2. Активація.* На кроці  $i$  активуємо нейрон, використовуючи вектор входу  $\mathbf{p}(i)$  і бажаний відгук  $t(i)$ .

*Крок 3. Обчислення фактичного виходу нейрона для входу  $\mathbf{p}(i)$ :*

$$y(i) = \text{hardlim}(\mathbf{w}^T(i-1), \mathbf{p}(i)).$$

*Крок 4. Адаптація вагового вектора:*  $\mathbf{w}(i) = \mathbf{w}(i-1) + \alpha e(i) \mathbf{p}(i)$ , де  $e(i) = t(i) - y(i)$ .

*Крок 5.* Якщо  $\sum_{i=1}^Q |e(i)| = 0$ , то навчання персептрона завершуємо; інакше, враховуючи, що  $i = i + 1$ , переходимо на крок 2.

## 2.4. Одношаровий персептрон Ф. Розенблатта

Ключовим внеском Ф. Розенблатта в терію НМ стало запропоноване ним правило навчання персептрона для вирішення проблеми розпізнавання образів (класифікації векторів) [12]. Він довів, що:

1) правило навчання персептрона гарантує збіжність до розв'язку за скінченну кількість кроків за умови, що такий розв'язок існує;

2) обчислювальні можливості персептрона обмежені: щоб він функціонував коректно, необхідно, щоб простір вхідних даних був лінійно подільним. Відповідно до Ф. Розенблатта вхідні значення персептрона є біполярними (дорівнюють нулю або одиниці), а вага і зсув набувають дійсних значень. Рівень активації  $i$ -го нейрона типу персептрон обчислюють як зважену суму його входів:

$u_i = \sum_{j=1}^{R+1} w_{ij} p_j$ , де  $p_{R+1} = 1$ ,  $w_{i(R+1)} = b$ , а його вихідне значення обчислюють за формулою

$$y_i = f(u_i) = \text{hardlims}(u_i) = \begin{cases} 1, & \text{якщо } \sum_{j=1}^R w_{ij} p_j + b \geq 0; \\ -1, & \text{якщо } \sum_{j=1}^R w_{ij} p_j + b < 0. \end{cases}$$

Щоб налагодити вагу персептрона використовують простий алгоритм навчання з учителем на основі корекції похибок. Нехай  $\alpha$  – сталий коефіцієнт швидкості навчання:  $0 < \alpha \leq 1$ , а  $t_i$  – цільове значення виходу.

Тоді налагодження вагового коефіцієнта для  $i$ -го компонента вхідного вектора виконується за формулою  $\Delta w_{ij} = \alpha(t_i - y_i)$ . Різниця між цільовим та вихідним значеннями може дорівнювати «0», «2» або «-2». Отже, для кожного елемента вхідного вектора виконуються такі дії:

1) якщо цільове та вихідне значення збігаються, нічого не відбувається;

2) якщо вихідне значення дорівнює «-1», а цільове «1», то ваговий коефіцієнт для  $i$ -го входу збільшується на  $2\alpha p_i$ ;

3) якщо вихідне значення дорівнює «1», а цільове «-1», то ваговий коефіцієнт для  $i$ -го входу зменшується на  $2\alpha p_i$ . Виконавши таку процедуру навчання персептрона, одержимо вагу, яка мінімізує середню помилку на множині навчання.

### Приклади розв'язання задач\*

**Приклад 2.2.** Нехай задано множину точок (рис. 2.8). Розв'язати задачу класифікації, визначивши границю поділу цих точок (граничний розв'язок). Для кожної границі визначити значення ваги та зсуву, які відповідають персептрону з одним нейроном.

*Розв'язання. Перший крок.* Проведемо прямі між множинами чорних і білих точок даних (рис. 2.9).

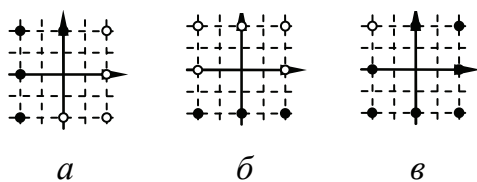


Рис. 2.8. Задача класифікації 2.2

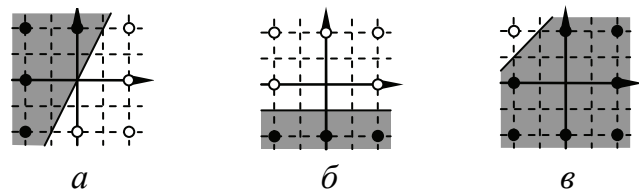


Рис. 2.9. Прямі між множинами чорних і білих заданих точок

*Другий крок.* Визначимо значення ваги (рис. 2.10) так, щоб вагові вектори були ортогональними до проведених границь і спрямованими у бік чорних точок ( $y = 1$ ), при цьому їхня довжина може бути довільною. Розглянемо, наприклад, такі значення вагових векторів для кожного випадку: **I.**  $\mathbf{w} = [-2 \ 1]^T$ . **II.**  $\mathbf{w} = [0 \ -2]^T$ . **III.**  $\mathbf{w} = [2 \ -2]^T$ .

Визначимо значення зсуву для кожного персептрона, вибравши довільну точку на границі й підставивши значення її координат у рівняння  $\mathbf{w}^T \mathbf{p} + b = 0$ . Одержимо  $b = -\mathbf{w}^T \mathbf{p}$ , тобто

\* Задачі взято з посібника [14].

$$\text{I. } b = -[-2 \ 1] \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0.$$

$$\text{II. } b = -[0 \ -2] \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -2.$$

$$\text{III. } b = -[2 \ -2] \cdot \begin{bmatrix} -2 \\ 1 \end{bmatrix} = 6.$$

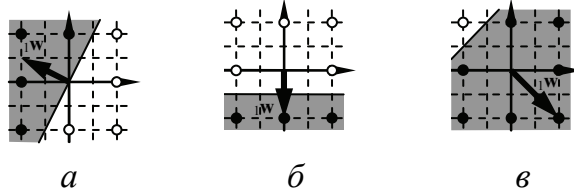


Рис. 2.10. Визначення значення ваги

Зазначимо, що замість першого та другого кроків можна використати формулу визначення рівняння прямої між двома точками:

$$\frac{p_1 - p_1^1}{p_1^2 - p_1^1} = \frac{p_2 - p_2^1}{p_2^2 - p_2^1}.$$

Перевіримо розв'язок та роботу мережі для п. I (рис. 2.10, а), розглянувши вектор входу  $\mathbf{p} = [-2 \ 2]^T$ :

$$y = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b) = \text{hardlim}\left([-2 \ 1] \cdot \begin{bmatrix} -2 \\ 2 \end{bmatrix} + 0\right) = \text{hardlim}(6) = 1,$$

звідки  $y = t = 1$ . Застосуємо цю НМ для класифікації вхідного вектора, якого немає в умові задачі, наприклад,  $\mathbf{p} = [1 \ 1]^T$ . Одержимо:

$$y = \text{hardlim}\left([-2 \ 1] \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0\right) = 0.$$

**Приклад 2.3.** Перетворити задачу класифікації наведених нижче точок в еквівалентну їй задачу, що складається з нерівностей, які обмежують значення ваги та зсуву:  $\{\mathbf{p}_1 = [0 \ 2]^T, t_1 = 1\}$ ;  $\{\mathbf{p}_2 = [1 \ 0]^T, t_2 = 1\}$ ;  $\{\mathbf{p}_3 = [0 \ -2]^T, t_3 = 0\}$ ;  $\{\mathbf{p}_4 = [2 \ 0]^T, t_4 = 0\}$ .

*Розв'язання.* Кожне значення  $t_i$  показує, яким має бути значення на виході ФА для вектора входу  $\mathbf{p}_i$ :  $y_i < 0$  або  $y_i \geq 0$ . Наприклад, оскільки  $t_1 = 1$ , то значення  $y_1$  на виході ФА, яке відповідає  $\mathbf{p}_1$ , має бути  $y_1 \geq 0$ . Отже, маємо таку нерівність:  $\mathbf{w}^T \mathbf{p}_1 + b \geq 0$ , звідки  $0w_{1 \ 1} + 2w_{1 \ 2} + b \geq 0$  або  $2w_{1 \ 2} + b \geq 0$ . Застосували цей метод до пар вхід-ціль  $\{\mathbf{p}_1, t_1\}$ ;

$\{\mathbf{p}_2, t_2\}$ ;  $\{\mathbf{p}_3, t_3\}$  і  $\{\mathbf{p}_4, t_4\}$ , отримаємо таку систему нерівностей:

$$\begin{cases} 0w_{11} + 2w_{12} + b \geq 0; \\ w_{11} + 0w_{12} + b \geq 0; \end{cases} \Rightarrow \begin{cases} 2w_{12} + b \geq 0; \\ w_{11} + b \geq 0; \end{cases} \quad (2.9)$$

$$\begin{cases} 0w_{11} - 2w_{12} + b < 0; \\ 2w_{11} + 0w_{12} + b < 0. \end{cases} \Rightarrow \begin{cases} -2w_{12} + b < 0; \\ 2w_{11} + b < 0. \end{cases} \quad (2.10)$$

$$\begin{cases} 0w_{11} - 2w_{12} + b < 0; \\ 2w_{11} + 0w_{12} + b < 0. \end{cases} \Rightarrow \begin{cases} -2w_{12} + b < 0; \\ 2w_{11} + b < 0. \end{cases} \quad (2.11)$$

$$\begin{cases} 0w_{11} - 2w_{12} + b < 0; \\ 2w_{11} + 0w_{12} + b < 0. \end{cases} \Rightarrow \begin{cases} -2w_{12} + b < 0; \\ 2w_{11} + b < 0. \end{cases} \quad (2.12)$$

Розв'язок цієї системи нерівностей можна знайти графічно. Для цього необхідно розглянути графічно дві пари нерівностей (2.9), (2.11) і (2.10), (2.12). Будь-які значення ваги та зсуву, які потрапляють в сірі області на графіках, є розв'язками задачі класифікації. Один із можливих розв'язків показано на рис. 2.11 символом «+», він відповідає значенням  $\mathbf{w} = [-2 \ 3]^T$ ,  $b = 3$ .

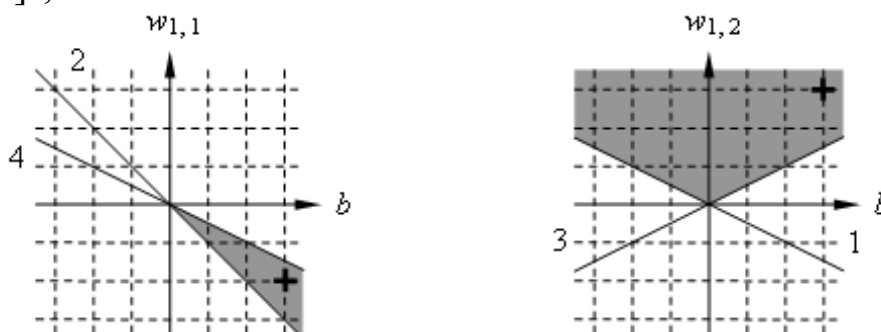


Рис. 2.11. Розв'язок системи нерівностей до прикл. 2.3

**Приклад 2.4.** Задано задачу класифікації з чотирма класами вхідних векторів, а саме:

клас 1:  $\{\mathbf{p}_1=[1 \ 1]^T, \mathbf{p}_2=[1 \ 2]^T\}$ ;

клас 2:  $\{\mathbf{p}_3=[2 \ -1]^T, \mathbf{p}_4=[2 \ 0]^T\}$ ;

клас 3:  $\{\mathbf{p}_5=[-1 \ 2]^T, \mathbf{p}_6=[-2 \ 1]^T\}$ ,

клас 4:  $\{\mathbf{p}_7=[-1 \ -1]^T, \mathbf{p}_8=[-2 \ -2]^T\}$ .

Визначити мережу на основі персептрона (вибір вагових векторів, ортогональних до границь поділу виконати графічним способом). Перевірити розв'язок, використовуючи всі вхідні вектори.

*Розв'язання.* Оскільки  $S$ -нейронний персептрон може розділити вхідні вектори на  $2^S$  класів, то для розв'язання задачі з чотирма класами вхідних векторів потрібен персептрон із двома нейронами (рис. 2.12). Зобразимо на графіку вхідні вектори (рис. 2.13): білі кола позначають клас 1, білі квадрати – клас 2, чорні кола – клас 3, а чорні квадрати – клас 4. Двонеуронний персептрон має дві лінійні границі поділу (рис. 2.14). Отже, задані вхідні

вектори можна лінійно поділити (при цьому вагові вектори мають бути ортогональними до границь поділу й спрямованими у бік областей, для яких виходи нейронів дорівнюють одиниці). Один із варіантів розв'язку зображено на рис. 2.15, де сірим кольором позначено області, що відповідають виходу 1.

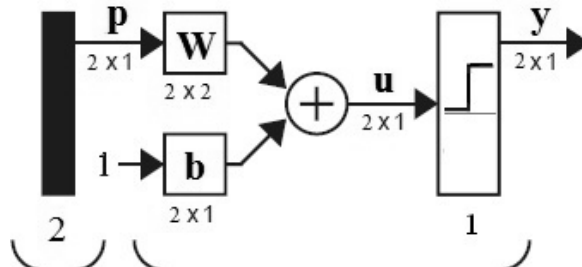


Рис. 2.12. Двонейронний персептрон:  
 $y = \text{hardlim}(Wp + b)$

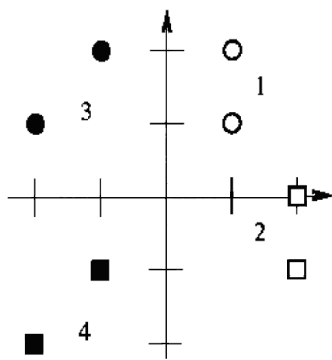


Рис. 2.13. Вхідні вектори

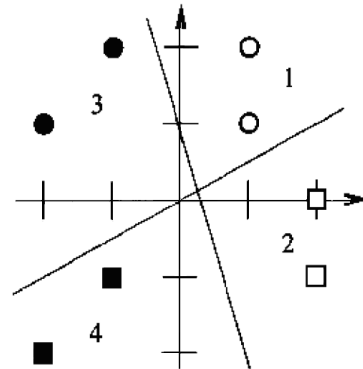


Рис. 2.14. Можливі границі поділу

Найтемніша область означає, що значення виходів обох нейронів дорівнюють одиниці. Цей розв'язок відповідає таким цільовим значенням:

$$\text{клас 1: } \left\{ \mathbf{t}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\};$$

$$\text{клас 2: } \left\{ \mathbf{t}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\};$$

$$\text{клас 3: } \left\{ \mathbf{t}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\};$$

$$\text{клас 4: } \left\{ \mathbf{t}_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Задамо вагові вектори:  ${}_1\mathbf{w} = [-3 \ -1]^T$

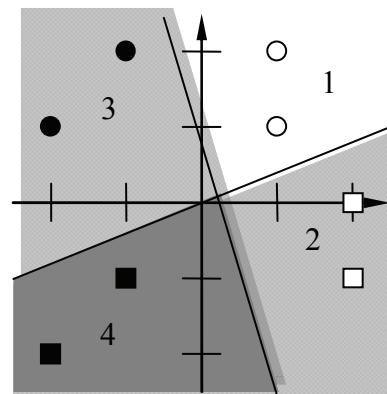


Рис. 2.15. Области розв'язку до прикл. 2.4

і  ${}_2\mathbf{w} = [1 \ -2]^T$ . Зазначимо, що довжина цих векторів не є суттєвою, а має значення лише їхній напрямок (вони мають бути ортогональними до граничних розв'язків). Далі визначимо зсув, обравши точки на границях і розв'язавши рівняння:

$$b_1 = -{}_1\mathbf{w}^T \mathbf{p}_1 = -[-3 \ -1] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1; \quad b_2 = -{}_2\mathbf{w}^T \mathbf{p}_2 = -[1 \ -2] \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0.$$

У матричному вигляді маємо:  $\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & -2 \end{bmatrix}$ ,  $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ , що завершує процес визначення мережі.

**Приклад 2.5.** Нехай маємо два класи вхідних векторів, які відповідають значенням  $t = 0$  і  $t = 1$ :  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0 \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1 \right\}; \left\{ \mathbf{p}_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0 \right\}; \left\{ \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}$ . Розв'язати задачу класифікації за допомогою правила навчання персептрона. Накреслити графік розв'язку. Як початкові значення ваги та зсуву розглянути такі:

$$\mathbf{w}(0) = [0 \ 0]^T, \quad b(0) = 0.$$

*Розв'язання. Крок 1.* Обчислимо значення виходу персептрона у для першого вхідного вектора  $\mathbf{p}_1$ :

$$y = \text{hardlim}(\mathbf{w}^T(0) \mathbf{p}_1 + b(0)) = \text{hardlim}\left([0 \ 0] \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right) = \text{hardlim}(0) = 1.$$

Оскільки  $y \neq t_1$ , то для знаходження ваги і зсуву можна використати правило навчання персептрона:

$$e = t_1 - y = 0 - 1 = -1;$$

$$\mathbf{w}(1) = \mathbf{w}(0) + e\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + (-1) \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix};$$

$$b(1) = b(0) + e = 0 + (-1) = -1.$$

Для другого вхідного вектора  $\mathbf{p}_2$ , використовуючи нові значення ваги та зсуву, одержимо:

$$y = \text{hardlim}(\mathbf{w}^T(1) \mathbf{p}_2 + b(1)) = \text{hardlim}\left([-2 \ -2] \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} - 1\right) = \text{hardlim}(1) = 1.$$

Оскільки  $y = t_2$ , то застосування правила навчання персептрона не приведе до змін, тобто  $\mathbf{w}(2) = \mathbf{w}(1)$ ,  $b(2) = b(1)$ .

Для третього вхідного вектора  $\mathbf{p}_3$  маємо

$$y = \text{hardlim}(\mathbf{w}^T(2) \mathbf{p}_3 + b(2)) = \text{hardlim}\left([-2 \ -2] \cdot \begin{bmatrix} -2 \\ 2 \end{bmatrix} - 1\right) = \text{hardlim}(-1) = 0.$$



Оскільки  $y = t_3$ , то вага та зсув не зміняться:  $\mathbf{w}(3) = \mathbf{w}(2)$ ,  $b(3) = b(2)$ .

Для останнього вхідного вектора  $\mathbf{p}_4$  маємо

$$y = \text{hardlim}(\mathbf{w}^T(3) \mathbf{p}_4 + b(3)) = \text{hardlim}\left(\begin{bmatrix} -2 & -2 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 1\right) = \text{hardlim}(-1) = 0.$$

Цього разу значення на виході  $y$  не збігається з цільовим значенням  $t_4$  і, застосувавши правило навчання персептрона, отримаємо нові значення  $\mathbf{w}$  і  $b$ :

$$e = t_4 - y = 1 - 0 = 1;$$

$$\mathbf{w}(4) = \mathbf{w}(3) + e\mathbf{p}_4 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} + (1) \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ -1 \end{bmatrix};$$

$$b(4) = b(3) + e = -1 + 1 = 0.$$

Крок 2. Знову перевіримо перший вхідний вектор  $\mathbf{p}_1$ :

$$y = \text{hardlim}(\mathbf{w}^T(4) \mathbf{p}_1 + b(4)) = \text{hardlim}\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right) = \text{hardlim}(-8) = 0.$$

Цього разу  $y = t_1$ , отже, вага та зсув не змінюють своїх значень, тобто  $\mathbf{w}(5) = \mathbf{w}(4)$ ,  $b(5) = b(4)$ . Подавши на вхід НМ вектор  $\mathbf{p}_2$ , одержимо похибку, яка вимагає корекції значень ваги та зсуву:

$$y = \text{hardlim}(\mathbf{w}^T(5) \mathbf{p}_2 + b(5)) = \text{hardlim}\left(\begin{bmatrix} -3 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0\right) = \text{hardlim}(-1) = 0.$$

Одержимо:

$$e = t_2 - y = 1;$$

$$b(6) = b(5) + e = 1;$$

$$\mathbf{w}(6) = \mathbf{w}(5) + e\mathbf{p}_2 = \begin{bmatrix} -3 \\ -1 \end{bmatrix} + (1) \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} -2 \\ -3 \end{bmatrix}.$$

Тепер введення вхідних векторів  $\mathbf{p}_1$ ,  $\mathbf{p}_2$ ,  $\mathbf{p}_3$  та  $\mathbf{p}_4$  до НМ не спричиняє похибок. Маємо:

$$y = \text{hardlim}(\mathbf{w}^T(6) \mathbf{p}_3 + b(6)) = \text{hardlim}\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_3;$$

$$y = \text{hardlim}(\mathbf{w}^T(6) \mathbf{p}_4 + b(6)) = \text{hardlim}\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 1\right) = 1 = t_4;$$

$$y = \text{hardlim}(\mathbf{w}^T(6) \mathbf{p}_1 + b(6)) = \text{hardlim}\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_1;$$

$$y = \text{hardlim}(\mathbf{w}^T(6) \mathbf{p}_2 + b(6)) = \text{hardlim}\left(\begin{bmatrix} -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1\right) = 1 = t_2.$$

Отже, алгоритм завершено й остаточний розв'язок має вигляд  $\mathbf{w} = [-2 \ -3]^T$ ,  $b = 1$ . Початкові дані навчання та границю поділу, яка описується рівнянням  $y = \mathbf{w}^T \mathbf{p} + b = w_{11} p_1 + w_{12} p_2 + b = -2 p_1 - 3 p_2 + 1 = 0$ , можна зобразити на графіку. Для визначення точки перетину границі поділу з координатною віссю  $p_2$  прирівнюємо  $p_1$  до нуля:  $p_2 = -\frac{b}{w_{12}} = \frac{1}{3}$ .

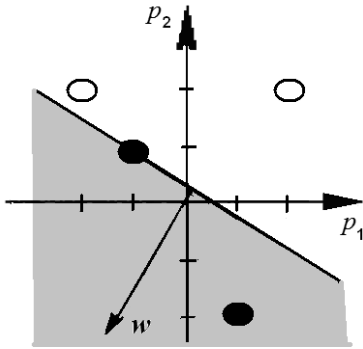


Рис. 2.16. Границя поділу до прикл. 2.5

Для визначення точки перетину границі поділу з координатною віссю  $p_1$  прирівнюємо  $p_2$  до нуля:  $p_1 = -\frac{b}{w_{11}} = \frac{1}{2}$ . Результуючу границю поділу, яка проходить через один із навчальних векторів, зображено на рис. 2.16.

Це допускається, оскільки функція *hardlim* набуває значення «1» для аргумента «0», а цільове значення для цього вектора дорівнює одиниці.

**Приклад 2.6.** Задано задачу класифікації з такими чотирма класами вхідних векторів:

$$\text{клас 1: } \left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\};$$

$$\text{клас 2: } \left\{ \mathbf{p}_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\};$$

$$\text{клас 3: } \left\{ \mathbf{p}_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \mathbf{t}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}; \left\{ \mathbf{p}_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\};$$

$$\text{клас 4: } \left\{ \mathbf{p}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{t}_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Розв'язати задачу класифікації векторів за допомогою правила навчання персептрона. Як початкові значення ваги та зсуву розглянути такі:  $\mathbf{W}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ,  $\mathbf{b}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ .

*Розв'язання. Крок 1.* Під час першої ітерації для вектора  $\mathbf{p}_1$  одержимо:

$$\mathbf{y} = \text{hardlim}(\mathbf{W}(0) \mathbf{p}_1 + \mathbf{b}(0)) = \text{hardlim} \left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix};$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{y} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix};$$

$$\mathbf{W}(1) = \mathbf{W}(0) + \mathbf{e} \mathbf{p}_1^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} \cdot [1 \quad 1] = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix};$$

$$\mathbf{b}(1) = \mathbf{b}(0) + \mathbf{e} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Під час другої ітерації для вектора  $\mathbf{p}_2$  одержимо:

$$\mathbf{y} = \text{hardlim}(\mathbf{W}(1)\mathbf{p}_2 + \mathbf{b}(1)) = \text{hardlim} \left( \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

$$\mathbf{W}(2) = \mathbf{W}(1), \mathbf{b}(2) = \mathbf{b}(1).$$

Під час третьої ітерації для вектора  $\mathbf{p}_3$  одержимо:

$$\mathbf{y} = \text{hardlim}(\mathbf{W}(2) \mathbf{p}_3 + \mathbf{b}(2)) = \text{hardlim} \left( \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 0 \end{bmatrix};$$

$$\mathbf{e} = \mathbf{t}_3 - \mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix};$$

$$\mathbf{W}(3) = \mathbf{W}(2) + \mathbf{e} \mathbf{p}_3^T = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \cdot [2 \quad -1] = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix};$$

$$\mathbf{b}(3) = \mathbf{b}(2) + \mathbf{e} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Введення векторів  $\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7, \mathbf{p}_8$  не приводить до зміни значень ваги та зсуву:  $\mathbf{W}(8) = \mathbf{W}(7) = \mathbf{W}(6) = \mathbf{W}(5) = \mathbf{W}(4) = \mathbf{W}(3)$ ,  $\mathbf{b}(8) = \mathbf{b}(7) = \mathbf{b}(6) = \mathbf{b}(5) = \mathbf{b}(4) = \mathbf{b}(3)$ .

Крок 2. Для вектора  $\mathbf{p}_1$  маємо:

$$\mathbf{y} = \text{hardlim}(\mathbf{W}(8) \mathbf{p}_1 + \mathbf{b}(8)) = \text{hardlim} \left( \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{y} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix};$$

$$\mathbf{W}(9) = \mathbf{W}(8) + \mathbf{e} \mathbf{p}_1^T = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \cdot [1 \quad 1] = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix};$$

$$\mathbf{b}(9) = \mathbf{b}(8) + \mathbf{e} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}.$$

Оскільки всі вхідні вектори правильно класифіковані, на цьому етапі алгоритм завершується. Остаточні границі поділу зображено на рис. 2.17.

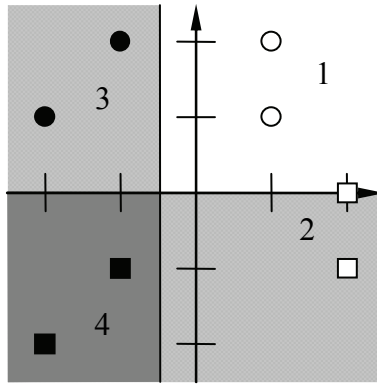


Рис. 2.17. Границі поділу до прикл. 2.6

### Контрольні запитання

1. Нарисуйте архітектуру нейрона типу персептрон.
2. Опишіть принцип функціонування (процедуру навчання) одно-нейронного персептрона.
3. Визначте ключову властивість персептрона, який складається з одного нейрона.
4. Опишіть принцип функціонування (процедуру навчання) одношарового персептрона.
5. Опишіть процедуру розв'язання задачі класифікації векторів на основі одношарового персептрона Ф. Розенблатта.

### Задачі для самостійного розв'язання

**Задача 2.1.** Задано завдання класифікації з двома класами вхідних векторів, які відповідають значенням  $t = 0$  і  $t = 1$ :  $\{\mathbf{p}_1 = [-1 \ 1]^T, t_1 = 1\}$ ;  $\{\mathbf{p}_2 = [0 \ 0]^T, t_2 = 1\}$ ;  $\{\mathbf{p}_3 = [1 \ -1]^T, t_3 = 1\}$ ;  $\{\mathbf{p}_4 = [1 \ 0]^T, t_4 = 0\}$ ;  $\{\mathbf{p}_5 = [0 \ 1]^T, t_5 = 0\}$ .

Розв'язати задачу класифікації за допомогою правила навчання персептрона. Як початкові значення ваги та зсуву розглянути такі:  $\mathbf{w}(0) = [0 \ 0]^T$ ,  $b(0) = 0$ . Нарисувати точки вхідних даних, позначити їх належність до класів на графіку розв'язку.

**Задача 2.2.** Задано задачу класифікації з двома класами вхідних векторів, які відповідають значенням  $t = 0$  і  $t = 1$ :  $\{\mathbf{p}_1 = [-1 \ 1]^T, t_1 = 1\}$ ;  $\{\mathbf{p}_2 = [-1 \ -1]^T, t_2 = 1\}$ ;  $\{\mathbf{p}_3 = [0 \ 0]^T, t_3 = 0\}$ ;  $\{\mathbf{p}_4 = [1 \ 0]^T, t_4 = 0\}$ . Визначити однеїронний персептрон для розв'язання цієї задачі (вибір

вагового вектора, ортогонального до границь поділу, виконати графічним способом). Перевірити розв'язок, використовуючи всі вхідні вектори.

**Задача 2.3.** Задано задачу класифікації з двома класами вхідних векторів (див. умову задачі 2.2). Визначити одонеуронний персептрон для розв'язання цієї задачі, використавши нерівності, які обмежують значення ваги та зсуву.

**Задача 2.4.** Задано задачу класифікації з двома класами вхідних векторів (див. умову задачі 2.2). Визначити одонеуронний персептрон для розв'язання цієї задачі. Розв'язати задачу класифікації за допомогою правила навчання персептрона з такими початковими значеннями ваги та зсуву:  $\mathbf{w}(0) = [0 \ 0]^T$ ;  $b(0) = 0$ .

**Задача 2.5.** Задано задачу класифікації з двома класами вхідних векторів:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = 0 \right\};$$
$$\left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_3 = 1 \right\}; \left\{ \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0 \right\}.$$

Довести математично (не графічно), що таку задачу неможливо розв'язати за допомогою одонеуронного персептрона (необхідно переписати умову задачі у формі нерівностей, які обмежують значення ваги та зсуву).

**Задача 2.6.** Іноді в персептронних НМ використовують симетричну порогову ФА *hardlims* (область її значень перебуває в межах інтервалу  $[-1, 1]$ ). Визначити функцію, яка відображає відрізок  $[0, 1]$  на відрізок  $[-1, 1]$ , а також функцію, яка виконує зворотнє відображення.

**Задача 2.7.** Задано два одонеуронних персептрона з однаковими значеннями ваги та зсуву. При цьому перша мережа використовує ФА *hardlim*, а друга – ФА *hardlims*. Чи будуть значення ваги залишатися однаковими для цих мереж, якщо на вхід подати один і той самий вектор  $\mathbf{p}$  і застосувати правило навчання персептрона? Маючи початкові значення ваги та зсуву для персептрона з ФА *hardlim*, запропонувати метод ініціалізації персептрона з ФА *hardlims* таким чином, щоб обидва нейрони мали однаковий вихід у разі навчання на одному наборі даних.

## Розділ 3. ЛІНІЙНІ ПЕРЕТВОРЕННЯ НЕЙРОННИХ МЕРЕЖ

### 3.1. Лінійні перетворення та їх матричне зображення

Зазвичай вхід і вихід мережі, а також рядки вагової матриці зручно зображувати у вигляді векторів, тому для розуміння теорії НМ слід знати такі поняття лінійної алгебри, як лінійний векторний простір, лінійно незалежні вектори, скалярний добуток, ортогональні вектори, процедура ортогоналізації Грама–Шмідта, розкладання векторів, обернені базисні вектори (дод. В).

Визначення добутку вагової матриці на вхідний вектор – одна з ключових операцій, яку виконує НМ. Ця операція є прикладом лінійного перетворення, яке використовують рекурентні мережі. *Рекурентна мережа* –

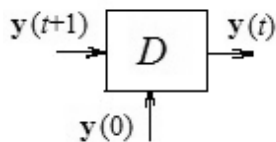


Рис. 3.1. Блок одинарної затримки  $y(t) = Dy(t+1)$  (відповідно до системи MatLab)

це мережа зі зворотним зв'язком, в якій частина виходів пов'язана із входами. Зворотний зв'язок передбачає використання елементів одинарної затримки (рис. 3.1) у вигляді операторів, які затримують вихідний сигнал відносно вхідного на один часовий крок. Вихід  $y(t)$  блока затримки обчислюють через його вхід  $y(t+1)$  за формулою  $y(t) = Dy(t+1)$ , яка вимагає, щоб вихід було визначено за значення  $t=0$  (цю початкову умову позначено на рис. 3.1 через  $y(0)$ ). Покажемо, що

операція множення матриць описує загальне лінійне перетворення (ЛП).

Розглянемо загальний вигляд ЛП та його основні характеристики. Поняття «власне число», «власний вектор» і «заміна базису» є важливими для розуміння ключових проблем НМ, таких як процес навчання та збіжність виходу рекурентної мережі Хопфілда. Зазначимо, що на кожній ітерації вихідне значення мережі множиться на вагову матрицю. Для ілюстрації змісту цієї операції та дослідження збіжності вихідного значення мережі розглянемо поняття «перетворення».

*Перетворенням* називають відображення  $\mathring{A}: X \rightarrow Y$ , яке кожному елементу  $x \in X$  ставить у відповідність елемент  $y \in Y$ , при цьому множину  $X$  називають областю визначення перетворення  $\mathring{A}$ , а множину  $Y$  – областю значень. Перетворення  $\mathring{A}$  називають *лінійним*, якщо:

- 1) для всіх  $x_1, x_2 \in X$ :  $\mathring{A}(x_1 + x_2) = \mathring{A}(x_1) + \mathring{A}(x_2)$ ;
- 2) для всіх  $x \in X$ ,  $a \in R$ :  $\mathring{A}(ax) = a\mathring{A}(x)$ .

Прикладом ЛП є поворот векторів у просторі  $\mathbb{R}^2$  на кут  $\theta$  (рис. 3.2, а). На рис. 3.2, б і в видно, що перший пункт означення поняття ЛП для цього

перетворення виконується: для того, щоб повернути суму векторів на кут  $\theta$ , можна спочатку повернути кожний з векторів окремо, а потім додати отримані вектори. Відповідно до другого пункту означення поняття ЛП (рис. 3.2, з) для того, щоб повернути вектор, помножений на деяке число, на кут  $\theta$ , можна спочатку повернути цей вектор, а потім помножити його на задане число.

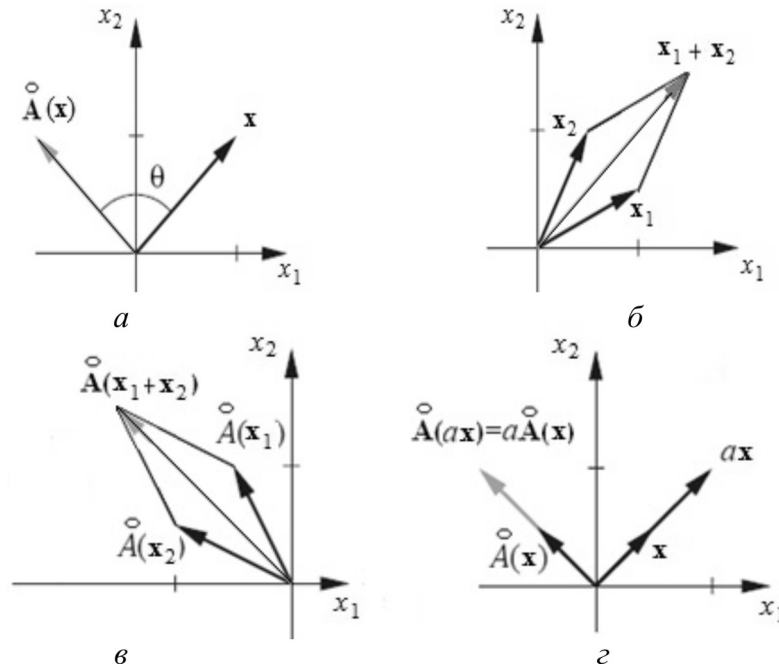


Рис. 3.2. Графічна ілюстрація ЛП у вигляді повороту векторів у просторі  $\mathbb{R}^2$  на кут  $\theta$

### 3.2. Матричне подання лінійних перетворень

Покажемо, що будь-яке ЛП двох векторних просторів скінченного розміру можна зобразити у вигляді матриці. Нагадаємо, що будь-який елемент векторного простору скінченного розміру можна зобразити у вигляді вектора-стовпця. Нехай  $\{v_1, \dots, v_n\}$  – базис векторного простору  $X$ ,  $\{u_1, \dots, u_m\}$  – базис векторного простору  $Y$ . Це означає, що будь-які два вектори  $x \in X$  і  $y \in Y$  можна зобразити у вигляді  $x = \sum_{i=1}^n x_i v_i$  та  $y = \sum_{i=1}^m y_i u_i$ .

Нехай  $\hat{A}$  – ЛП з областю визначення  $X$  та областю значень  $Y$  (тобто  $\hat{A}: X \rightarrow Y$ ). Тоді вираз  $\hat{A}(x) = y$  можна записати у вигляді

$$\hat{A} \left( \sum_{j=1}^n x_j v_j \right) = \sum_{i=1}^m y_i u_i. \quad (3.1)$$

Використавши властивість ЛП, з рівняння (3.1) одержимо:

$$\sum_{j=1}^n x_j \hat{A}(v_j) = \sum_{i=1}^m y_i u_i. \quad (3.2)$$

У зв'язку з тим, що вектори  $\mathring{A}(\mathbf{v}_j)$  є елементами векторного простору  $Y$ , а  $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$  – його базисом, то  $\mathring{A}(\mathbf{v}_j)$  можна записати у вигляді

$$\mathring{A}(\mathbf{v}_j) = \sum_{i=1}^m a_{ij} \mathbf{u}_i. \quad (3.3)$$

Якщо підставити вираз (3.3) у (3.2), одержимо:  $\sum_{j=1}^n x_j \sum_{i=1}^m a_{ij} \mathbf{u}_i = \sum_{i=1}^m y_i \mathbf{u}_i$ .

Змінивши порядок підсумовування, одержимо:  $\sum_{i=1}^m \mathbf{u}_i \sum_{j=1}^n a_{ij} x_j = \sum_{i=1}^m y_i \mathbf{u}_i$ ,

звідки маємо

$$\sum_{i=1}^m \mathbf{u}_i \sum_{j=1}^n (a_{ij} x_j - y_i) = 0. \quad (3.4)$$

Оскільки вектори  $\mathbf{u}_i$  формують базисну множину, то вони незалежні. Це означає, що кожний коефіцієнт, який множиться на  $\mathbf{u}_i$  у рівнянні (3.4), повинен дорівнювати нулю, тому маємо  $\sum_{j=1}^n a_{ij} x_j = y_i$ , або в матри-

чній формі:  $\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix}$ . Таким чином, для кожного ЛП

двох векторних просторів скінченного розміру існує матрична форма зображення, при цьому кожне рівняння вигляду (3.3) визначає стовпець матриці перетворення. Слід пам'ятати, що матрична форма зображення не визначена однозначно (так само, як і зображення звичайного вектора у вигляді вектора-стовпця чисел). Якщо змінити базисну множину для області визначення або для області значень, матричне зображення теж зміниться. Використаємо цю властивість далі.

**Приклад 3.1.** Розглянемо ЛП у вигляді повороту векторів у просторі  $\mathcal{R}^2$  на кут  $\theta$ . Визначимо матрицю, яка відображає це перетворення. Для цього спочатку необхідно перетворити кожний базисний вектор області визначення, а потім розкласти його за базисними векторами області значень. У цьому прикладі область визначення збігається з областю значень ( $X = Y = \mathcal{R}^2$ ).

Використаємо стандартний базис для обох областей:  $\mathbf{u}_i = \mathbf{v}_i = \mathbf{s}_i$ ,  $i = 1, 2$  (рис. 3.3, а). Спочатку перетворимо перший базисний вектор  $\mathbf{s}_1$  і розкладемо його за базисними векторами. Якщо повернути вектор  $\mathbf{s}_1$  проти годинникової стрілки на кут  $\theta$ , то одержимо (рис. 3.3, б):

$$\mathring{A}(\mathbf{s}_1) = \cos(\theta) \mathbf{s}_1 + \sin(\theta) \mathbf{s}_2 = a_{11} \mathbf{s}_1 + a_{21} \mathbf{s}_2.$$



Перетворимо другий базисний вектор: повернувши вектор  $s_2$  проти годинникової стрілки на кут  $\theta$ , одержимо (рис. 3.3, в):

$$\hat{A}(s_2) = -\sin(\theta)s_1 + \cos(\theta)s_2 = a_{11}s_1 + a_{22}s_2.$$

Матриця перетворення  $A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$

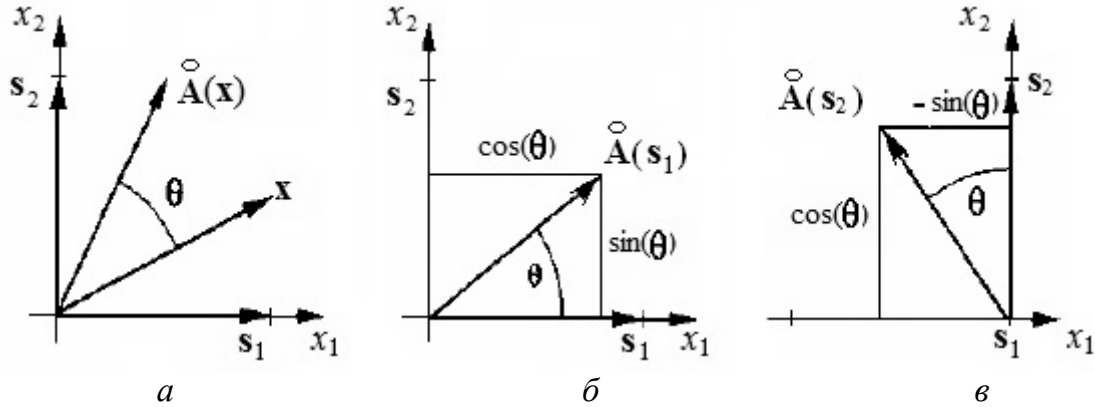


Рис. 3.3. Графічна ілюстрація прикл. 3.1

Якщо помножити вектор  $x$  на матрицю  $A$ , то в результаті отримаємо поворот цього вектора на кут  $\theta$ . Таким чином, матриця  $A$  відображає перетворення  $\hat{A}$ .

### 3.3. Зміна базису

Як було показано вище, матричне зображення ЛП не є унікальним: воно залежить від вибору базисних множин області визначення та області значень перетворення. Розглянемо вплив зміни базисної множини на зміну матричного зображення ЛП [14]. Припустимо, що  $\hat{A}: X \rightarrow Y$  – ЛП,  $\{v_1, \dots, v_n\}$  – базис векторного простору  $X$ ;  $\{u_1, \dots, u_m\}$  – базис векторного простору  $Y$ .

Будь-який вектор  $x \in X$  можна зобразити у вигляді  $x = \sum_{i=1}^n x_i v_i$ , а будь-який

вектор  $y \in Y$  – у вигляді  $y = \sum_{i=1}^m y_i u_i$ . Якщо  $\hat{A}(x) = y$ , то матричне зображен-

ня має вигляд 
$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix},$$

або

$$Ax = y. \quad (3.5)$$

Нехай  $\{t_1, \dots, t_n\}$  і  $\{w_1, \dots, w_m\}$  – нові базиси векторних просторів  $X$  і  $Y$  відповідно. Тоді вектори  $x$  і  $y$  можна розкласти за цими базисами

таким чином:  $\mathbf{x} = \sum_{i=1}^n x'_i \mathbf{t}_i$ ,  $\mathbf{y} = \sum_{i=1}^m y'_i \mathbf{w}_i$ . Одержимо нове матричне зобра-

ження перетворення: 
$$\begin{bmatrix} a'_{11} & \dots & a'_{1n} \\ \dots & \dots & \dots \\ a'_{m1} & \dots & a'_{mn} \end{bmatrix} \cdot \begin{bmatrix} x'_1 \\ \dots \\ x'_n \end{bmatrix} = \begin{bmatrix} y'_1 \\ \dots \\ y'_m \end{bmatrix},$$

або

$$\mathbf{A}'\mathbf{x}' = \mathbf{y}'. \quad (3.6)$$

Щоб визначити зв'язок між матрицями  $\mathbf{A}$  та  $\mathbf{A}'$ , необхідно знайти відношення між двома базисними множинами.

Кожний вектор  $\mathbf{t}_i$  є елементом векторного простору  $X$ , тому його можна розкласти за базисом  $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ :  $\mathbf{t} = \sum_{j=1}^n t_{ji} \mathbf{v}_j$ . Аналогічно кожен

вектор  $\mathbf{w}_i$  можна розкласти за базисом  $\{\mathbf{u}_1, \dots, \mathbf{u}_m\}$ :  $\mathbf{w}_i = \sum_{j=1}^m w_{ji} \mathbf{u}_j$ . Таким чином, базисні вектори  $\mathbf{t}_i$ ,  $\mathbf{w}_i$  можна записати у вигляді стовпців чисел:

$$\mathbf{t}_i = \begin{bmatrix} t_{1i} \\ \dots \\ t_{ni} \end{bmatrix}; \quad \mathbf{w}_i = \begin{bmatrix} w_{1i} \\ \dots \\ w_{mi} \end{bmatrix}.$$

Визначимо матрицю, стовпцями якої є вектори  $\mathbf{t}_i$ , як  $\mathbf{B}_t = [\mathbf{t}_1 \dots \mathbf{t}_n]$ . Тоді можна зобразити вектор  $\mathbf{x}$  у матричній формі (зв'язок між двома різними формами зображення вектора  $\mathbf{x}$ ):

$$\mathbf{x} = x'_1 \mathbf{t}_1 + \dots + x'_n \mathbf{t}_n = \mathbf{B}_t \mathbf{x}'. \quad (3.7)$$

Визначимо матрицю, стовпцями якої є вектори  $\mathbf{w}_i$ , як  $\mathbf{B}_w = [\mathbf{w}_1 \dots \mathbf{w}_m]$ . Тоді вектор  $\mathbf{y}$  у матричній формі має такий вигляд (зв'язок між двома різними формами зображення вектора  $\mathbf{y}$ ):

$$\mathbf{y} = \mathbf{B}_w \mathbf{y}'. \quad (3.8)$$

Тепер підставимо рівняння (3.7) і (3.8) у (3.5), одержимо:  $\mathbf{A}\mathbf{B}_t \mathbf{x}' = \mathbf{B}_w \mathbf{y}'$ . Помноживши обидві частини цього рівняння на  $\mathbf{B}_w^{-1}$ , одержимо:

$$[\mathbf{B}_w^{-1} \mathbf{A} \mathbf{B}_t] \mathbf{x}' = \mathbf{y}'. \quad (3.9)$$

Порівнюючи вирази (3.6) і (3.9), одержимо формулу для заміни базису (зв'язок між двома матричними формами, які зображають одне ЛП):

$$\mathbf{A}' = \mathbf{B}_w^{-1} \mathbf{A} \mathbf{B}_t. \quad (3.10)$$

**Приклад 3.2** (зміна базисної множини). Знову розглянемо приклад 3.1, коли матричне зображення було отримано за допомогою стандартної базисної множини  $\{s_1, s_2\}$ . Визначимо нове зображення, використовуючи вектори  $\{t_1, t_2\}$ , подані на рис. 3.4. Нагадаємо, що в прикладі 3.1 застосовано одну базисну множину для областей визначення  $X$  та значень  $Y$ . Розкладемо  $t_1$  і  $t_2$  на стандартну базисну множину. Досліджуючи рис. 3.4, виявимо, що:  $t_1 = s_1 + 0,5s_2$ ,  $t_2 = -s_1 + s_2$ . Тому можна записати:  $t_1 = \begin{bmatrix} 1 \\ 0,5 \end{bmatrix}$ ;  $t_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ .

Сформуємо матрицю  $B_t$ :  $B_t = [t_1 \ t_2] = \begin{bmatrix} 1 & -1 \\ 0,5 & 1 \end{bmatrix}$ . Оскільки використано одну й ту саму базисну множину для областей визначення й значень, перетворення має вигляд  $B_w = B_t = \begin{bmatrix} 1 & -1 \\ 0,5 & 1 \end{bmatrix}$ . Використовуючи рівняння (3.10), можемо розрахувати нове матричне зображення заміни базису:

$$\begin{aligned} A' = B_w^{-1} A B_t &= \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ 0,5 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} \frac{1}{3} \sin \theta + \cos \theta & -\frac{4}{3} \sin \theta \\ \frac{5}{6} \sin \theta & -\frac{1}{3} \sin \theta + \cos \theta \end{bmatrix}. \end{aligned}$$

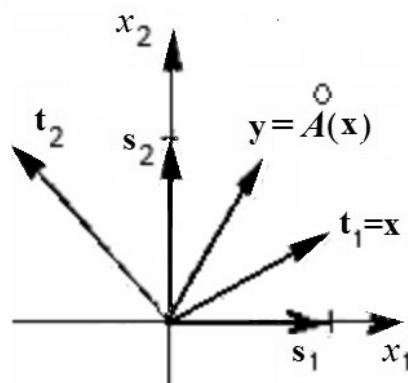


Рис. 3.4. Графічна ілюстрація до прикл. 3.2

Якщо  $\theta = 30^\circ$ , маємо:  $A' = \begin{bmatrix} 1,033 & -0,667 \\ 0,417 & 0,699 \end{bmatrix}$ ;  $A = \begin{bmatrix} 0,866 & -0,5 \\ 0,5 & 0,866 \end{bmatrix}$ .

Щоб перевірити, чи ці матриці є правильними, скористаємося

вектором-тестом:  $\mathbf{x} = \begin{bmatrix} 1 \\ 0,5 \end{bmatrix}$ , якому відповідає вектор  $\mathbf{x}' = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ . Перетворений вектор-тест має такий вигляд:  $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0,866 & -0,5 \\ 0,5 & 0,866 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0,5 \end{bmatrix} = \begin{bmatrix} 0,616 \\ 0,933 \end{bmatrix}$ , що повинно відповідати  $\mathbf{y}' = \mathbf{A}'\mathbf{x}' = \begin{bmatrix} 1,033 & -0,667 \\ 0,416 & 0,699 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1,033 \\ 0,416 \end{bmatrix}$ . Вектори  $\mathbf{x}$ ,  $\mathbf{y}$  зображено на рис. 3.3, а.

Перевіримо, чи відповідають один одному вектори  $\mathbf{y}'$  й  $\mathbf{y}$ : вони обидва мають бути формами зображення одного вектора  $\mathbf{y}$  на різних базисних множинах:  $\mathbf{y}$  – на базисі  $\{\mathbf{s}_1, \mathbf{s}_2\}$ , а  $\mathbf{y}'$  – на базисі  $\{\mathbf{t}_1, \mathbf{t}_2\}$ . Вище поворотні базисні вектори було використано для перетворення однієї форми зображення на іншу. Відповідно до цієї концепції маємо:

$$\mathbf{y}' = \mathbf{B}^{-1}\mathbf{y} = \begin{bmatrix} 1 & -1 \\ 0,5 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0,616 \\ 0,933 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} 0,616 \\ 0,933 \end{bmatrix} = \begin{bmatrix} 1,033 \\ 0,416 \end{bmatrix}, \text{ що під-}$$

тверджує попередній результат.

### 3.4. Ключові поняття, пов'язані з лінійними перетвореннями

Розглянемо ключові поняття, пов'язані з лінійними перетвореннями: власні значення та власні вектори. Нехай  $\mathbf{A}: X \rightarrow X$  – ЛП. Ненульові вектори  $\mathbf{z} \in X$  та скаляри  $\lambda$ , що задовольняють умову

$$\mathbf{A}(\mathbf{z}) = \lambda\mathbf{z}, \quad (3.11)$$

називають відповідно *власними векторами* ( $\mathbf{z}$ ) і *власними значеннями* ( $\lambda$ ). Зазначимо, що власні вектори утворюють векторний простір, оскільки, якщо  $\mathbf{z}$  задовольняє рівняння (3.11), то  $a\mathbf{z}$  теж буде задовольняти це рівняння за довільного значення  $a$ . Знову розглянемо приклад 3.1 перетворення у вигляді операції обертання. Зазначимо, що вектора, який при повороті на кут  $30^\circ$  продовжував би вказувати в тому ж напрямку, не існує, оскільки не існує дійсних власних значень.

Покажемо, як можна обчислити власні значення та власні вектори. Припустимо, що був обраний базис для  $n$ -вимірного векторного простору  $X$ . Тоді матрична форма для рівняння (3.11) набуде вигляду  $\mathbf{A}\mathbf{z} = \lambda\mathbf{z}$  або  $(\mathbf{A} - \lambda\mathbf{I})\mathbf{z} = 0$ . Це означає, що стовпці матриці  $[\mathbf{A} - \lambda\mathbf{I}]$  залежні, тому

її визначник повинен дорівнювати нулю, тобто  $|\mathbf{A} - \lambda\mathbf{I}| = 0$ . Цей визначник є багаточленом степеня  $n$ . Отже, рівняння  $|\mathbf{A} - \lambda\mathbf{I}| = 0$  завжди має  $n$  коренів, деякі з яких можуть бути комплексними або повторюватися.

Якщо використати стандартну базисну множину (прикл. 3.1), то матриця перетворення  $\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ , тому рівняння  $|\mathbf{A} - \lambda\mathbf{I}| = 0$  можна записати у вигляді  $\begin{vmatrix} \cos\theta - \lambda & -\sin\theta \\ \sin\theta & \cos\theta - \lambda \end{vmatrix} = 0$ , або  $\lambda^2 - 2\lambda\cos\theta + ((\cos\theta)^2 + (\sin\theta)^2) = \lambda^2 - 2\lambda\cos\theta + 1 = 0$ . Корені цього рівняння  $\lambda_1 = \cos\theta + i\sin\theta$ ;  $\lambda_2 = \cos\theta - i\sin\theta$ . Отже, таке перетворення не має дійсних власних значень (якщо  $\sin\theta \neq 0$ ). Це означає, що під час перетворення будь-якого дійсного вектора він буде мати новий напрямок.

Розглянемо іншу матрицю:  $\mathbf{A} = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}$ . Щоб визначити її власні значення, необхідно розв'язати рівняння  $\begin{vmatrix} -1-\lambda & 1 \\ 0 & -2-\lambda \end{vmatrix} = 0$ , або  $\lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2) = 0$ . Одержимо такі власні значення:  $\lambda_1 = -1$ ,  $\lambda_2 = -2$ . Щоб визначити власні вектори, необхідно розв'язати рівняння вигляду  $\begin{bmatrix} -1-\lambda & 1 \\ 0 & -2-\lambda \end{bmatrix} \mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ . Одержимо такі варіанти:

1. Якщо  $\lambda = \lambda_1$ , то маємо  $\begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \mathbf{z}_1 = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , або  $z_{21} = 0$  без обмежень для  $z_{11}$ . Отже, першим власним вектором є  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ , або  $a\mathbf{z}_1$ , де  $a$  – будь-який скалярний множник.

2. Якщо  $\lambda = \lambda_2$ , то маємо  $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{z}_2 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} z_{12} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , або  $z_{22} = -z_{11}$ .

Отже, другим власним вектором є  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ , або  $a\mathbf{z}_2$ , де  $a$  – будь-який скалярний множник.

Для перевірки одержаних результатів розглянемо перетворення:

$$\mathbf{A}\mathbf{z}_1 = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} = (-1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda_1 \mathbf{z}_1;$$

$$\mathbf{A}\mathbf{z}_2 = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} = (-2) \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_2 \mathbf{z}_2.$$

Визначимо матрицю перетворення подібності, використовуючи власні вектори як базисні. З рівняння (3.10) одержимо:

$$\mathbf{A}' = \mathbf{B}_w^{-1} \mathbf{A} \mathbf{B}_t = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}.$$

Зазначимо, що це діагональна матриця, на діагоналі якої стоять власні значення. Таким чином, якщо відомі власні значення, то матрицю перетворення можна звести до діагонального вигляду, використовуючи власні вектори як базисні.

### 3.5. Зведення лінійних перетворень до діагонального вигляду

Відомо, що за наявності  $n$  різних власних значень можна гарантовано знайти  $n$  незалежних власних векторів. Отже, власні вектори утворюють базисну множину для векторного простору перетворення. Нехай  $\mathbf{B} = [\mathbf{z}_1 \dots \mathbf{z}_n]$ , де  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$  – власні вектори симетричної матриці

перетворення  $\mathbf{A}$ . Тоді маємо:  $\mathbf{B}^{-1} \mathbf{A} \mathbf{B} = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \lambda_n \end{bmatrix}$ , де  $\{\lambda_1, \dots, \lambda_n\}$  – влас-

ні значення матриці  $\mathbf{A}$ . Цей результат є корисним для аналізу функціонування деяких НМ.

**Приклад. 3.3.** Нехай матриця перетворення  $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ . Тоді маємо

$$\begin{vmatrix} 1-\lambda & 1 \\ 1 & 1-\lambda \end{vmatrix} = 0, \text{ або } \lambda^2 - 2\lambda = \lambda(\lambda - 2) = 0. \text{ Одержимо такі власні значення: } \lambda_1 = 0; \lambda_2 = 2. \text{ Визначимо власні вектори:}$$

1) якщо  $\lambda_1 = 0$ , маємо

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{z}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ або } z_{21} = -z_{11}, \mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix};$$

2) якщо  $\lambda_2 = 2$ , маємо  $\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{z}_2 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} z_{12} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , або  $z_{22} = z_{12}$ ,

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Діагональна форма матриці перетворення має вигляд

$$\mathbf{A}' = \mathbf{B}^{-1} \mathbf{A} \mathbf{B} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}.$$

### Приклади розв'язання задач\*

**Приклад 3.4.** Нехай одонеуронний персептрон задано у вигляді мережі (рис. 3.5). Граничний розв'язок для цієї мережі задано у вигляді  $\mathbf{w}^T \mathbf{p} + b = 0$ . Показати, що за  $b = 0$  граничний розв'язок має вигляд лінійного векторного простору.

*Розв'язання.* У випадку, коли  $b = 0$ , граничний розв'язок буде лінійним векторним простором, якщо для нього виконуються десять умов відповідного означення (дод. В). Спочатку перевіримо виконання першої умови.

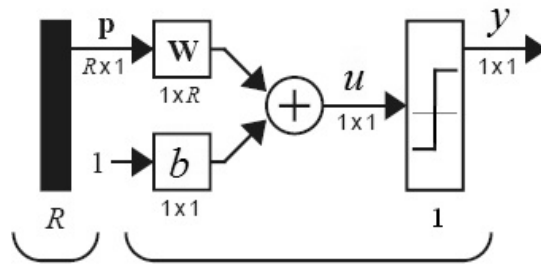


Рис. 3.5. Одонеуронний персептрон:  $y = \text{hardlims}(\mathbf{w}^T \mathbf{p} + b)$

Нехай  $\mathbf{p}_1, \mathbf{p}_2$  – два вектори, розміщені на граничному розв'язку, тобто  $\mathbf{w}^T \mathbf{p}_1 = 0$ ;  $\mathbf{w}^T \mathbf{p}_2 = 0$ . Додавши ці дві рівності, отримаємо:  $\mathbf{w}^T (\mathbf{p}_1 + \mathbf{p}_2) = 0$ . Тому вектор  $(\mathbf{p}_1 + \mathbf{p}_2)$  також розміщений на граничному розв'язку.

Виконання другої та третьої умов очевидне.

Четверта умова виконується, оскільки нульовому вектору відповідає граничний розв'язок  $\mathbf{w}^T \cdot \mathbf{0} = 0$ .

Якщо вектор  $\mathbf{p}$  розміщений на граничному розв'язку, то  $\mathbf{w}^T \mathbf{p} = 0$ . Помноживши обидві частини цієї рівності на  $-1$ , отримаємо:  $\mathbf{w}^T (-\mathbf{p}) = 0$ . Отже, вектор  $-\mathbf{p}$  також міститься на граничному розв'язку й п'ята умова виконується.

Аналогічно, помноживши обидві частини рівності  $\mathbf{w}^T \mathbf{p} = 0$  на  $a$ , можна показати виконання шостої умови. Виконання сьомої, восьмої, дев'ятої і десятої умов очевидне. Отже, граничний розв'язок одонеуронного персептрона з нульовим зсувом має вигляд лінійного векторного простору.

\* Задачі взято з посібника [14].

**Приклад 3.5.** Показати, що множина  $Y$  невід'ємних неперервних функцій ( $f(t) \geq 0$ ) не є лінійним векторним простором.

*Розв'язання.* Покажемо, що для множини  $Y$  не виконуються п'ята і шоста умови означення лінійного векторного простору (дод. В). П'ята умова не виконується, оскільки множина  $Y$  не містить жодного від'ємного вектора.

Розглянемо функцію  $f(t) = |t|$ , яка є елементом множини  $Y$ . Тоді за  $a = -2$  одержимо:  $af(2) = -2 \cdot |2| = -4 < 0$  і  $af(t)$  не належить  $Y$ , і шоста умова не виконується.

**Приклад 3.6.** Встановити, які з наведених груп векторів лінійно незалежні, визначити розмірність векторного простору, породженого кожною групою:

$$\text{I. } \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}. \quad \text{II. } \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}.$$

*Розв'язання. I.* Розглянемо декілька способів розв'язання цієї задачі.

*Перший спосіб.* Припустимо, що вектори є лінійно залежними. Тоді виконується така рівність:

$$a_1 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + a_2 \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + a_3 \cdot \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (3.12)$$

Ця рівність виконується за  $a_1 = 2$ ,  $a_2 = -1$  та  $a_3 = -1$ , тому вектори є лінійно залежними.

*Другий спосіб.* Відповідно до матричної форми запису рівності (3.12)

$$\text{маємо } \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \text{ Якщо матриця } \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix} \text{ у цьому рівнянні}$$

має обернену матрицю (її визначник не дорівнює нулю), то всі коефіцієнти дорівнюють нулю і вектори лінійно незалежні. Якщо хоча б один з коефіцієнтів не дорівнює нулю, то вектори є лінійно залежними. Використовуючи розкладання визначника матриці за першим стовпцем, отримаємо

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \end{vmatrix} = 1 \cdot \begin{vmatrix} 0 & 2 \\ 1 & 1 \end{vmatrix} + (-1) \cdot \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix} + 1 \cdot \begin{vmatrix} 1 & 1 \\ 0 & 2 \end{vmatrix} = -2 + 0 + 2 = 0.$$

Отже, розглянуті вектори є лінійно залежними. Розмірність простору, породженого цими векторами, дорівнює двом, оскільки можна показати, що будь-які два вектори із цього набору є лінійно незалежними.



**II.** Розглянемо інший випадок, коли кількість векторів менша, ніж розмірність векторного простору (три вектори із  $\mathbb{R}^4$ ). Тому матриця, що складається з цих векторів, не є квадратною і не можна обчислити її визначник, однак можна використати грам'яни – визначник матриці Грама, в якій  $ij$ -м елементом є скалярний добуток  $i$ -го та  $j$ -го векторів. Вектори є лінійно залежними, якщо грам'яни дорівнює нулю:

$$G = \begin{vmatrix} (\mathbf{x}_1, \mathbf{x}_1) & (\mathbf{x}_1, \mathbf{x}_2) & (\mathbf{x}_1, \mathbf{x}_3) \\ (\mathbf{x}_2, \mathbf{x}_1) & (\mathbf{x}_2, \mathbf{x}_2) & (\mathbf{x}_2, \mathbf{x}_3) \\ (\mathbf{x}_3, \mathbf{x}_1) & (\mathbf{x}_3, \mathbf{x}_2) & (\mathbf{x}_3, \mathbf{x}_3) \end{vmatrix},$$

де  $\mathbf{x}_1 = [1 \ 1 \ 1 \ 1]^T$ ;  $\mathbf{x}_2 = [1 \ 0 \ 1 \ 1]^T$ ;  $\mathbf{x}_3 = [1 \ 2 \ 1 \ 1]^T$ , звідки маємо:

$$G = \begin{vmatrix} 4 & 3 & 5 \\ 3 & 3 & 3 \\ 5 & 3 & 7 \end{vmatrix} = 4 \cdot \begin{vmatrix} 3 & 3 \\ 3 & 7 \end{vmatrix} + (-3) \cdot \begin{vmatrix} 3 & 5 \\ 3 & 7 \end{vmatrix} + 5 \cdot \begin{vmatrix} 3 & 5 \\ 3 & 3 \end{vmatrix} = 48 - 18 - 30 = 0.$$

Можна показати, що ці вектори є лінійно залежними:

$$2\mathbf{x}_1 - 1\mathbf{x}_2 - 1\mathbf{x}_3 = [0 \ 0 \ 0 \ 0]^T.$$

Зазначимо, що вектори  $\mathbf{x}_1, \mathbf{x}_2$  є лінійно незалежними, оскільки  $G = \begin{vmatrix} 4 & 3 \\ 3 & 3 \end{vmatrix} = 4 \neq 0$ . Отже, розмірність простору, породженого векторами  $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ , дорівнює двом.

**Приклад 3.7.** Нагадаємо, що одношаровий перцептрон можна використати для розпізнавання образів, якщо вони є лінійно подільними. Визначити, чи будуть два лінійно подільні образи лінійно незалежними.

*Розв'язання.* Розглянемо перцептрон із двома входами (рис. 3.6, а). Нехай необхідно поділити два вектори  $\mathbf{p}_1 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} 1,5 \\ 1,5 \end{bmatrix}$  на два класи. Якщо вагу та зсув вибрати таким чином (рис. 3.6, б):  ${}_1\mathbf{w}^T = [w_{11} \ w_{12}] = [1 \ 1]$ ,  $b = -2$ , то граничний розв'язок становитиме  ${}_1\mathbf{w}^T \mathbf{p} + b = 0$ , тому вектори  $\mathbf{p}_1, \mathbf{p}_2$  є лінійно подільними, однак вони не є лінійно незалежними, оскільки  $\mathbf{p}_2 = 3\mathbf{p}_1$ .

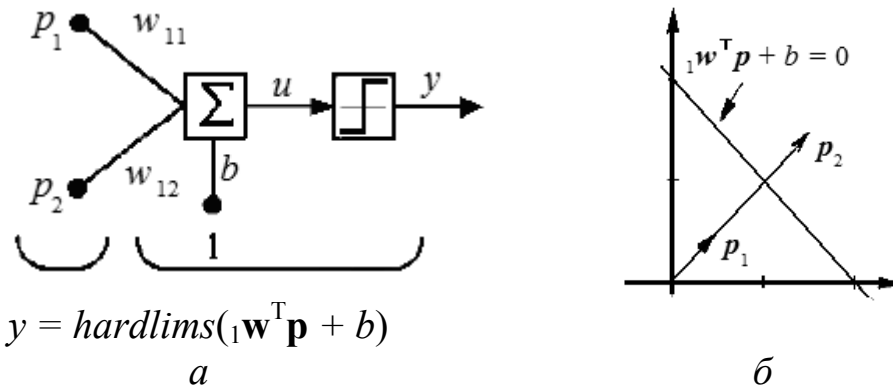


Рис. 3.6. Графічна ілюстрація до прикл. 3.7: а – одонеуронний перцептрон; б – приклад лінійно залежних векторів

**Приклад 3.8.** Провести процедуру ортогоналізації Грама–Шмідта для векторів  $\mathbf{y}_1 = [1 \ 1 \ 1]^T$ ;  $\mathbf{y}_2 = [1 \ 0 \ 0]^T$ ;  $\mathbf{y}_3 = [0 \ 1 \ 0]^T$ .

*Розв’язання.* Використаємо такий алгоритм.

*Крок 1:*  $\mathbf{v}_1 = \mathbf{y}_1 = [1 \ 1 \ 1]^T$ .

*Крок 2:*

$$\mathbf{v}_2 = \mathbf{y}_2 - \frac{\mathbf{v}_1^T \mathbf{y}_2}{\mathbf{v}_1^T \mathbf{v}_1} \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \frac{[1 \ 1 \ 1] \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{[1 \ 1 \ 1] \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}.$$

*Крок 3:*

$$\begin{aligned} \mathbf{v}_3 = \mathbf{y}_3 - \frac{\mathbf{v}_1^T \mathbf{y}_3}{\mathbf{v}_1^T \mathbf{v}_1} \mathbf{v}_1 - \frac{\mathbf{v}_2^T \mathbf{y}_3}{\mathbf{v}_2^T \mathbf{v}_2} \mathbf{v}_2 &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \frac{[1 \ 1 \ 1] \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}{[1 \ 1 \ 1] \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \frac{\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}{\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{3} \\ \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}} \times \\ &\times \begin{bmatrix} \frac{2}{3} \\ \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}. \end{aligned}$$

**Приклад 3.9.** Показати, що вираз  $(\mathbf{x}, \mathbf{y}) = \int_{-1}^1 x(t)y(t)dt$  задає скалярний добуток на лінійному векторному просторі всіх багаточленів, визначених на відрізку  $[-1; 1]$ .

*Розв’язання.* Перевіримо виконання умов, які має задовольняти скалярний добуток:

1)  $(\mathbf{x}, \mathbf{x}) = \int_{-1}^1 x(t)x(t)dt = \int_{-1}^1 x^2(t)dt \geq 0$ , рівність виконується тоді й тільки тоді, коли  $x(t) = 0$ ,  $-1 \leq t \leq 1$ ;

$$2) (\mathbf{x}, \mathbf{y}) = \int_{-1}^1 x(t)y(t)dt = \int_{-1}^1 y(t)x(t)dt = (\mathbf{y}, \mathbf{x});$$

$$3) (\mathbf{x}, a\mathbf{y}_1 + b\mathbf{y}_2) = \int_{-1}^1 x(t)(ay_1(t) + by_2(t))dt = a \int_{-1}^1 (x(t), y_1(t))dt + b \int_{-1}^1 (x(t), y_2(t))dt = a(\mathbf{x}, \mathbf{y}_1) + b(\mathbf{x}, \mathbf{y}_2).$$

**Приклад 3.10.** Провести процедуру ортогоналізації Грамма–Шмідта для векторів  $\mathbf{y}_1(t) = 1 + t$ ;  $\mathbf{y}_2(t) = 1 - t$  з лінійного векторного простору всіх багаточленів, визначених на відрізку  $[-1; 1]$ .

*Розв’язання.* Використаємо такий алгоритм.

*Крок 1:*  $\mathbf{v}_1 = \mathbf{y}_1 = 1 + t$ .

$$\text{Крок 2: } \mathbf{v}_2 = \mathbf{y}_2 - \frac{(\mathbf{v}_1, \mathbf{y}_2)}{(\mathbf{v}_1, \mathbf{v}_1)} \mathbf{v}_1,$$

де

$$(\mathbf{v}_1, \mathbf{y}_2) = \int_{-1}^1 (1+t)(1-t)dt = \left(t - \frac{t^3}{3}\right) \Big|_{-1}^1 = \frac{2}{3} - \left(\frac{2}{3}\right) = \frac{4}{3};$$

$$(\mathbf{v}_1, \mathbf{v}_1) = \int_{-1}^1 (1+t)^2 dt = \frac{(1+t)^3}{3} \Big|_{-1}^1 = \frac{8}{3} - (0) = \frac{8}{3}.$$

$$\text{Отже, } \mathbf{v}_2 = (1-t) - \frac{\frac{4}{3}}{\frac{8}{3}}(1+t) = \frac{1}{2} - \frac{3}{2}t.$$

**Приклад 3.11.** Розкласти вектор  $\mathbf{x}^s = [6 \quad 9 \quad 9]^T$  за базисом:  $\mathbf{v}_1 = [1 \quad 1 \quad 1]^T$ ;  $\mathbf{v}_2 = [1 \quad 2 \quad 3]^T$ ;  $\mathbf{v}_3 = [1 \quad 3 \quad 2]^T$ .

*Розв’язання.* Визначимо обернений базис  $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ .

$$\text{Нехай } \mathbf{V} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{bmatrix}, \text{ тоді } \mathbf{V}^T = \mathbf{V}^{-1} = \begin{bmatrix} \frac{5}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix}, \text{ тому маємо:}$$

$$\mathbf{b}_1 = \begin{bmatrix} \frac{5}{3} \\ -\frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}; \mathbf{b}_2 = \begin{bmatrix} -\frac{1}{3} \\ -\frac{1}{3} \\ \frac{2}{3} \end{bmatrix}; \mathbf{b}_3 = \begin{bmatrix} -\frac{1}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix}. \text{ Обчислимо коефіцієнти розкладу век-}$$

$$\text{тора } \mathbf{x}^s \text{ за базисом } \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}: a_1 = \mathbf{b}_1^T \mathbf{x}^s = \begin{bmatrix} \frac{5}{3} & -\frac{1}{3} & -\frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 4;$$

$$a_2 = \mathbf{b}_2^T \mathbf{x}^s = \begin{bmatrix} -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 1; a_3 = \mathbf{b}_3^T \mathbf{x}^s = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 1.$$

$$\text{Отже, розкладання має вигляд } \mathbf{x}^s = a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + a_3 \mathbf{v}_3 = 4 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} +$$

$$+ 1 \cdot \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}, \text{ або у матричній формі: } \mathbf{a} = \mathbf{V}^{-1} \mathbf{x}^s = \begin{bmatrix} \frac{5}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}.$$

Вектори  $\mathbf{x}^s$ ,  $\mathbf{a}$  зображують один і той самий вектор у різних базисах (вектор  $\mathbf{x}^s$  використовує стандартний базис у  $\mathcal{R}^3$ ).

**Приклад 3.12.** Розглянемо одношарову мережу (рис. 3.7), яка має лінійну ФА. Визначити, чи є перетворення в НМ вектора входу у вектор виходу лінійним.

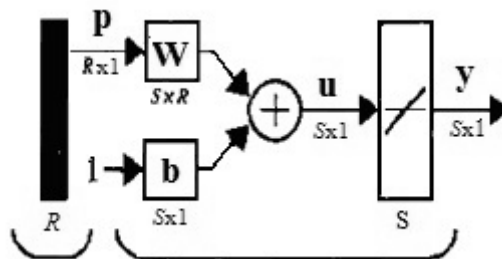


Рис. 3.7. Одношаровий перцептрон:  $\mathbf{y} = \text{purelin}(\mathbf{W}\mathbf{p} + \mathbf{b})$

*Розв'язання.* Рівняння мережі  $y = \mathring{A}(p) = Wp + b$ . Для того, щоб це перетворювання було лінійним, воно має задовольняти такі умови:  $\mathring{A}(p_1 + p_2) = \mathring{A}(p_1) + \mathring{A}(p_2)$  та  $\mathring{A}(ap) = a\mathring{A}(p)$ .

Спочатку перевіримо першу умову, одержимо:  $\mathring{A}(p_1 + p_2) = W(p_1 + p_2) + b = Wp_1 + Wp_2 + b$ , порівняємо її з виразом  $\mathring{A}(p_1) + \mathring{A}(p_2) = Wp_1 + b + Wp_2 + b = Wp_1 + Wp_2 + 2b$ . Очевидно, що ці два вирази будуть збігатися лише у випадку, коли  $b = 0$ , тому ця мережа реалізує нелінійне перетворення, хоча вона має лінійну ФА.

**Приклад 3.13.** Проекцію вектора  $p$  на вектор  $v$  можна розрахувати за формулою  $y = \mathring{A}(p) = \frac{(p, v)}{(v, v)} v$ , де  $(p, v)$  – це скалярний добуток векторів  $p$  і  $v$ . Чи є лінійним перетворенням операція проєкції вектора  $p$  на вектор  $v$ ?

*Розв'язання.* Щоб показати, що  $\mathring{A}$  – ЛП, необхідно перевірити, чи задовольняє це перетворення дві умови лінійності. Перевіримо першу умову:  $\mathring{A}(p_1 + p_2) = \frac{(p_1 + p_2, v)}{(v, v)} v = \frac{(p_1, v) + (p_2, v)}{(v, v)} v = \frac{(p_1, v)}{(v, v)} v + \frac{(p_2, v)}{(v, v)} v = \mathring{A}(p_1) + \mathring{A}(p_2)$ , де використано властивість лінійності скалярних добутків. Перевіримо другу умову:  $\mathring{A}(ap) = \frac{(ap, v)}{(v, v)} v = \frac{a(p, v)}{(v, v)} v = a\mathring{A}(p)$ . Отже, операція проєкції є ЛП.

**Приклад 3.14.** Нехай  $\mathring{A}$  – перетворення, одержане відображенням вектора  $x$  на простір  $\mathcal{R}^2$  відносно прямої  $x_1 + x_2 = 0$  (рис. 3.8). Визначити матрицю цього перетворення щодо стандартного базису в  $\mathcal{R}^2$ .

*Розв'язання.* У цьому випадку базисна множина має такий вигляд:  $\{s_1, s_2\}$ . Виконаємо перетворення векторів  $s_1, s_2$  відносно прямої  $x_1 + x_2 = 0$ .

Одержимо для вектора  $s_1$  (рис. 3.9, а):  $\mathring{A}(s_1) = -s_2 = a_{11}s_1 + a_{21}s_2 = 0s_1 + (-1)s_2$ , для вектора  $s_2$  (рис. 3.9, б):  $\mathring{A}(s_2) = -s_1 = a_{12}s_1 + a_{22}s_2 = (-1)s_1 + (0)s_2$ . Отже, матриця  $A$  має вигляд  $A = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$ . Перевіримо одержаний результат, виконавши перетворення вектора  $x = [1 \quad 1]^T$ :  $Ax = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ . Це і буде відображенням вектора  $x$  відносно прямої  $x_1 + x_2 = 0$ .

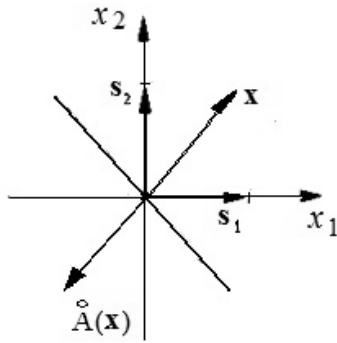
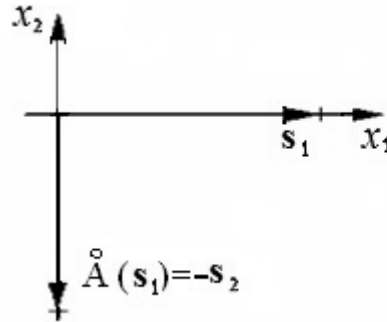
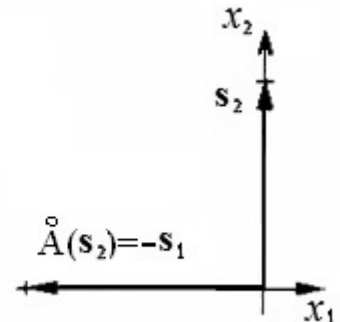


Рис. 3.8. Графічний вигляд перетворення  $\hat{A}$



а

Рис. 3.9. Перетворення векторів  $s_1$  і  $s_2$  відносно прямої  $x_1 + x_2 = 0$



б

**Приклад 3.15.** Розглянемо векторний простір комплексних чисел  $X$  із базисною множиною:  $\{1 + i, 1 - i\}$ . Нехай  $\hat{A}: X \rightarrow X$  – перетворення (оператор) спряження, тобто для довільного  $z \in X$ ,  $z = a + bi$ :  $\hat{A}(z) = a - bi$ .

**I.** Визначити матрицю перетворення  $A$  відносно базисної множини  $\{1 + i, 1 - i\}$ .

**II.** Знайти власні значення та власні вектори цього перетворення.

**III.** Визначити матричне зображення  $A$  відносно власних векторів як базисних.

*Розв'язання.* **I.** Позначимо  $v_1 = 1 + i$ ;  $v_2 = 1 - i$ . Розглянемо перетворення кожного з базисних векторів, визначаючи спряжений до базисного вектор:

$$\hat{A}(v_1) = \hat{A}(1 + i) = 1 - i = v_2 = a_{11}v_1 + a_{21}v_2 = 0v_1 + 1v_2;$$

$$\hat{A}(v_2) = \hat{A}(1 - i) = 1 + i = v_1 = a_{12}v_1 + a_{22}v_2 = 1v_1 + 0v_2.$$

Це дозволяє одержати таке матричне зображення перетворення:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

**II.** Щоб визначити власні значення, необхідно розв'язати рівняння  $|\mathbf{A} - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & 1 \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - 1 = (\lambda - 1)(\lambda + 1) = 0$ . Отже, маємо такі власні значення:  $\lambda_1 = 1$ ,  $\lambda_2 = -1$ . Щоб визначити власні вектори, треба розв'язати

$$\text{рівняння } [\mathbf{A} - \lambda \mathbf{I}] \mathbf{z} = \begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix} \mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}:$$

$$\text{— якщо } \lambda = \lambda_1 = 1, \text{ то } \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{z}_1 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ або } z_{11} = z_{21},$$

тому, перший власний вектор має такий вигляд:  $z_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ , або цей вектор помножений на будь-який скаляр;

$$\text{— якщо } \lambda = \lambda_2 = -1, \text{ то } \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} z_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} z_{12} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ або } z_{12} = -z_{22},$$

тому другий власний вектор має такий вигляд:  $z_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ , або цей вектор помножений на будь-який скаляр.

Зазначимо, що хоча власні вектори  $z_1$  і  $z_2$  зображені у вигляді стовпців чисел, насправді це комплексні числа:

$$z_1 = 1v_1 + 1v_2 = (1+i) + (1-i) = 2; z_2 = 1v_1 + (-1)v_2 = (1+i) - (1-i) = 2i.$$

Покажемо, що вони насправді є власними векторами, одержимо:  $\hat{A}(z_1) = 2 = \lambda_1 z_1$ ;  $\hat{A}(z_2) = -2i = \lambda_2 z_2$ . Визначимо власні значення такого перетворення за допомогою системи MatLab:

$$A = [0, 1; 1, 0], \quad [V, D] = \text{eig}(A) \\ V = \begin{bmatrix} -0.7071 & 0.7071 \\ 0.7071 & 0.7071 \end{bmatrix} \quad D = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Діагональні елементи матриці  $D$  зображують власні значення  $\lambda_1$  і  $\lambda_2$ , тоді як стовпці матриці  $V$  зображують власні вектори.

**III.** Для зміни базису необхідно використати рівняння  $A' = B_w^{-1} A B_t = B^{-1} A B$ , де  $B = [z_1 \ z_2] = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ , тому маємо  $A' = \begin{bmatrix} 0,5 & 0,5 \\ 0,5 & -0,5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ .

Одержано діагональне матричне зображення перетворення.

**Приклад 3.16.** Нехай  $B = [z_1 \ \dots \ z_n]$ , де  $\{z_1, \dots, z_n\}$  – власні вектори матриці  $A = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}$ . Звести матрицю  $A$  до діагонального вигляду

$$\begin{bmatrix} \lambda_1 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \lambda_n \end{bmatrix}, \text{ де } \{\lambda_1, \dots, \lambda_n\} \text{ – власні значення матриці } A.$$

*Розв'язання.* Спочатку визначимо власні значення матриці  $A$ :

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{vmatrix} = \lambda^2 - 5\lambda + 4 = (\lambda - 1)(\lambda - 4) = 0. \text{ Отже, маємо такі власні значення: } \lambda_1 = 1; \lambda_2 = 4. \text{ Для визначення власних векторів}$$

розв'яжемо рівняння  $[\mathbf{A} - \lambda \mathbf{I}] \mathbf{z} = \begin{bmatrix} 2 - \lambda & -2 \\ -1 & 3 - \lambda \end{bmatrix} \mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ :

— якщо  $\lambda = \lambda_1 = 1$ , то  $\begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix} \mathbf{z}_2 = \begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , або  $z_{11} = 2z_{12}$ ,

тому перший власний вектор, наприклад, має вигляд  $\mathbf{z}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ;

— якщо  $\lambda = \lambda_2 = 4$ , то  $\begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix} \mathbf{z}_2 = \begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , або  $z_{21} = -z_{22}$ ,

тому другий власний вектор, наприклад, має вигляд  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ .

Для зведення матриці до діагонального вигляду використаємо рівняння:  $\mathbf{A}' = \mathbf{B}^{-1} \mathbf{A} \mathbf{B}$ , де  $\mathbf{B} = [\mathbf{z}_1 \ \mathbf{z}_2] = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$ . Отже, одержимо:

$$\mathbf{A}' = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

**Приклад 3.17.** Розглянемо перетворення  $\mathcal{A}: \mathcal{R}^3 \rightarrow \mathcal{R}^2$ , матричне зображення якого відносно стандартної базисної множини має вигляд  $\mathbf{A} = \begin{bmatrix} 3 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ . Визначити матрицю перетворення  $\mathbf{A}'$  відносно базис-

них множин:  $T = \left\{ \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}; \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}; \begin{bmatrix} 0 \\ -2 \\ 3 \end{bmatrix} \right\}$ ;  $W = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \begin{bmatrix} 0 \\ -2 \end{bmatrix} \right\}$ .

*Розв'язання.* Сформуємо базисні матриці:  $\mathbf{B}_T = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & -2 \\ 1 & 0 & 3 \end{bmatrix}$ ;

$\mathbf{B}_W = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}$ . Нову матрицю перетворення  $\mathbf{A}'$  відносно базисних множин  $T$  і  $W$  можна отримати з рівняння



$$\mathbf{A}' = \mathbf{B}_W^{-1} \mathbf{A} \mathbf{B}_T = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 3 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & -2 \\ 1 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 6 & 1 & 2 \\ -\frac{1}{2} & 0 & -\frac{3}{2} \end{bmatrix}.$$

**Приклад 3.18.** Розглянемо перетворення  $\hat{A}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . Одну базисну множину для  $\mathbb{R}^2$  задано у вигляді  $V = \{\mathbf{v}_1, \mathbf{v}_2\}$ .

**I.** Визначити матрицю перетворення  $\mathbf{A}$  відносно базисної множини  $V$ , якщо відомо, що:  $\hat{A}(\mathbf{v}_1) = \mathbf{v}_1 + 2\mathbf{v}_2$ ;  $\hat{A}(\mathbf{v}_2) = \mathbf{v}_1 + \mathbf{v}_2$ .

**II.** Знайти матрицю перетворення  $\mathbf{A}$  відносно нової базисної множини  $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2\}$ , якщо  $\mathbf{w}_1 = \mathbf{v}_1 + \mathbf{v}_2$ ;  $\mathbf{w}_2 = \mathbf{v}_1 - \mathbf{v}_2$ .

*Розв'язання.* **I.** Кожне з двох заданих рівнянь надає стовпець матриці  $\mathbf{A}$ , тому матриця перетворення має такий вигляд:  $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$ .

**II.** Зобразимо множину базисних векторів  $W$  у такому вигляді:  $\mathbf{w}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{w}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ . Тепер можна сформулювати базисну матрицю, необхідну для виконання перетворень:  $\mathbf{B}_w = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ . Нову матрицю перетворення можна отримати з рівняння

$$\mathbf{A}' = \mathbf{B}_w^{-1} \mathbf{A} \mathbf{B}_w = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \frac{5}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

**Приклад 3.19.** Розглянемо векторний простір  $P^2$  всіх багаточленів, степені яких не перевищує двох. Однією з можливих базисних множин для цього векторного простору є множина  $V = \{1, t, t^2\}$ ; розглянемо перетворення диференціювання  $\hat{D}$ . Визначити матрицю цього перетворення відносно базисної множини  $V$ , власні значення та власні вектори цього перетворення.

*Розв'язання.* Визначимо перетворення кожного з базисних векторів:  $\hat{D}(1) = 0 = (0)1 + (0)t + (0)t^2$ ;  $\hat{D}(t) = 1 = (1)1 + (0)t + (0)t^2$ ;  $\hat{D}(t^2) = 2t = (0)1 + (2)t + (0)t^2$ . Отже, матриця перетворення матиме вигляд

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

Для визначення власних значень необхідно розв'язати рівняння

$$|\mathbf{D} - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 2 \\ 0 & 0 & -\lambda \end{vmatrix} = -\lambda^3 = 0, \text{ тому всі три власні значення дорівнюють нулю.}$$

Для визначення власних векторів необхідно розв'язати рівняння

$$[\mathbf{D} - \lambda \mathbf{I}] \mathbf{z} = \begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 2 \\ 0 & 0 & -\lambda \end{bmatrix} \mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \text{ Якщо } \lambda = 0, \text{ одержимо } \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Це означає, що  $z_2 = z_3 = 0$ . Отже, маємо єдиний власний вектор:  $\mathbf{z} = [1 \ 0 \ 0]^T$ .

**Приклад 3.20.** Розглянемо перетворення  $\mathring{A}: \mathcal{R}^2 \rightarrow \mathcal{R}^2$ . Визначити матричне зображення перетворення, заданого у вигляді прикладів перетворення векторів відносно стандартної базисної множини (рис. 3.10).

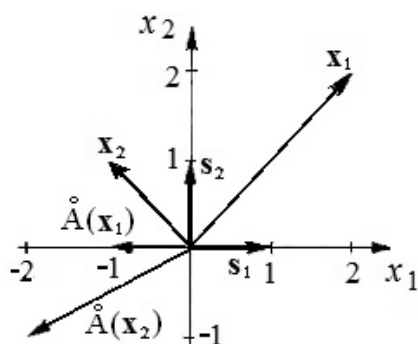


Рис. 3.10. Перетворення векторів

*Розв'язання.* З умови задачі невідомо, як перетворюються базисні вектори, тому для визначення матричного зображення перетворення не можна використовувати рівняння (3.12), однак відомо, як ці вектори були перетворені та як їх можна зобразити за допомогою стандартної базисної множини. Використовуючи рис. 3.10, можна записати такі рівняння:

$$\mathring{A} \left( \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}; \quad \mathring{A} \left( \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -2 \\ -1 \end{bmatrix}. \text{ Якщо}$$

$$\text{об'єднати обидва рівняння, одержимо } \mathbf{A} \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 \\ 0 & -1 \end{bmatrix}.$$

Отже, маємо:

$$\mathbf{A} = \begin{bmatrix} -1 & -2 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} -1 & -2 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{5}{4} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} - \text{матричне}$$

зображення перетворення відносно стандартної базисної множини.

## Контрольні запитання

1. Опишіть процедуру ортогоналізації векторів Грамма–Шмідта (див. дод. В, п. В7).
2. Дайте визначення поняттю «лінійне перетворення». Опишіть загальний вигляд ЛП. Які його основні характеристики?
3. Визначте вплив зміни базисної множини на зміну матричного зображення ЛП.
4. Опишіть процедуру зведення лінійних перетворень до діагонального вигляду.

## Задачі для самостійного розв'язання

**Задача 3.1.** Нехай одонеуронний персептрон заданий у вигляді мережі (рис. 3.5). Граничний розв'язок для цієї мережі має вигляд  $\mathbf{w}^T \mathbf{p} + b = 0$ . Покажіть, що граничний розв'язок є лінійним векторним простором за  $b \neq 0$ .

**Задача 3.2.** Розгляньте всі можливі варіанти неперервних функцій, які задовольняють умову  $f(0) = 0$ . Доведіть, що ці функції утворюють лінійний векторний простір.

**Задача 3.3.** Які з наведених груп векторів є лінійно незалежними? Визначте розмір векторного простору, утвореного кожною групою.

$$1) \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}; \quad 2) 1 + t, 1 - t; \quad 3) \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \\ 4 \\ 3 \end{bmatrix}.$$

**Задача 3.4.** Визначте кути між кожним із наведених векторів  $\mathbf{p}_1$  (образ апельсина),  $\mathbf{p}_2$  (образ яблука) та тестовим вектором-зразком  $\mathbf{p}_3$  (довгий апельсин):  $\mathbf{p}_1 = [1 \ -1 \ -1]^T$ ;  $\mathbf{p}_2 = [1 \ 1 \ -1]^T$ ;  $\mathbf{p}_3 = [-1 \ -1 \ -1]^T$ . Переконайтеся, що ці кути можна отримати інтуїтивно.

**Задача 3.5.** Використовуючи задані базисні вектори  $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ , за допомогою ортогоналізації Грамма–Шмідта визначте ортогональні до них вектори:  $\mathbf{y}_1 = [1 \ 0 \ 0]^T$ ;  $\mathbf{y}_2 = [1 \ 1 \ 0]^T$ ;  $\mathbf{y}_3 = [1 \ 1 \ 1]^T$ .

**Задача 3.6.** Розкладіть вектор  $\mathbf{x}^S = [1 \ 2 \ 2]^T$  за базисом:  $\mathbf{v}_1 = [-1 \ 1 \ 0]^T$ ;  $\mathbf{v}_2 = [1 \ 1 \ -2]^T$ ,  $\mathbf{v}_3 = [1 \ 1 \ 0]^T$ .

**Задача 3.7.** Чи є операція транспонування матриці лінійним перетворенням?

**Задача 3.8.** Розгляньте лінійне перетворення  $\mathring{A}$ , зображене на рис. 3.11. Визначте матричне зображення цього перетворення відносно стандартної базисної множини  $\{\mathbf{s}_1, \mathbf{s}_2\}$  та матрицю цього перетворення для базисної множини  $\{\mathbf{v}_1, \mathbf{v}_2\}$ .

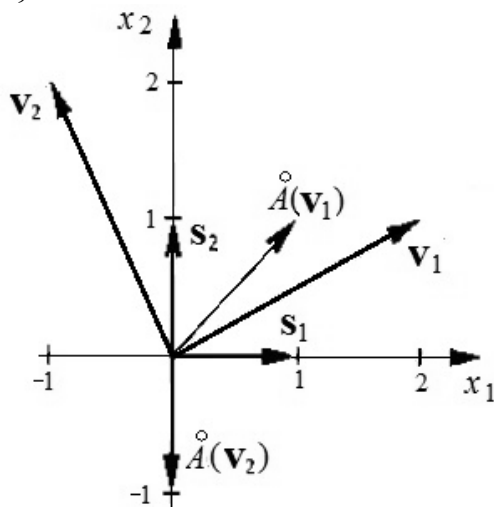


Рис. 3.11. Приклад перетворення

**Задача 3.9.** Нехай  $X$  – векторний простір комплексних чисел з базисною множиною  $\{1 + i, 1 - i\}$ ,  $\mathring{A}: X \rightarrow X$  – перетворення (оператор) множення на  $(1 + i)$ , тобто для довільного  $\mathbf{z} \in X$ :  $\mathring{A}(\mathbf{z}) = \mathbf{z}(1 + i)$ . Визначте: а) матрицю перетворення  $\mathbf{A}$  відносно базисної множини  $\{1 + i, 1 - i\}$ ; б) власні значення та власні вектори такого перетворення; в) матричне зображення перетворення  $\mathbf{A}$  відносно власних векторів як базисних.

**Задача 3.10.** Розгляньте перетворення  $\mathring{A}: P^2 \rightarrow P^3$  із простору поліномів другого порядку у простір поліномів третього порядку, яке визначається таким чином:  $\mathbf{x} = a_0 + a_1 t + a_2 t^2$ ;  $\mathring{A}(\mathbf{x}) = a_0(t + 1) + a_1(t + 1)^2 + a_2(t + 1)^3$ . Визначте матричне зображення  $\mathbf{A}$  цього перетворення відносно базисних множин  $V^2 = \{1, t, t^2\}$  і  $V^3 = \{1, t, t^2, t^3\}$ .

**Задача 3.11.** Нехай лінійне перетворення  $\mathring{A}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$  має матричне зображення  $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  відносно стандартної базисної множини  $\{\mathbf{s}_1, \mathbf{s}_2\}$ . Визначте матричне зображення цього перетворення відносно нової базисної множини:  $V = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \end{bmatrix} \right\}$ .

**Задача 3.12.** Нехай лінійне перетворення  $\mathring{A}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$  має такі власні значення та власні вектори відносно стандартної базисної множини  $\{\mathbf{s}_1, \mathbf{s}_2\}$ :  $\lambda_1 = 1$ ,  $\mathbf{z}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ ;  $\lambda_2 = 2$ ,  $\mathbf{z}_2 = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ . Визначте матрицю перетворення  $\mathbf{A}$

відносно стандартної базисної множини  $\{\mathbf{s}_1, \mathbf{s}_2\}$  та матричне зображення перетворення  $A'$  відносно нової базисної множини:  $V = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}$ .

**Задача 3.13.** Нехай задано базис для  $\mathbb{R}^2$  вигляду:  $V = \{\mathbf{v}_1, \mathbf{v}_2\} = \left\{ \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right\}$  відносно стандартної базисної множини  $\{\mathbf{s}_1, \mathbf{s}_2\}$ . Визначте взаємно-обернені базисні вектори для цієї базисної множини.

**Задача 3.14.** Нехай задано перетворення  $A: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ , матричне зображення якого відносно стандартного базису  $\mathbb{R}^2$  має вигляд  $A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$ . Визначте розкладання  $A(\mathbf{v}_1), A(\mathbf{v}_2)$  відносно базису  $V$  (використайте взаємнообернені базисні вектори). Визначте матрицю перетворення  $A$  відносно базису  $V$ .

## Розділ 4. АСОЦІАТИВНЕ КОНТРОЛЬОВАНЕ НАВЧАННЯ ХЕББА

### 4.1. Постулат навчання Хебба

*Асоціація* – це зв'язок між входом системи та її виходом, за якого образ *A*, поданий на вхід системи, зумовлює на виході образ *B*. Зазвичай, якщо два образи пов'язані асоціацією, то вхідний образ зображається у вигляді вхідного сигналу, а вихідний – у вигляді вихідного.

Асоціативна пам'ять має вигляд системи, яка визначає взаємну залежність векторів. Головне її завдання зводиться до запам'ятовування образів у вигляді векторів-прототипів навчання таким чином, щоб після введення нового образу система змогла згенерувати відповідь, який із векторів-прототипів є найбільш близьким до введенного в систему. Найчастіше мірою близькості окремих векторів виступає міра Хеммінга.

У 1949 р. Дональд Хебб у роботі «Організація поведінки» підвів підсумок своїх досліджень за два десятиліття і запропонував можливий механізм синаптичних змін у мозку людини, який згодом почав використовуватися для навчання штучних НМ. Найбільше досягнення Хебба – це постулат, відомий як навчання Хебба, згідно з яким:

*Якщо аксон клітини A перебуває достатньо близько до клітини B, щоб впливати на її стан, і постійно або періодично бере участь у її збудженні, то спостерігається процес метаболічних змін в одному або в обох нейронах, який полягає в тому, що ефективність нейрона A як одного зі збудників нейрона B зростає.*

Хебб запропонував покласти це спостереження в основу процесу асоціативного навчання (на клітинному рівні). На його думку, це мало привести до постійної модифікації активності просторово розподіленого «ансамблю нервових клітин». Це твердження було зроблене в нейробіологічному контексті, але його можна перефразувати у вигляді правила, яке складається з двох частин:

1. Якщо два нейрони по обидва боки синаптичного з'єднання активізуються одночасно (синхронно), то міцність цього з'єднання зростає.

2. Якщо два нейрони по обидва боки синапсу активізуються асинхронно, то такий синапс послаблюється або взагалі відмирає.

Синапс, який функціонує таким чином, називають синапсом Хебба.

Розглянемо алгоритм навчання Хебба (одне з перших правил навчання НМ) і покажемо, як правило Хебба можна використати у навчанні мереж розпізнавати образи.

## 4.2. Лінійний асоціатор і контрольоване правило навчання Хебба

У 1972 р. Джеймс Андерсон і Тейво Кохонен незалежно один від одного запропонували лінійний асоціатор (ЛА) (рис. 4.1) – одношарову мережу, вихід якої визначається за формулою  $y = \text{purelin}(\mathbf{Wp}) = \mathbf{Wp}$ , або

$$y_i = \sum_{j=1}^R w_{ij} p_j, 1 \leq i \leq S. \quad (4.1)$$

Лінійний асоціатор – приклад мережі, яку називають *асоціативною пам'яттю*. Асоціативна пам'ять є однією з основних функцій мозку людини: наприклад, співвіднести обличчя товариша з його іменем, ім'я з номером телефона.

Нехай на основі множини навчання у вигляді  $Q$  пар даних вхід–вихід,  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , мережа навчається таким чином, що якщо на вхід мережі подається вектор  $\mathbf{p} = \mathbf{p}_q$ , то вихід становитиме  $\mathbf{y} = \mathbf{t}_q$  для  $q = 1, \dots, Q$ . Крім того, якщо вхідні дані трохи зміняться, тобто  $\mathbf{p} = \mathbf{p}_q + \delta$ , то вихідні дані також трохи зміняться:  $\mathbf{y} = \mathbf{t}_q + \mathbf{e}$ . Якщо  $\mathbf{t}_q = \mathbf{p}_q$ , то маємо *автоасоціативну пам'ять*. Покажемо, як можна математично інтерпретувати постулат Хебба, щоб використовувати його для навчання ЛА (або налаштування вагової матриці).

Спочатку перефразуємо постулат таким чином: «Якщо два нейрони з обох боків синапсу активізуються одночасно, то сила синаптичного зв'язку зросте». Із формули (4.1) одержимо, що синаптичний зв'язок між входом  $\mathbf{p}_j$  і виходом  $\mathbf{y}_i$  зображується у вигляді вагового коефіцієнта  $w_{ij}$ . Отже, постулат Хебба стверджує, що якщо додатний вхід  $\mathbf{p}_j$  породжує додатний вихід  $\mathbf{y}_i$ , то значення  $w_{ij}$  має зростати.

Одна з математичних інтерпретацій постулату Хебба має такий вигляд:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha f_i(y_{iq}) g_j(p_{jq}), \quad (4.2)$$

де  $p_{jq}$  –  $j$ -й елемент  $q$ -го вхідного вектора  $\mathbf{p}_q$ ;  $y_{iq}$  –  $i$ -й елемент виходу мережі, якщо до мережі подається вхідний вектор  $\mathbf{p}_q$ ;  $\alpha$  – додатна константа, яку називають коефіцієнтом навчання;  $g_j(p_{jq})$  – передсинаптичний сигнал;  $f_i(y_{iq})$  – постсинаптичний сигнал.

Рівняння (4.2) підтверджує, що вага  $w_{ij}$  змінюється пропорційно добутку функцій активності обох боків синапсу. Спростивши рівняння (4.2), отримаємо

$$w_{ij}^{new} = w_{ij}^{old} + \alpha y_{iq} p_{jq}. \quad (4.3)$$

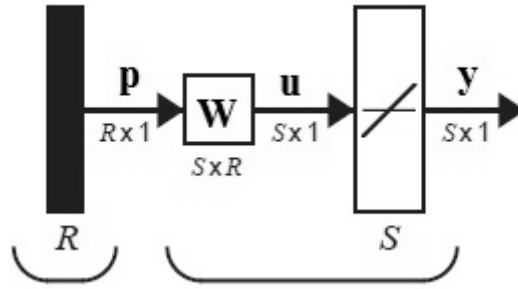


Рис. 4.1. Лінійний асоціатор:  $y = \text{purelin}(Wp)$

Правило Хебба, визначене рівнянням (4.3) – це неконтрольоване правило навчання, яке не вимагає ніякої інформації щодо цільового виходу. Рівняння (4.3) розширює постулат Хебба за межі його чіткої інтерпретації: оскільки зміна ваги відбувається пропорційно добутку функцій активності з обох боків синапсу, то вага  $w_{ij}$  збільшується не тільки коли  $p_{jq}$  та  $y_{iq}$  додатні, але й коли обидва вони є від’ємними. Крім того, з рівняння (4.3) видно, що вага зменшується, коли  $p_j$  та  $y_i$  мають протилежні знаки.

Розглянемо, як правило Хебба використовують у випадку контрольованого навчання (навчання з учителем), коли цільовий вихід відомий для кожного вхідного вектора. Для цього замінимо фактичний вихід цільовим, повідомляючи мережі, що саме вона повинна робити. У результаті отримаємо рівняння (для простоти візьмемо коефіцієнт навчання  $\alpha = 1$ )

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}, \quad (4.4)$$

де  $t_{iq}$  –  $i$ -й елемент  $q$ -го цільового вектора  $\mathbf{t}_q$ . У векторній формі рівняння (4.4) має такий вигляд:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T. \quad (4.5)$$

Якщо припустити, що початковим значенням вагової матриці  $\mathbf{W}$  є нульова матриця й до кожної з  $Q$  пар вхід–ціль застосовано рівняння (4.5), то матрицю  $\mathbf{W}$  можна записати у вигляді  $\mathbf{W} = \sum_{q=1}^Q \mathbf{t}_q \mathbf{p}_q^T$ , або в матричній формі:

$$\mathbf{W} = [\mathbf{t}_1 \dots \mathbf{t}_Q] \cdot \begin{bmatrix} \mathbf{p}_1^T \\ \dots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T} \mathbf{P}^T, \quad (4.6)$$

де  $\mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_Q]$ ;  $\mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_Q]$ .



### 4.3. Аналіз ефективності навчання Хебба

Проаналізуємо ефективність навчання Хебба для лінійного асоціатора, розглянемо два випадки.

**Перший випадок.** Вектори  $\mathbf{p}_q$  є ортонормованими (ортогональними, їх довжина дорівнює одиниці). Якщо на вхід мережі подається вектор  $\mathbf{p}_k$ , то вихід мережі можна обчислити за формулою

$$\mathbf{y} = \mathbf{W}\mathbf{p}_k = \left( \sum_{q=1}^Q \mathbf{t}_q \mathbf{p}_q^T \right) \mathbf{p}_k = \sum_{q=1}^Q \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k). \quad (4.7)$$

Оскільки вектори  $\mathbf{p}_q$ ,  $q = 1, \dots, Q$  – ортонормовані, то маємо:

1)  $(\mathbf{p}_q, \mathbf{p}_k) = \mathbf{p}_q^T \mathbf{p}_k = 1$ , якщо  $q = k$ ;

2)  $(\mathbf{p}_q, \mathbf{p}_k) = \mathbf{p}_q^T \mathbf{p}_k = 0$ , якщо  $q \neq k$ .

Тоді рівняння (4.7) можна переписати у вигляді  $\mathbf{y} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k$ , тобто вихід мережі дорівнює цільовому значенню.

Отже, якщо вхідні вектори-прототипи є ортонормованими, то правило Хебба дозволяє отримати правильні вихідні дані для кожного входу.

**Другий випадок.** Вектори  $\mathbf{p}_q$  не є ортонормованими. Припустимо, що вектори  $\mathbf{p}_q$  мають довжину, яка дорівнює одиниці, але вони не є ортогональними. У цьому разі рівняння (4.6) має такий вигляд:  $\mathbf{y} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \mathbf{e}_k$ , де  $\mathbf{e}_k = \sum_{q \neq k} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k)$ . Таким чином, оскільки вектори  $\mathbf{p}_q$  не є ор-

тогональними, то мережа не згенерує правильних вихідних даних, при цьому значення похибки залежатиме від сумарної кореляції між вхідними векторами-прототипами.

**Приклад 4.1.** Нехай вектори-прототипи вхід-ціль ЛА мають такий

вигляд:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 0,5 \\ -0,5 \\ 0,5 \\ -0,5 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 0,5 \\ 0,5 \\ -0,5 \\ -0,5 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$ . Необхідно виз-

начити вагову матрицю заданого ЛА. Перевіримо, чи є вхідні вектори ортонормованими. Перевірку проведемо у два етапи.

*Етап 1.* Перевіримо ортогональність:

$$(\mathbf{p}_1, \mathbf{p}_2) = [0,5 \quad -0,5 \quad 0,5 \quad -0,5] \cdot \begin{bmatrix} 0,5 \\ 0,5 \\ -0,5 \\ -0,5 \end{bmatrix} = 0.$$

Етап 2. Визначимо довжину векторів:

$$(\mathbf{p}_1, \mathbf{p}_1) = [0,5 \ 0,5 \ 0,5 \ 0,5] \cdot \begin{bmatrix} 0,5 \\ -0,5 \\ 0,5 \\ -0,5 \end{bmatrix} = 1; \quad (\mathbf{p}_2, \mathbf{p}_2) = [0,5 \ 0,5 \ -0,5 \ -0,5] \cdot \begin{bmatrix} 0,5 \\ 0,5 \\ -0,5 \\ -0,5 \end{bmatrix} = 1.$$

Вагова матриця має вигляд

$$\mathbf{W} = \mathbf{TP}^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0,5 & -0,5 & 0,5 & -0,5 \\ 0,5 & 0,5 & -0,5 & -0,5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}.$$

Якщо протестувати цю матрицю на вхідних векторах  $\mathbf{p}_1$  і  $\mathbf{p}_2$ , то отримаємо такий результат:

$$\mathbf{Wp}_1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0,5 \\ -0,5 \\ 0,5 \\ -0,5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \mathbf{t}_1; \quad \mathbf{Wp}_2 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0,5 \\ 0,5 \\ -0,5 \\ -0,5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \mathbf{t}_2.$$

Отже, вихідні дані мережі збігаються з цільовими.

**Приклад 4.2.** Визначити вагову матрицю ЛА, який може розпізнати образи та відповідні їм вектори-прототипи такого вигляду:  $\mathbf{p}_1 = [1 \ -1 \ -1]^T$  – вектор-прототип апельсина;  $\mathbf{p}_2 = [1 \ 1 \ -1]^T$  – вектор-прототип яблука. Зазначимо, що ці вхідні образи не є ортогональними. Якщо їх пронормувати (тобто поділити на  $\sqrt{3}$ ) і припустити, що цільові виходи дорівнюють відповідно «-1» і «1», то отримаємо таку множину пар вхід–ціль:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 0,5774 \\ -0,5774 \\ -0,5774 \end{bmatrix}, \mathbf{t}_1 = -1 \right\}; \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0,5774 \\ 0,5774 \\ -0,5774 \end{bmatrix}, \mathbf{t}_2 = 1 \right\}. \quad \text{У цьому разі вагова}$$

матриця має вигляд:

$$\mathbf{W} = \mathbf{TP}^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0,5774 & -0,5774 & -0,5774 \\ 0,5774 & 0,5774 & -0,5774 \end{bmatrix} = \begin{bmatrix} 0 & 1,1548 & 0 \end{bmatrix}.$$

Визначимо вхідні вектори-прототипи  $\mathbf{p}_1$  і  $\mathbf{p}_2$ :

$$\text{– для апельсина: } \mathbf{Wp}_1 = \begin{bmatrix} 0 & 1,1548 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0,5774 \\ -0,5774 \\ -0,5774 \end{bmatrix} = -0,6668;$$

$$\text{– для яблука: } \mathbf{Wp}_2 = \begin{bmatrix} 0 & 1,1548 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0,5774 \\ 0,5774 \\ -0,5774 \end{bmatrix} = 0,6668.$$

Отже, виходи ЛА майже дорівнюють цільовому значенню, але не збігаються з ним, тому можна стверджувати, що якщо множина вхідних даних, які зображують вектори-прототипи образів, не є ортогональною, правило Хебба породжує деякі похибки. Існують процедури, використання яких зменшує ці похибки. Однією з таких процедур є *псевдообернене правило*.

#### 4.4. Псевдообернене правило

Нагадаємо, що завдання ЛА полягає в генерації вихідних даних  $\mathbf{t}_q$ , які відповідають вхідним даним  $\mathbf{p}_q$ , тобто:

$$\mathbf{W}\mathbf{p}_q = \mathbf{t}_q, q = 1, \dots, Q. \quad (4.8)$$

Якщо не можна підібрати вагову матрицю так, щоб ці рівняння виконувалися точно, то бажано, щоб вони виконувалися приблизно. У цьому разі метод повинен підібрати (налаштувати) вагову матрицю таким чином, щоб мінімізувати показник якості функціонування НМ у вигляді сумарної квадратичної похибки:

$$F(\mathbf{W}) = \sum_{q=1}^Q \|\mathbf{t}_q - \mathbf{W}\mathbf{p}_q\|^2. \quad (4.9)$$

Якщо всі вхідні вектори-прототипи  $\mathbf{p}_q$  є ортонормованими і для знаходження вагової матриці  $\mathbf{W}$  було використане правило Хебба, то  $F(\mathbf{W}) = 0$ . Якщо вхідні вектори не є ортогональними і для знаходження вагової матриці  $\mathbf{W}$  використовувалося правило Хебба, то  $F(\mathbf{W}) \neq 0$ . У цьому разі незрозуміло, чи буде значення  $F(\mathbf{W})$  мінімальним. Виявляється, що вагову матрицю, яка мінімізує функціонал  $F(\mathbf{W})$ , можна отримати, використовуючи псевдообернену матрицю. Розглянемо процедуру її визначення.

Спочатку перепишемо рівняння (4.9) у матричному вигляді:

$$\mathbf{W}\mathbf{P} = \mathbf{T}, \text{ де } \mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_Q]; \mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_Q]. \quad (4.10)$$

Рівняння (4.10) можна переписати у вигляді  $F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$ , де  $\mathbf{E} = \mathbf{T} - \mathbf{W}\mathbf{P}$ . Зазначимо, що якщо  $\mathbf{W}$  – розв’язок рівняння (4.10), то  $F(\mathbf{W}) = 0$ . Якщо існує матриця, обернена до матриці  $\mathbf{P}$ , то розв’язок рівняння (4.10) має вигляд  $\mathbf{W} = \mathbf{T}\mathbf{P}^{-1}$ , проте цей випадок трапляється дуже рідко. Зазвичай вектори  $\mathbf{p}_q$  (стовпці матриці  $\mathbf{P}$ ) є незалежними, а  $R$  (розмір векторів  $\mathbf{p}_q$ ) більший за  $Q$  (кількість векторів  $\mathbf{p}_q$ ). Отже, найчастіше матриця  $\mathbf{P}$  не є квадратною, тому оберненої до неї матриці не існує.

Дослідження показали, що вагову матрицю, яка мінімізує рівняння (4.9), можна отримати, використовуючи псевдообернене правило:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+, \quad (4.11)$$

де  $\mathbf{P}^+$  – псевдообернення матриці  $\mathbf{P}$ . Псевдооберненням дійсної матриці  $\mathbf{P}$  називають унікальну матрицю  $\mathbf{P}^+$ , яка задовольняє такі умови:

$$\mathbf{P}\mathbf{P}^+\mathbf{P} = \mathbf{P}; \quad \mathbf{P}^+\mathbf{P}\mathbf{P}^+ = \mathbf{P}^+; \quad \mathbf{P}^+\mathbf{P} = (\mathbf{P}^+\mathbf{P})^T; \quad \mathbf{P}\mathbf{P}^+ = (\mathbf{P}\mathbf{P}^+)^T.$$

Якщо кількість  $R$  рядків матриці  $\mathbf{P}$  більша за кількість  $Q$  її стовпців і стовці матриці  $\mathbf{P}$  незалежні, то псевдообернену матрицю можна обчислити за формулою

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T. \quad (4.12)$$

Якщо вектори-пототиби  $\mathbf{p}_q$  ортогональні, то  $\mathbf{P}^T\mathbf{P} = \mathbf{I}$  і  $\mathbf{P}^+ = \mathbf{P}^T$ .

**Приклад 4.3.** Визначити вагову матрицю ЛА з прикл. 4.2 за псевдооберненим правилом (рівняння (4.11)). Нагадаємо, що за умовою задачі множина векторів-прототипів вхід–ціль має вигляд

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = -1 \right\}; \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = 1 \right\}.$$

Звертаємо увагу на те, що під час використання псевдооберненого правила не потрібно виконувати нормування вхідних векторів. Вагову матрицю обчислюють за формулою

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+ = [-1 \quad 1] \cdot \left( \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \right)^+, \quad \text{де псевдообернену матрицю } \mathbf{P}^+$$

визначають таким чином:

$$(\mathbf{P}^T\mathbf{P})^{-1} = \left( \begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix} \right)^{-1} = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{3}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{3}{8} \end{pmatrix},$$

$$\begin{aligned} \mathbf{P}^+ &= (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0,375 & -0,125 \\ -0,125 & 0,375 \end{bmatrix} \times \\ &\times \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0,25 & -0,5 & -0,25 \\ 0,25 & 0,5 & -0,25 \end{bmatrix}. \end{aligned}$$

Отже, вагова матриця має вигляд

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+ = [-1 \quad 1] \cdot \begin{bmatrix} 0,25 & -0,5 & -0,25 \\ 0,25 & 0,5 & -0,25 \end{bmatrix} = [0 \quad 1 \quad 0].$$

Для визначених вхідних векторів-прототипів  $\mathbf{p}_1, \mathbf{p}_2$  одержимо:

$$\mathbf{W}\mathbf{p}_1 = [0 \quad 1 \quad 0] \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = -1; \quad \mathbf{W}\mathbf{p}_2 = [0 \quad 1 \quad 0] \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = 1.$$

Отже, вихідні дані точно відповідають цільовим і псевдообернене правило дає точні результати.

## 4.5. Застосування правила Хебба на практиці

Розглянемо приклад використання правила Хебба для вирішення завдання розпізнавання образів [14]. Для збереження множини образів використаємо *автоасоціативну пам'ять*, для якої необхідне виконання такої умови: цільовий вектор дорівнює вхідному:  $\mathbf{t}_q = \mathbf{p}_q$ . Якщо на вхід мережі подаються спотворені образи, то автоасоціативна пам'ять має їх відтворити.

Нехай образи, які потрібно зберегти (рис. 4.2), мають вигляд цифр  $\{0, 1, 2\}$ , зображених у формі матриці розмірністю  $[6 \times 5]$ . Оскільки використовується автоасоціативна пам'ять, то ці образи зображують одночасно вхідні та цільові вектори. Перетворимо зображення заданих цифр у вектори-прототипи: кожен білий квадрат (піксел) буде відповідати значенню «-1», кожен темний – значенню «1». Щоб визначити вхідні вектори, скануємо по одному стовпцю кожної матриці  $[6 \times 5]$  за один раз. Наприклад, перший вектор-прототип має вигляд

$$\mathbf{p}_1 = [-1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad \dots \quad 1 \quad -1]^T.$$

Вектор  $\mathbf{p}_1$  відповідає зображенню цифри «0»,  $\mathbf{p}_2$  – цифри «1»,  $\mathbf{p}_3$  – цифри «2». Зазначимо, що розмір кожного з цих векторів дорівнює 30. Використовуючи правило Хебба, обчислимо вагову матрицю:  $\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T$  (оскільки використовується автоасоціативна пам'ять, то в рівнянні (4.6)  $\mathbf{t}_q$  замінюється на  $\mathbf{p}_q$ ). Оскільки елементи векторів-прототипів можуть набувати лише одного із двох значень «-1» або «1», модифікуємо лінійний асоціатор таким чином, щоб елементи вихідних векторів також могли набувати лише одного з цих значень. Це можна зробити в системі MatLab, замінивши лінійну ФА симетричною функцією з чітким порогом *hardlims* (отриману мережу зображено на рис. 4.3). Розглянемо функціонування цієї мережі у випадку, коли на вхід мережі подаються спотворені образи прототипів. Під час першого тестування (рис. 4.4) на вхід мережі подається образ прототипа, в якому немає нижньої частини образу. У кожному випадку мережа згенерувала

на виході правильний образ. Під час наступного тестування на вхід мережі подається образ прототипу, в якому видалено нижню частину, що становить 2/3 від усього образу (рис. 4.5). У цьому випадку тільки цифра «1» відтворюється правильно (це загальна проблема використання асоціативної пам'яті).

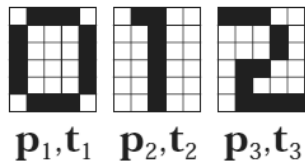


Рис. 4.2. Образи цифр {0, 1, 2}

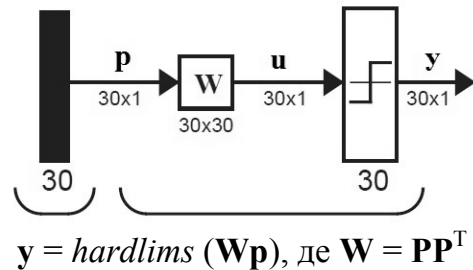


Рис. 4.3. Автоасоціативна мережа для розпізнавання цифр {0, 1, 2}

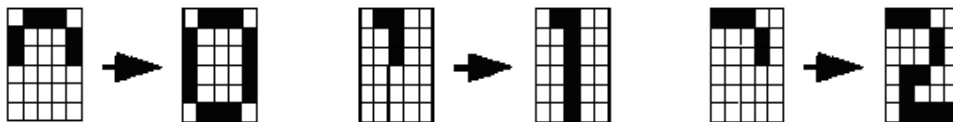


Рис. 4.4. Відтворення образів за 50 % обсягу інформації

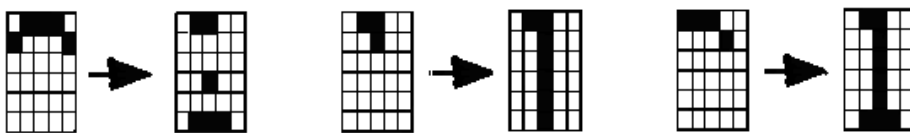


Рис. 4.5. Відтворення образів за 33 % обсягу інформації

## 4.6. Правило дельти як різновид навчання Хебба

Існують різні варіації правил навчання, основаних на правилі Хебба. Однією з проблем правила Хебба є те, що в результаті його використання за наявності в навчальній послідовності великої кількості образів у вигляді векторів-прототипів можна отримати вагову матрицю з дуже великими значеннями елементів.

Розглянемо стандартне правило Хебба:  $\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$ . Додатний коефіцієнт навчання  $\alpha < 1$  можна використати для обмеження елементів вагової матриці:  $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T$ . Щоб правило навчання поведилося подібно загладжуваному фільтру, краще запам'ятовуючи

останні вхідні дані, можна додати зменшувальний елемент  $\gamma(0 < \gamma < 1)$ :

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1 - \gamma) \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T. \quad (4.13)$$

Якщо  $\gamma = 0$ , то правило навчання стає стандартним правилом; якщо  $\gamma = 1$ , то мережа швидко забуває старі вхідні дані та пам'ятає лише останні вхідні образи (це утримує вагову матрицю від необмеженого зростання). Важливою є ідея регулювання змін вагової матриці, тому модифікуємо рівняння (4.13), замінивши цільовий вихід на різницю між цільовим і фактичним значеннями. Отримаємо інше важливе правило навчання:  $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha (\mathbf{t}_q - \mathbf{y}_q) \mathbf{p}_q^T$ , яке відоме як правило дельти, або алгоритм Відроу–Хоффа. Це правило регулює вагу таки чином, щоб мінімізувати середню квадратичну похибку, тому воно приводить до тих самих результатів, що й псевдообернене правило, яке мінімізує суму квадратів похибок (4.9).

*Перевага правила дельти* полягає в тому, що воно може налаштувати вагову матрицю після того, як кожний новий образ буде подано на вхід мережі. На відміну від цього, псевдообернене правило обчислює вагу за один крок, якщо всі пари вхід–ціль відомі. Зазначимо, що тут правило Хебба використано для контрольованого навчання, з припущенням, що відомі цільові вихідні значення мережі  $\mathbf{t}_q$ . У неконтрольованому правилі Хебба фактичний вихід мережі використовується замість цільового виходу за формулою  $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{y}_q \mathbf{p}_q^T$ , де  $\mathbf{y}_q$  – вихід мережі, який відповідає входу  $\mathbf{p}_q$ . Неконтрольоване правило Хебба не вимагає знань про цільові вихідні дані й є більш точною інтерпретацією постулату Хебба, ніж розглянуте в цьому розділі контрольоване.

### **Приклади розв'язання задач\***

**Приклад 4.4.** На рис. 4.6 зображено лінійний асоціатор. Нехай навчальні пари вхід–ціль мають такі значення:  $\{\mathbf{p}_1 = [1 \ -1 \ 1 \ -1]^T, \mathbf{t}_1 = [1 \ -1]^T\}$ ;  $\{\mathbf{p}_2 = [1 \ 1 \ -1 \ -1]^T, \mathbf{t}_2 = [1 \ 1]^T\}$ .

**I.** Використовуючи правило Хебба, визначити відповідну вагову матрицю для заданого ЛА.

**II.** Виконати п. I, використовуючи псевдообернене правило.

**III.** Протестувати отримані у п. I та II вагові матриці на вхідному векторі  $\mathbf{p}_1$ .

---

\* Задачі взято з посібника [14].

Розв'язання. **I.** На першому кроці необхідно сформувати матриці  $\mathbf{P}$  і

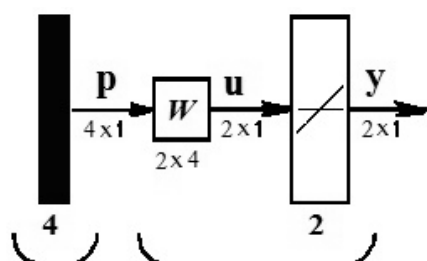


Рис. 4.6. Лінійний асоціатор:  
 $y = \text{purelin}(\mathbf{Wp})$

$\mathbf{T}$ :  $\mathbf{P} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}$ ;  $\mathbf{T} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ . За правилом Хебба вагова матриця має вигляд

$$\mathbf{W}^h = \mathbf{TP}^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \end{bmatrix}.$$

**II.** Для псевдооберненого правила використаємо рівняння  $\mathbf{W}^P = \mathbf{TP}^+$ . У зв'язку з тим, що кількість рядків матриці  $\mathbf{P}$  перевищує кількість її стовпців і стовпці матриці  $\mathbf{P}$  є незалежними, псевдообернену матрицю можна обчислити за формулою  $\mathbf{P}^+ = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T$ . Отже, маємо:

$$\begin{aligned} \mathbf{P}^+ &= \left( \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \\ &= \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix}. \end{aligned}$$

Тепер можна обчислити вагову матрицю:

$$\mathbf{W}^P = \mathbf{TP}^+ = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix}.$$

**III.** Протестуємо одержані в п. I та II вагові матриці на входному векторі  $\mathbf{p}_1$ :

$$\mathbf{W}^h \mathbf{p}_1 = \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \end{bmatrix} \neq \mathbf{t}_1;$$



$$\mathbf{W}^P \mathbf{p}_1 = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \mathbf{t}_1.$$

Отже, псевдообернене правило надає на виході правильний результат.

Щоб показати, чому правило Хебба не привело до правильного результату, розглянемо рівняння (4.7). Оскільки вектори  $\mathbf{p}_1$  і  $\mathbf{p}_2$  є ортогональними (перевірте, що це справді так), то це рівняння можна записати у вигляді  $\mathbf{y} = \mathbf{W}^h \mathbf{p}_1 = \mathbf{t}_1(\mathbf{p}_1, \mathbf{p}_1)$ . Але вектор  $\mathbf{p}_1$  не є нормованим, тому  $(\mathbf{p}_1, \mathbf{p}_1) \neq 1$ . Отже, вихід НМ не буде дорівнювати  $\mathbf{t}_1$ . Між тим, псевдообернене правило дозволяє одержати мінімальне значення виразу:  $\sum_{q=1}^2 \|\mathbf{t}_q - \mathbf{W}^P \mathbf{p}_q\|^2$ , яке в цьому разі дорівнює нулю.

**Приклад 4.5.** Деякі образи та вектори-прототипи, які їм відповідають, зображено на рис. 4.7:  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  – вектори навчання,  $\mathbf{p}_t$  – тестовий вектор.

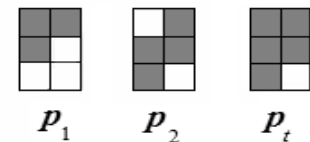


Рис. 4.7. Образи навчання та тестовий образ до прикл. 4.5

**I.** Визначити, чи є вектори-прототипи навчання  $\mathbf{p}_1$  і  $\mathbf{p}_2$  ортогональними.

**II.** Використовуючи правило Хебба, спроектувати автоасоціатор для образів, яким відповідають вектори-прототипи навчання  $\mathbf{p}_1$  і  $\mathbf{p}_2$ .

**III.** Яку відповідь надасть мережа, якщо на її вхід подати вектор-прототип  $\mathbf{p}_t$ ?

**Розв'язання.** **I.** Перетворимо задані образи на вектори-прототипи: кожний чорний квадрат (піксел) будемо відображати значенням «1», кожний білий – значенням «-1». Щоб перетворити двовимірний образ у вектор, проскануємо образ стовпець за стовпцем (за бажанням можна також сканувати рядки). У результаті одержимо два вектори-прототипи навчання:  $\mathbf{p}_1 = [1 \ 1 \ -1 \ 1 \ -1 \ -1]^T$ ;  $\mathbf{p}_2 = [-1 \ 1 \ 1 \ 1 \ 1 \ -1]^T$ . Щоб перевірити їх на ортогональність, обчислимо скалярний добуток  $(\mathbf{p}_1, \mathbf{p}_2)$ , одержимо:  $(\mathbf{p}_1, \mathbf{p}_2) = [1 \ 1 \ -1 \ 1 \ -1 \ -1] \cdot [-1 \ 1 \ 1 \ 1 \ 1 \ -1]^T = 0$ . Отже, вектори  $\mathbf{p}_1$  і  $\mathbf{p}_2$  ортогональні. Зазначимо, що ці вектори не є нормованими, оскільки  $(\mathbf{p}_1, \mathbf{p}_1) = (\mathbf{p}_2, \mathbf{p}_2) = 6$ .

**II.** Використаємо автоасоціатор, в якому кількість входів і виходів дорівнює шести. Для визначення вагової матриці використаємо правило Хебба:  $\mathbf{W} = \mathbf{T}\mathbf{P}^T$ . Враховуючи те, що для автоасоціатора виконується

умова  $\mathbf{T} = \mathbf{P}$ , маємо:  $\mathbf{P} = \mathbf{T} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$ , вагова матриця має вигляд

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix}.$$

III. Протестуємо мережу за допомогою тестового образу, перетворивши його на вектор-прототип вигляду  $\mathbf{p}_t = [1 \ 1 \ 1 \ 1 \ 1 \ -1]^T$ . Одержимо таку відповідь НМ:

$$\begin{aligned} \mathbf{y} = \text{hardlims}(\mathbf{W}\mathbf{p}_t) &= \text{hardlims} \left( \begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \right) = \\ &= \text{hardlims} \left( \begin{bmatrix} -2 \\ 6 \\ 2 \\ 6 \\ 2 \\ -6 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \mathbf{p}_2. \end{aligned}$$

Щоб перевірити одержану відповідь, необхідно визначити вектор навчання, найближчий до тестового вектора  $\mathbf{p}_t$ . У цьому розв'язку відстань Хеммінга між вхідним тестовим вектором  $\mathbf{p}_t$  та вектором-прототипом навчання  $\mathbf{p}_2$  становить  $d(\mathbf{p}_2, \mathbf{p}_t) = \frac{1}{2} \sum_{k=1}^6 |p_{tk} - p_{2k}| = 1$ , а  $d(\mathbf{p}_1, \mathbf{p}_t) = 2$ . Отже, мережа надала правильну відповідь. Зазначимо, що у цьому прикладі вектори-прототипи образів навчання не є нормованими, але це не викликало жодної проблеми у функціонуванні мережі (наприклад, проблеми, яка виникла під час розв'язання прикл. 4.2).

Це пов'язано з тим, що *hardlims* – нелінійна функція, яка на виході має лише одне з двох значень: «1» або «-1». Найбільш корисні властивості НМ є наслідком ефекту нелінійності.

**Приклад 4.6.** Деякі образи та відповідні їм вектори-прототипи зображено на рис. 4.8:  $\mathbf{p}_1$ ,  $\mathbf{p}_2$ ,  $\mathbf{p}_3$  – вектори навчання,  $\mathbf{p}_t$  – тестовий вектор.

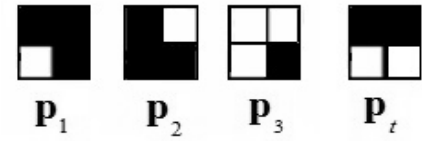


Рис. 4.8. Образи навчання та тестовий образ до прикл. 4.6

**I.** Використовуючи правило Хебба, розробити НМ типу персептрон, здатну розпізнати три задані образи навчання, яким відповідають вектори-прототипи  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  і  $\mathbf{p}_3$ .

**II.** Перевірити функціонування НМ на тестовому образі, якому відповідає вектор-прототип  $\mathbf{p}_t$  (визначити вихід мережі та дослідити його на правильність).

**Розв'язання. I.** Перетворимо задані образи на числові вектори-прототипи, одержимо:  $\mathbf{p}_1 = [1 \ -1 \ 1 \ 1]^T$ ;  $\mathbf{p}_2 = [1 \ 1 \ -1 \ 1]^T$ ;  $\mathbf{p}_3 = [-1 \ -1 \ -1 \ 1]^T$ ;  $\mathbf{p}_t = [1 \ -1 \ 1 \ -1]^T$ .

Для вхідних векторів навчання  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  і  $\mathbf{p}_3$  визначимо цільові вихідні вектори таким чином (можлива будь-яка комбінація значень «1» і «-1»):  $\mathbf{t}_1 = [-1 \ -1]^T$ ;  $\mathbf{t}_2 = [-1 \ 1]^T$ ;  $\mathbf{t}_3 = [1 \ -1]^T$ . У результаті одержимо мережу, зображену на рис. 4.9. Використовуючи правило Хебба, визначимо вагову матрицю:

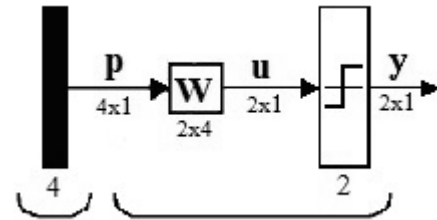


Рис. 4.9. Мережа типу персептрон:  
 $\mathbf{y} = \text{hardlims}(\mathbf{W}\mathbf{p})$

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -1 & -1 & -1 \\ 1 & 3 & -1 & -1 \end{bmatrix}.$$

**II.** Подамо на вхід мережі тестовий вектор  $\mathbf{p}_t$ , одержимо:

$$\begin{aligned} \mathbf{y} = \text{hardlims}(\mathbf{W}\mathbf{p}_t) &= \text{hardlims} \left( \begin{bmatrix} -3 & -1 & -1 & -1 \\ 1 & 3 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \right) = \\ &= \text{hardlims} \left( \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \mathbf{t}_1. \end{aligned}$$

Отже, тестовий вхідний вектор  $\mathbf{p}_t$  найбільш близький до вектора навчання  $\mathbf{p}_1$ . Цей результат є правильним, оскільки відстань Хеммінга від вектора  $\mathbf{p}_t$  до вектора  $\mathbf{p}_1$  дорівнює одиниці, тоді як до векторів  $\mathbf{p}_2$  і  $\mathbf{p}_3$  — трьома.

**Приклад 4.7.** Задано ЛА, здатний розпізнати  $Q$  ортогональних вхідних векторів-прототипів довжиною  $R$ , елементи яких набувають значень «1» або «-1», та навчений на основі правила Хебба.

**I.** Показати, що  $Q$  векторів-прототипів є власними векторами вагової матриці.

**II.** Визначити інші  $(R - Q)$  власні вектори вагової матриці.

*Розв'язання.* **I.** Припустимо, що навчання мережі відбувається на наборі векторів  $\mathbf{p}_1, \dots, \mathbf{p}_Q$ . Оскільки в автоасоціативній пам'яті задані входи є одночасно цільовими виходами мережі, то маємо  $\mathbf{T} = \mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_Q]$ . Використовуючи правило Хебба, знайдемо вагову матрицю за формулою

$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \sum_{q=1}^Q \mathbf{p}_q \mathbf{p}_q^T$ . Подамо на вхід мережі один із навчальних

векторів  $\mathbf{p}_k$ , одержимо  $\mathbf{y} = \mathbf{W}\mathbf{p}_k = \left( \sum_{q=1}^Q \mathbf{p}_q \mathbf{p}_q^T \right) \mathbf{p}_k = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q^T \mathbf{p}_k)$ . Оскільки

вектори-прототипи є ортогональними, то цей вираз має вигляд  $\mathbf{y} = \mathbf{p}_k (\mathbf{p}_k^T \mathbf{p}_k)$ . У зв'язку з тим, що елементи вхідних векторів набувають значення «1» або «-1», вихід мережі має вигляд  $\mathbf{y} = R\mathbf{p}_k$ . У загальному випадку  $\mathbf{y} = \mathbf{W}\mathbf{p}_k$ , звідки маємо  $\mathbf{W}\mathbf{p}_k = R\mathbf{p}_k$ .

Отже, вектор  $\mathbf{p}_k$  — це власний вектор, який входить до складу матриці  $\mathbf{W}$ , а  $R$  — відповідне йому власне значення. Унаслідок довільності вектора  $\mathbf{p}_k$  отримаємо, що кожний вектор-прототип є власним вектором вагової матриці  $\mathbf{W}$  із власним значенням  $R$ .

**II.** Зауважимо, що власне значення  $R$  має  $Q$ -вимірний власний простір, утворений  $Q$  ортогональними векторами-прототипами. Розглянемо підпростір, який є ортогональним до певного власного простору. Кожний вектор цього підпростору має бути ортогональним до всіх векторів-прототипів. Розмір усього ортогонального підпростору дорівнюватиме  $(R - Q)$ .

Розглянемо довільний базис цього ортогонального простору:  $\mathbf{z}_1, \dots, \mathbf{z}_{R-Q}$ . Якщо подати на вхід НМ будь-який базисний вектор  $\mathbf{z}_k$ , то, враховуючи, що він є ортогональним до кожного вектора входу  $\mathbf{p}_q$ , одержимо

$\mathbf{y} = \mathbf{W}\mathbf{z}_k = \left( \sum_{q=1}^Q \mathbf{p}_q \mathbf{p}_q^T \right) \mathbf{z}_k = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q^T \mathbf{z}_k) = \mathbf{0}$ . Це означає, що кожний вектор

$\mathbf{z}_k$  є власним вектором вагової матриці  $\mathbf{W}$  із власним значенням «0».

Наприклад, якщо довжина вхідного вектора  $R = 4$ , а кількість вхідних векторів  $Q = 3$ , то маємо 4-вимірний простір, який має два підпростори: 1) простір  $\mathcal{R}^3$ , базис якого утворений трьома векторами-прототипами  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ ; 2) простір  $\mathcal{R}^1$ , базис якого утворений вектором  $\mathbf{z}$ . У зв'язку з тим, що вихідний простір є 4-вимірним, для побудови його базису потрібні чотири ортогональні вектори. Тому базисний вектор простору  $\mathcal{R}^1$  є ортогональним до всіх базисних векторів простору  $\mathcal{R}^3$ . Отже, вагова матриця  $\mathbf{W}$  має два власні значення: « $R$ » і « $0$ ».

**Приклад 4.8.** Використовуючи псевдообернене правило, спроектувати мережу однеї нейронного персептрона зі зсувом (рис. 4.10) для розпізнавання образів, яким відповідають вектори-прототипи:  $\mathbf{p}_1 = [1 \ 1]^T$ ;  $\mathbf{p}_2 = [2 \ 2]^T$ .

Пояснити, навіщо для вирішення цієї проблеми потрібен зсув.

*Розв'язання.* Відомо, що граничним розв'язком для мережі однеї нейронного персептрона є пряма, визначена рівнянням  $\mathbf{W}\mathbf{p} + b = 0$ .

Якщо зсуву немає, тобто  $b = 0$ , ця пряма проходить через початок координат (рис. 4.11) й описується рівнянням  $\mathbf{W}\mathbf{p} = 0$ .

Розглянемо два задані вектори  $\mathbf{p}_1$  і  $\mathbf{p}_2$  (рис. 4.11). У зв'язку з тим, що жодний граничний розв'язок (пряма), який проходить через початок координат, не може розділити ці два вектори – для вирішення цієї проблеми потрібний зсув, тобто  $b \neq 0$ .

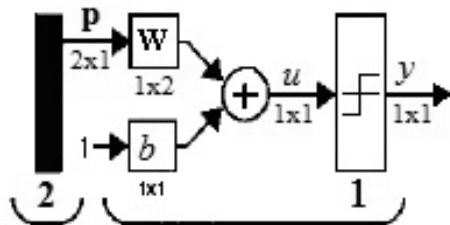


Рис. 4.10. Однеї нейронний персептрон:  $y = \text{hardlims}(\mathbf{W}\mathbf{p} + b)$

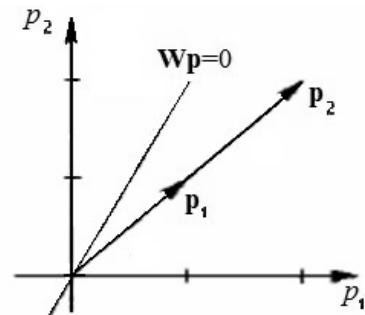


Рис. 4.11. Умова  $\mathbf{W}\mathbf{p} = 0$  до прикл. 4.8

Щоб використати псевдообернене правило (або правило Хебба) проектування мережі зі зсувом, зобразимо зсув як додатковий останній елемент вектора входу зі значенням «1», одержимо:  $\mathbf{p}'_1 = [1 \ 1 \ 1]^T$ ;  $\mathbf{p}'_2 = [2 \ 2 \ 1]^T$ . Нехай цільові виходи мережі мають вигляд  $t_1 = 1$ ,  $t_2 = -1$ .

Тоді маємо:  $\mathbf{P} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$ ;  $\mathbf{T} = [1 \ -1]$ . Сформуємо псевдообернену матрицю:

$$\mathbf{P}^+ = \left( \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} = \left( \begin{bmatrix} 3 & 5 \\ 5 & 9 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} -0,5 & -0,5 & 2 \\ 0,5 & 0,5 & -1 \end{bmatrix}.$$

Обчислимо розширену вагову матрицю:  $\mathbf{W}' = \mathbf{TP}^+ = \begin{bmatrix} 1 & -1 \end{bmatrix} \times$   
 $\times \begin{bmatrix} -0,5 & -0,5 & 2 \\ 0,5 & 0,5 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 3 \end{bmatrix}$ . Звідси можна виділити вагу та зсув:  
 $\mathbf{w} = \begin{bmatrix} -1 & -1 \end{bmatrix}^T$ ,  $b = 3$ . Отже, граничний розв'язок – це пряма  $\mathbf{w}^T \mathbf{p} + b = 0$   
(рис. 4.12), яка розділяє два задані вектори-прототипи.

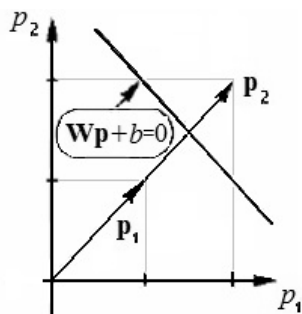


Рис. 4.12. Граничний розв'язок до прикл. 4.8

**Приклад 4.9.** У розглянутих прикладах задач розпізнавання образів образам відповідали вектори-прототипи, елементами яких були значення «1» та «-1» для зображення відповідно темних і світлих пікселів.

Розглянути, як зміниться правило Хебба, якщо замість вказаних значень використати значення «1» та «0».

**Розв'язання.** Щоб розрізняти два види зображення даних, введемо такі позначення: біполярне – у разі використання значень  $\{-1, 1\}$  і бінарне (двійкове) – для  $\{0, 1\}$ . Біполярне зображення пар вхід–ціль позначимо як  $\{\mathbf{p}_1, \mathbf{t}_1\}$ , ...,  $\{\mathbf{p}_Q, \mathbf{t}_Q\}$ , а бінарне – як  $\{\mathbf{p}'_1, \mathbf{t}'_1\}$ , ...,  $\{\mathbf{p}'_Q, \mathbf{t}'_Q\}$ . Взаємозв'язок між двома зображеннями

задається у вигляді такого рівняння:  $\mathbf{p}'_q = \left( \frac{1}{2} \mathbf{p}_q + \frac{1}{2} \mathbf{1} \right)$ ,  $\mathbf{p}_q = 2\mathbf{p}'_q - \mathbf{1}$ ,

де  $\mathbf{1}$  – вектор, який складається з одиниць. Визначимо форму бінарної асоціативної НМ, використовуючи для цього мережу, зображену на рис. 4.13. Між бінарною та біполярною асоціативними мережами існує суттєва відмінність: вектор зсуву в бінарній асоціативній НМ є необхідним, оскільки всі бінарні вектори входять в один квадрат векторного простору, а границя, що проходить через початок координат, не завжди здатна їх поділити (див. прикл. 4.8).

Визначимо вагову матрицю та вектор зсуву для цієї мережі.

Щоб бінарна НМ (рис. 4.13) мала таку ж відповідь, як і біполярна НМ (див. рис. 4.9), вхід мережі має бути однаковим для обох мереж:  $\mathbf{W}' \mathbf{p}' + \mathbf{b} = \mathbf{W} \mathbf{p}$ . Це гарантує, що кожного разу, коли біполярна мережа

буде видавати значення «1», бінарна також видасть «1», і коли біполярна мережа видасть значення «-1», бінарна видасть «0». Якщо замінити  $\mathbf{p}'$  на функцію залежності  $\mathbf{p}'$  від  $\mathbf{p}$ , то одержимо:

$$\mathbf{W}' \left( \frac{1}{2} \mathbf{p} + \frac{1}{2} \mathbf{1} \right) + \mathbf{b} =$$

$$= \frac{1}{2} \mathbf{W}' \mathbf{p} + \frac{1}{2} \mathbf{W}' \mathbf{1} + \mathbf{b} = \mathbf{W} \mathbf{p}.$$

Отже, щоб отримати такі самі результати, як у разі використання біполярної НМ, слід обрати  $\mathbf{W}' = 2\mathbf{W}$ ,  $\mathbf{b} = -\mathbf{W}\mathbf{1}$ , де  $\mathbf{W}$  – вагова матриця біполярної НМ.

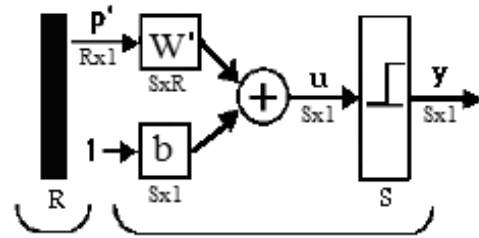


Рис. 4.13. Бінарна асоціативна НМ:  
 $y = \text{hardlim}(\mathbf{W}'\mathbf{p}' + \mathbf{b})$

### Контрольні запитання

1. Дайте визначення поняттю «асоціативна пам'ять». Які її різновиди?
2. Дайте визначення поняттю «лінійний асоціатор» (структура, особливості навчання).
3. Що таке правило Хебба та псевдообернене правило?
4. Охарактеризуйте правило дельти як різновид навчання Хебба.

### Задачі для самостійного розв'язання

**Задача 4.1.** На рис. 4.14 зображено деякі образи та вектори-прототипи, які їм відповідають:  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  – вектори навчання;  $\mathbf{p}_t$  – тестовий вектор.

**I.** Визначити, чи є вектори-прототипи навчання  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  ортогональними.

**II.** Використовуючи правило Хебба, спроектувати автоасоціатор для образів, яким відповідають вектори-прототипи навчання  $\mathbf{p}_1$  і  $\mathbf{p}_2$ .

**III.** Яку відповідь надасть мережа, якщо на її вхід подати вектор-прототип  $\mathbf{p}_t$ ?

**Задача 4.2.** Використовуючи правило Хебба для мережі типу персептрон (рис. 4.15, а), визначити вагову матрицю для розпізнавання заданих образів (рис. 4.15, б).

**Задача 4.3.** У прикл. 4.9 було показано, як мережі типу персептрон можуть навчатися на основі правила Хебба, якщо вектори-прототипи задано в двійковій та біполярній формах.



Рис. 4.14. Образи навчання та тестовий образ

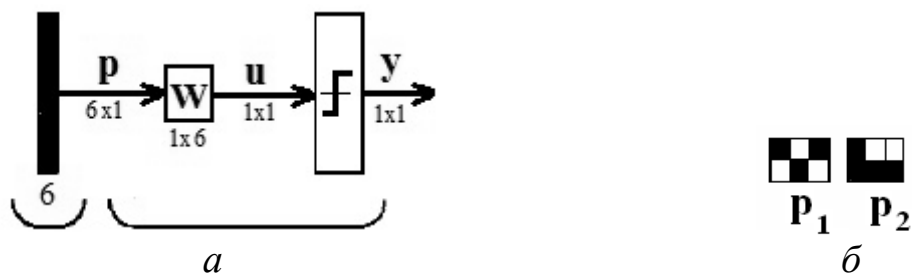


Рис. 4.15. Графічна ілюстрація до задачі 4.2:

$a$  – мережа типу персептрон:  $y = \text{hardlims}(\mathbf{W}\mathbf{p})$ ;  $b$  – образи навчання

Виконати задачу 4.1, використовуючи двійкове зображення векторів-прототипів. Показати, що відповідь мережі, яка обробляє двійкові вектори, збігається з відповіддю біполярної мережі.

**Задача 4.4.** Нехай маємо три пари навчання вхід–ціль:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{t}_1 = 1 \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_2 = -1 \right\}; \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{t}_3 = 1 \right\}.$$

Використовуючи псевдообернене правило, визначити мережу для класифікації заданих векторів-прототипів. Перевірити, чи правильно мережа відтворює вектори-прототипи. Показати, що цю задачу не можна розв'язати без використання зсуву.

**Задача 4.5.** Покажіть, що автоасоціативна мережа (див. прикл. 4.7) продовжить надавати правильний результат, якщо всі діагональні елементи вагової матриці, одержаної на основі правила Хебба, перетворити на нуль:  $\mathbf{W} = \mathbf{P}\mathbf{P}^T - Q\mathbf{I}$ , де  $Q$  – кількість ортогональних векторів-прототипів навчання мережі;  $\mathbf{I}$  – одинична матриця розмірністю  $[R \times R]$ ;  $\mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_Q]$ . При цьому ортогональні вектори-прототипи множини навчання залишаться власними векторами нової вагової матриці.



## Розділ 5. ОПТИМІЗАЦІЯ ХАРАКТЕРИСТИК ФУНКЦІОНУВАННЯ МЕРЕЖІ

### 5.1. Сутність навчання на основі корекції похибок

Визначимо, що означає поняття «*ефективність функціонування мережі*», тобто показник (кількісну міру) якості функціонування НМ. Якість функціонування НМ залежить від ефективності її навчання. Процес навчання, зазвичай, зображують у вигляді сукупності правил, які налаштовують параметри мережі таким чином, щоб оптимізувати показник ефективності функціонування НМ.

Під час навчання з учителем параметри мережі коригуються відповідно до векторів навчання і значення похибки, де похибка – це різниця між бажаним (цільовим) сигналом і поточним виходом НМ. Коригування параметрів виконується покроково з метою імітації мережею поведінки вчителя. Ця імітація в деякому статистичному сенсі має бути оптимальною. Таким чином, у процесі навчання знання вчителя передаються мережі у максимально повному обсязі. Після закінчення навчання НМ починає працювати самостійно (без учителя).

Описана форма навчання з учителем є *навчанням на основі корекції похибок*. Це замкнена система зі зворотним зв'язком, продуктивність якої можна оцінювати на навчальній вибірці в термінах середньоквадратичної похибки, зображеної у вигляді функції, яка залежить від вільних параметрів системи. Для такої функції можна побудувати багатовимірну поверхню похибки в координатах вільних параметрів, при цьому реальна поверхня похибки усереднюється на всіх можливих прикладах пар вхід–ціль. Результат будь-якої дії системи з учителем зображується однією точкою на поверхні похибок. Для підвищення продуктивності системи значення похибки на поверхні похибок має прямувати до мінімуму, який може бути як локальним, так і глобальним. Це можна зробити, якщо система має інформацію про градієнт поверхні похибок, яка відповідає поточному стану системи. Градієнт поверхні похибок у будь-якій точці – це вектор, який визначає напрямок найшвидшого спуску по цій поверхні. Під час навчання з учителем на прикладах обчислюється оцінка вектора градієнта, в якій вхідний вектор вважається функцією часу. У разі використання результатів такої оцінки переміщення точки по поверхні похибок зазвичай має вигляд «випадкового блукання». При використанні алгоритму мінімізації функції цілі, який залежить від набору прикладів навчання у вигляді пар вхід–ціль, і наявності достатнього часу для навчання мережі процедура

навчання з учителем здатна вирішувати такі завдання, як класифікація образів та апроксимація функцій. Таким чином, існує потреба розглянути алгоритми навчання, спрямовані на оптимізацію функціонування мережі, які досліджують поверхню ефективності її функціонування. Для цього слід пригадати основні поняття з *теорії оптимізації*.

Існують різні алгоритми навчання, які різняться тим, що у процесі введення даних і функціонування мережі параметри мережі (вагові коефіцієнти та зсув) налаштовуються таким чином, щоб оптимізувати показник ефективності функціонування мережі. Припустимо спочатку, що показник ефективності функціонування мережі (функція похибки) заданий. На другому кроці процесу оптимізації виконаємо пошук параметрів у просторі станів (налаштування параметрів ваги та зсуву НМ), які б збільшили її ефективність (зменшили цей показник). Дослідимо характеристики поверхні ефективності функціонування і визначимо умови, які гарантують, що поверхня справді має мінімальну точку (точку оптимуму). Далі розглянемо процедури визначення оптимальних точок.

Квадратична функція (КФ) – це єдиний універсальний вид функції, яка описує поведінку показника ефективності функціонування НМ. Крім того, багато функцій на невеликому інтервалі (особливо в околі локальних точок мінімуму) можна апроксимувати КФ, тому розглянемо їх особливості.

## 5.2. Квадратичні функції та їх особливості

Квадратична функція має такий вигляд:

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c, \quad (5.1)$$

де  $\mathbf{A}$  – симетрична матриця;  $\mathbf{d}$  – вектор;  $c$  – число. Щоб визначити градієнт цієї функції, використаємо такі його корисні властивості:

$$\nabla(\mathbf{h}^T \mathbf{x}) = \nabla(\mathbf{x}^T \mathbf{h}) = \mathbf{h},$$

де  $\mathbf{h}$  – сталий вектор,  $\nabla(\mathbf{x}^T \mathbf{Q} \mathbf{x}) = \mathbf{Q} \mathbf{x} + \mathbf{Q}^T \mathbf{x} = 2\mathbf{Q} \mathbf{x}$  для симетричної матриці  $\mathbf{Q}$ . Тоді градієнт функції  $F(\mathbf{x})$  має вигляд  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$ , а її матриця Гессе  $\nabla^2 F(\mathbf{x}) = \mathbf{A}$ . Усі похідні більш високого порядку КФ дорівнюють нулю, тому перші три елементи ряду Тейлора (рівняння (Г.3) дод. Г) дають точне зображення функції.

**Система власних векторів матриці Гессе.** Розглянемо КФ, що має стаціонарну точку, в якій її значення дорівнює нулю:  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$ .

Використаємо власні вектори  $\{\mathbf{z}_1 \dots \mathbf{z}_n\}$  матриці Гессе (матриці  $\mathbf{A}$ ) як нові базисні вектори. Оскільки  $\mathbf{A}$  симетрична, то її власні вектори

взаємно ортогональні. Це означає, що якщо сформувати матрицю  $\mathbf{B} = [\mathbf{z}_1 \dots \mathbf{z}_n]$ , стовпцями якої є нормовані власні вектори  $\mathbf{A}$ , то  $\mathbf{B}^{-1} = \mathbf{B}^T$ . Якщо тепер змінити базис так, щоб власні вектори стали базисними, то нова матриця  $\mathbf{A}'$  матиме вигляд

$$\mathbf{A}' = [\mathbf{B}^T \mathbf{A} \mathbf{B}] = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \lambda_n \end{bmatrix} = \mathbf{\Lambda},$$

де  $\lambda_i$  – власні значення матриці  $\mathbf{A}$ , звідки

$$\mathbf{A} = \mathbf{B} \mathbf{\Lambda} \mathbf{B}^T. \quad (5.2)$$

Використаємо поняття похідної за напрямком для пояснення фізичного змісту власних значень і власних векторів матриці  $\mathbf{A}$  й того, як вони визначають форму поверхні КФ.

Розглянемо другу похідну функції  $F(\mathbf{x})$  у напрямку вектора  $\mathbf{p}$ :

$$\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2}. \quad (5.3)$$

Визначимо вектор  $\mathbf{p}$  у новому базисі:  $\mathbf{p} = \mathbf{B} \mathbf{c}$ , де  $\mathbf{c}$  – зображення вектора  $\mathbf{p}$  відносно власних векторів матриці  $\mathbf{A}$ . Використовуючи рівняння (5.2), перепишемо вираз (5.3):

$$\frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{c}^T \mathbf{B}^T (\mathbf{B} \mathbf{\Lambda} \mathbf{B}^T) \mathbf{B} \mathbf{c}}{\mathbf{c}^T \mathbf{B}^T \mathbf{B} \mathbf{c}} = \frac{\mathbf{c}^T \mathbf{\Lambda} \mathbf{c}}{\mathbf{c}^T \mathbf{c}} = \frac{\sum_{i=1}^n \lambda_i c_i^2}{\sum_{i=1}^n c_i^2}. \quad (5.4)$$

Отже, друга похідна – це зважене середнє власних значень, тому вона не може бути більшою за найбільше власне значення  $\lambda_{\max}$  або меншою за найменше власне значення  $\lambda_{\min}$ , тобто  $\lambda_{\min} \leq \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} \leq \lambda_{\max}$ .

Визначимо, за якої умови друга похідна дорівнюватиме найбільшому власному значенню.

Оберемо  $\mathbf{p} = \mathbf{z}_{\max}$ , де  $\mathbf{z}_{\max}$  – власний вектор, пов'язаний із  $\lambda_{\max}$ . У цьому разі вектор  $\mathbf{c}$  набуде вигляду  $\mathbf{c} = \mathbf{B}^T \mathbf{p} = \mathbf{B}^T \mathbf{z}_{\max} = [0 \dots 0 \ 1 \ 0 \dots 0]^T$ , тобто виконується умова  $\mathbf{c}_{\max} = 1$  (оскільки власні вектори є ортогональними). Якщо замінити вектор  $\mathbf{p}$  на  $\mathbf{z}_{\max}$  у рівнянні (5.4), отримаємо

$$\frac{\mathbf{z}_{\max}^T \mathbf{A} \mathbf{z}_{\max}}{\|\mathbf{z}_{\max}\|^2} = \frac{\sum_{i=1}^n \lambda_i c_i^2}{\sum_{i=1}^n c_i^2} = \lambda_{\max}.$$

Таким чином, друга похідна є максимальною в напрямку вектора, який відповідає найбільшому власному значенню  $\lambda_{\max}$ .

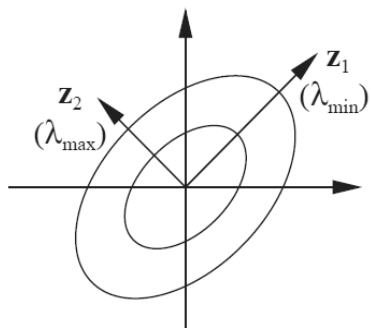


Рис. 5.1. Контурні лінії КФ

Фактично за кожним із напрямків  $\mathbf{z}_k$  другі похідні дорівнюють відповідному власному значенню  $\lambda_k$ . За іншими напрямками друга похідна буде середньозваженою величиною власних значень. Власні вектори утворюють систему координат контурів функції. На рис. 5.1 зображено випадок, коли власне значення  $\lambda_{\min} = \lambda_1$  є меншим за  $\lambda_{\max} = \lambda_2$  у декартовій системі координат, тому кривизна є мінімальною в напрямку власного вектора  $\mathbf{z}_1$ , а викривлення максимальне у напрямку власного вектора  $\mathbf{z}_2$ ; обидва власні значення мають однаковий знак (спостерігається або стійкий мінімум, або стійкий максимум), і контурні лінії мають форму еліпса.

Розглянемо приклади КФ, коли власні значення мають протилежні знаки й одне із власних значень дорівнює нулю [14].

**Приклад 5.1.** Задано КФ вигляду  $F(\mathbf{x}) = x_1^2 + x_1 x_2 + x_2^2 = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$ .

Матриця Гессе, її власні значення та власні вектори мають вигляд  $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ ;  $\lambda_1 = 1$ ;  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ ;  $\lambda_2 = 3$ ;  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . У цьому разі максимальне викривлення спостерігається у напрямку  $\mathbf{z}_2$ . Контурні лінії та 3D частину заданої функції  $F(\mathbf{x})$  зображено на рис. 5.2.

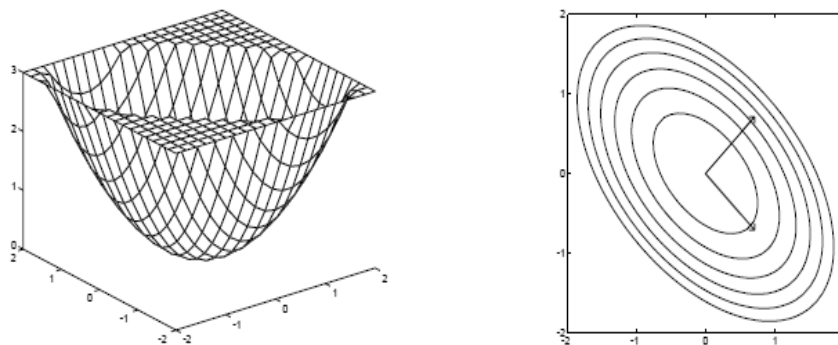


Рис. 5.2. Графік КФ та її контурні лінії (до прикл. 5.1)

**Приклад 5.2.** Розглянемо КФ вигляду  $F(\mathbf{x}) = -\frac{1}{4}x_1^2 - \frac{3}{2}x_1x_2 - \frac{1}{4}x_2^2 = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} -0,5 & -1,5 \\ -1,5 & -0,5 \end{bmatrix} \mathbf{x}$ . Матриця Гессе, її власні значення та власні вектори мають вигляд  $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} -0,5 & -1,5 \\ -1,5 & -0,5 \end{bmatrix}$ ;  $\lambda_1 = 1$ ,  $\mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ ;  $\lambda_2 = -2$ ,  $\mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ .

Значення  $\lambda_1 > 0$ , тому маємо додатне викривлення у напрямку  $\mathbf{z}_1$ ; значення  $\lambda_2 < 0$ , тому маємо від'ємне викривлення у напрямку  $\mathbf{z}_2$ .

Контурні лінії та 3D частину заданої функції  $F(\mathbf{x})$  зображено на рис. 5.3. Зазначимо, що стаціонарна точка  $\mathbf{x}^* = [0 \ 0]^T$  не є точкою сильного мінімуму, бо матриця Гессе не є додатно визначеною. Оскільки власні значення матриці Гессе мають протилежний знак, то матриця Гессе невизначена, а точка  $\mathbf{x}^*$  є сідловою: мінімумом функції  $F(\mathbf{x})$  відносно власного вектора  $\mathbf{z}_1$  і максимумом – відносно власного вектора  $\mathbf{z}_2$ .

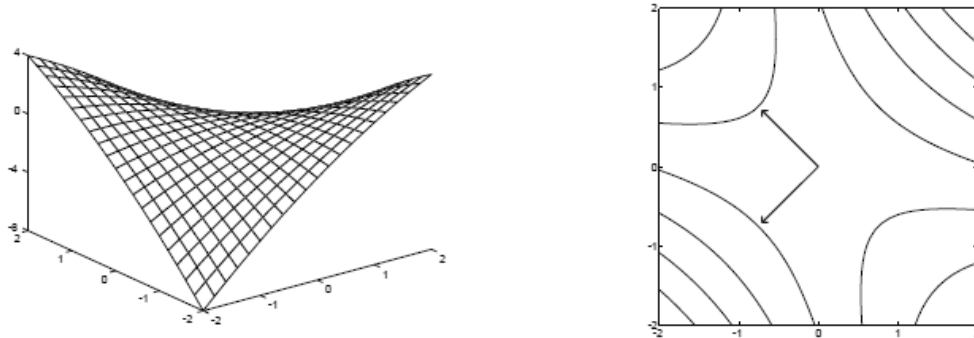


Рис. 5.3. Графік функції та її контурні лінії (до прикл. 5.2)

**Приклад 5.3.** Розглянемо КФ вигляду  $F(\mathbf{x}) = \frac{1}{2}x_1^2 - x_1x_2 + \frac{1}{2}x_2^2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{x}$ . Матриця Гессе, її власні значення та власні вектори мають вигляд  $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ ;  $\lambda_1 = 2$ ;  $\mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ ;  $\lambda_2 = 0$ ;  $\mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ .

Власне значення  $\lambda_2$  дорівнює нулю, тому виникає нульове викривлення по осі  $\mathbf{z}_2$ .

Контурні лінії та 3D частину заданої функції  $F(\mathbf{x})$  зображено на рис. 5.4. Оскільки матриця Гессе додатно напіввизначена, то спостерігаємо слабкий мінімум уздовж ліній  $x_1 = x_2$ , які відповідають власному значенню  $\lambda_2$ .

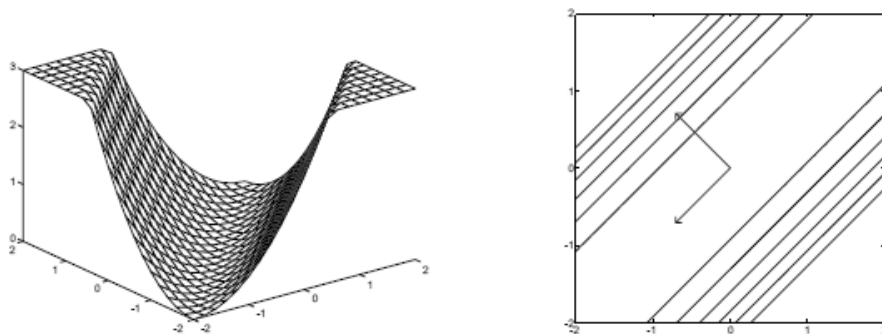


Рис. 5.4. Графік функції та її контурні лінії (до прикл. 5.3)

Визначимо властивості КФ:

1) якщо всі власні значення матриці Гессе є додатними/від'ємними, то функція має один сильний мінімум/максимум;

2) якщо деякі власні значення додатні, а інші від'ємні, то функція має одну сідлову точку;

3) якщо всі власні значення є невід'ємними (не додатні), і деякі власні значення дорівнюють нулю, то функція або має слабкий мінімум/максимум, або взагалі не має стаціонарних точок.

У випадку, коли КФ має стаціонарну точку, в якій її значення дорівнює нулю, елементи вектора  $\mathbf{d}$  і значення  $c$  у рівнянні (5.1) мають одночасно дорівнювати нулю. Якщо  $c \neq 0$ , то функція збільшується на величину  $c$  у кожній точці, але форма контурів і стаціонарна точка не змінюються.

Якщо елементи вектора  $\mathbf{d}$  не дорівнюють нулю та існує обернена до  $\mathbf{A}$  матриця, то форма контурів не змінюється, а стаціонарною точкою функції є точка  $\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d}$ . Якщо матриця  $\mathbf{A}$  не має оберненої матриці (тобто матриця  $\mathbf{A}$  має нульове власне значення) і всі елементи вектора  $\mathbf{d}$  не дорівнюють нулю, то КФ не має стаціонарних точок.

### 5.3. Визначення точок, які відповідають оптимальним значенням параметрів заданої мережі

Розкладання в ряд Тейлора – це засіб аналізу функціонала, що описує функціонування мережі, та визначення умов, яким повинні відповідати оптимальні точки цього функціонала. У дод. Г розглянуто приклади функціоналів, які описують функціонування мережі, та формулу розкладу в ряд Тейлора для визначення оптимальних точок цих функціоналів. Розглянемо три методи визначення оптимальних (мінімальних) точок, які відповідають оптимальним значенням вагових параметрів та зсуву заданої мережі (методи найшвидшого спуску, Ньютона та спряжених градієнтів).

Основні принципи оптимізації були створені ще в XVII ст. такими вченими-математиками, як Й. Кеплер, П. Ферма, І. Ньютон та Г.-В. Лейбніц. У 1950 р. їх математичні викладки (зокрема алгоритми) почали використовувати у комп'ютерних програмах. У наш час положення з теорії та практики оптимізації застосовуються до навчання НМ.

Розглянемо *алгоритми оптимізації показника ефективності функціонування НМ* у вигляді характеристичної функції  $F(\mathbf{x})$ . Під поняттям «оптимізація» розумітимемо пошук такого значення  $\mathbf{x}$ , яке мінімізує функцію  $F(\mathbf{x})$ . Усі розглядувані у цьому підрозділі алгоритми оптимізації є ітераційними. Вони починаються з деякого початкового значення (припущення)  $\mathbf{x}_0$ , яке уточнюється за формулою

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad (5.5)$$

або  $\Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k \mathbf{p}_k$ , де вектор  $\mathbf{p}_k$  є напрямком пошуку, а додатний скаляр  $\alpha_k$  – коефіцієнтом швидкості навчання, який визначає довжину кроку (рис. 5.5). Розглянемо три алгоритми оптимізації, які різняться один від одного методом вибору напрямку  $\mathbf{p}_k$ . Основою трьох алгоритмів оптимізації – методу найшвидшого спуску, методу Ньютона та методу спряжених градієнтів – є формула розкладу в ряд Тейлора.

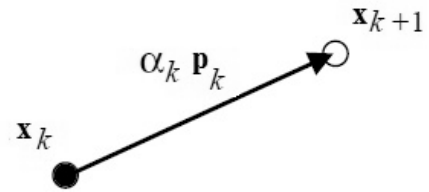


Рис. 5.5. Один ( $k$ -й) крок алгоритму оптимізації

**1. Метод найшвидшого спуску.** Припустимо, що для рівняння (5.5) на кожній ітерації виконується умова  $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$ . Покажемо, як вибрати напрямок  $\mathbf{p}_k$ , щоб за невеликої швидкості навчання  $\alpha_k$  рухатися «вниз».

Розкладемо функцію  $F(\mathbf{x})$  у ряд Тейлора першого порядку в околі попереднього значення  $\mathbf{x}_k$ , одержимо:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k, \quad (5.6)$$

де  $\mathbf{g}_k$  значення градієнта функції  $F$  у точці  $\mathbf{x}_k$ ,  $\mathbf{g}_k = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$ . Для виконання умови  $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$  необхідно, щоб  $\mathbf{g}_k^T \Delta \mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$ . Обравши додатне мале значення  $\alpha_k$ , одержимо  $\mathbf{g}_k^T \mathbf{p}_k < 0$ , де  $\mathbf{p}_k$  – напрямок спуску. Визначимо, який напрямок є напрямком найшвидшого спуску (в якому напрямку функція зменшується найшвидше). Припустимо, що довжина  $\mathbf{p}_k$  не змінюється, а змінюється тільки напрямок. Вираз  $\mathbf{g}_k^T \mathbf{p}_k$  – це скалярний добуток градієнта функції на вектор напрямку, який набуває найбільшого значення, якщо вектор напрямку протилежний до вектора градієнта. Таким чином, вектор, який вказує напрямок найшвидшого спуску, має вигляд  $\mathbf{p}_k = -\mathbf{g}_k$ . Підставивши це значення у формулу (5.5), одержимо таку формулу *методу найшвидшого спуску*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k. \quad (5.7)$$

Для цього методу є два способи визначення коефіцієнта  $\alpha_k$ : *перший спосіб* використовує фіксоване значення (наприклад,  $\alpha_k = 0,02$ ), або змінну  $\alpha_k$ , значення якої формується за визначеним правилом (наприклад,  $\alpha_k = \frac{1}{k}$ ); *другий спосіб* мінімізує функціонал  $F(\mathbf{x})$  щодо  $\alpha_k$  на кожній ітерації (по суті, це є мінімізацією вздовж прямої  $\mathbf{x}_k - \alpha_k \mathbf{g}_k$ ).

Розглянемо приклади визначення коефіцієнта  $\alpha_k$  [14].

**Приклад 5.4.** Застосуємо метод найшвидшого спуску до функції

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + [1 \ 0]\mathbf{x}, \text{ розпочавши з припу-}$$

$$\text{щення } \mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}. \text{ Визначимо градієнт, одержимо: } \nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} =$$

$$= \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix}. \text{ Значення градієнта в точці початкового припущення}$$

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}. \text{ Використаємо стале значення навчання } \alpha = \alpha_k = 0,1.$$

Виконаємо алгоритм найшвидшого спуску.

$$\text{Крок 1: } \mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0,2 \end{bmatrix}.$$

$$\text{Крок 2: } \mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0,2 \\ 0,2 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} 1,8 \\ 1,2 \end{bmatrix} = \begin{bmatrix} 0,02 \\ 0,08 \end{bmatrix}.$$

Продовживши процес, одержимо траєкторію (рис. 5.6, а). Траєкторія найшвидшого спуску за малої швидкості навчання проходить шляхом, який пролягає під прямим кутом (ортогонально) до контурних ліній, оскільки градієнт є ортогональним до контурних ліній. Якщо  $\alpha_k$  збільшити до 0,37, одержимо траєкторію, зображену на рис. 5.6, б; якщо  $\alpha_k$  збільшити до 0,39, алгоритм стає нестабільним (рис. 5.6, в). У загальному випадку неможливо визначати максимально допустиму швидкість навчання, але для квадратичних функцій можна встановити верхню межу.

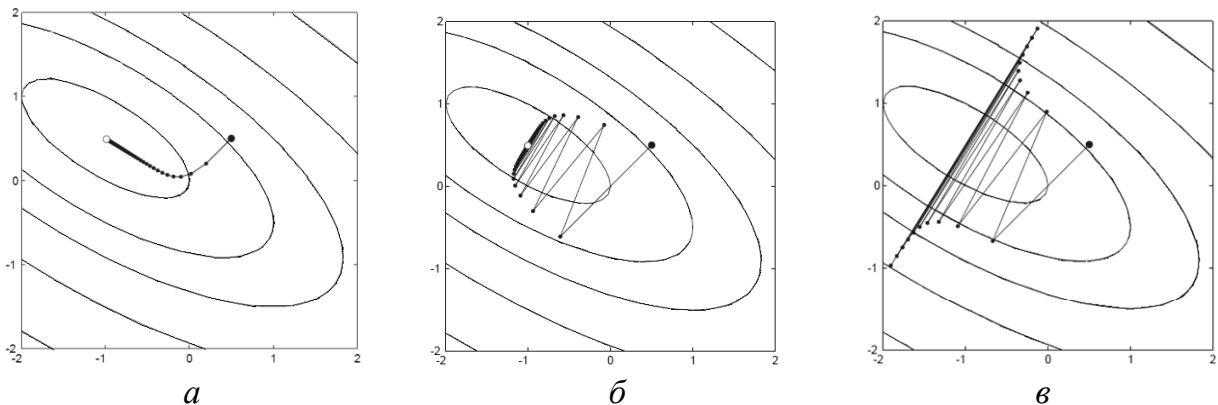


Рис. 5.6. Траєкторії найшвидшого спуску для  $\alpha = 0,1$  (а);  $\alpha = 0,37$  (б);  $\alpha = 0,39$  (в)



Існують алгоритми оптимізації, які ґрунтуються на різних способах вибору  $\alpha_k$ .

**Постійна швидкість навчання  $\alpha_k$  (мінімізація квадратичної функції).** Припустимо, що показник функціонування системи має вигляд КФ:  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$ , градієнт якої  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$ . Підставимо цей вираз у формулу (5.7) алгоритму найшвидшого спуску з постійною швидкістю навчання  $\alpha$ , одержимо:  $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k = \mathbf{x}_k - \alpha (\mathbf{A} \mathbf{x}_k + \mathbf{d})$ , звідки  $\mathbf{x}_{k+1} = [\mathbf{I} - \alpha \mathbf{A}] \mathbf{x}_k - \alpha \mathbf{d}$ . Ця рівність описує лінійну динамічну систему, яка є стабільною, якщо власні значення матриці  $[\mathbf{I} - \alpha \mathbf{A}]$  менші за одиницю. Власні значення цієї матриці можна зобразити через власні значення матриці Гессе  $\mathbf{A}$ .

Нехай  $\{\lambda_1, \dots, \lambda_n\}$  і  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$  – власні значення та власні вектори матриці Гессе. Тоді одержимо:  $[\mathbf{I} - \alpha \mathbf{A}] \mathbf{z}_i = \mathbf{z}_i - \alpha \mathbf{A} \mathbf{z}_i = \mathbf{z}_i - \alpha \lambda_i \mathbf{z}_i = (1 - \alpha \lambda_i) \mathbf{z}_i$ . Оскільки власні вектори матриці  $[\mathbf{I} - \alpha \mathbf{A}]$  збігаються з власними векторами матриці  $\mathbf{A}$ , а її власними значеннями є  $(1 - \alpha \lambda_i)$ , то умова стабільності (збіжності) алгоритму найшвидшого спуску набуде вигляду

$$|1 - \alpha \lambda_i| < 1. \quad (5.8)$$

Якщо припустити, що КФ має точку стійкого мінімуму, то її власні значення мають бути додатними. Тоді рівняння (5.8) зводиться до вигляду  $\alpha < \frac{2}{\lambda_i}$  для всіх власних значень матриці Гессе, тому маємо

$\alpha < \frac{2}{\lambda_{\max}}$ . Отже, максимально можливе значення постійної швидкості

навчання є обернено пропорційним до максимальної кривизни КФ. Кривизна показує, як швидко змінюється градієнт: якщо градієнт змінюється надто швидко, можна перескочити через точку мінімуму так, що градієнт наступної точки буде більшим (але в іншому напрямку) за градієнт у попередній точці (це призведе до збільшення розміру кроків на кожній ітерації).

**Приклад 5.5.** Застосуємо одержаний результат до прикл. 5.4. Матриця Гессе для КФ – це матриця вигляду  $\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$ . Власні значення

та власні вектори матриці  $\mathbf{A}$  мають вигляд  $\left\{ \lambda_1 = 0,764, \mathbf{z}_1 = \begin{bmatrix} 0,851 \\ -0,526 \end{bmatrix} \right\}$ ;

$\left\{ \lambda_2 = 5,24, \mathbf{z}_2 = \begin{bmatrix} 0,526 \\ 0,851 \end{bmatrix} \right\}$ . Таким чином, максимально прийнятна швид-

кість навчання має вигляд  $\alpha < \frac{2}{\lambda_{\max}} = \frac{2}{5,24} = 0,38$ . Цей результат підтвер-

джено експериментально (рис. 5.6). Наведений приклад показує, що: 1) швидкість навчання обмежена найбільшим власним значенням матриці Гессе; 2) алгоритм найшвидшого спуску збігається найшвидше в напрямку власного вектора, який відповідає найбільшому власному значенню, та найдовше в напрямку власного вектора, який відповідає найменшому власному значенню (якщо різниця між найбільшим та найменшим власним значенням є великою, то алгоритм найшвидшого спуску буде збігатися повільно).

**Мінімізація функціонала уздовж прямої.** Інший підхід до вибору коефіцієнта  $\alpha$  – мінімізація функціонала  $F(\mathbf{x})$  щодо  $\alpha_k$  на кожній ітерації, тобто  $\alpha_k$  обирається таким чином, щоб мінімізувати

$$F(\mathbf{x}_k + \alpha_k \mathbf{p}_k). \quad (5.9)$$

Щоб зробити це для довільної функції, необхідна лінія (вектор) пошуку. Для КФ лінійну мінімізацію можна виконати аналітично. Можна показати, що похідна від виразу (5.9) за  $\alpha_k$  для квадратичної функції  $F(\mathbf{x})$  має вигляд

$$\frac{d}{d\alpha_k} F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k. \quad (5.10)$$

Якщо прирівняти похідну до нуля та розв'язати рівняння, одержимо вираз для  $\alpha_k$ :

$$\alpha_k = - \frac{\nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k} = - \frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}, \quad (5.11)$$

де  $\mathbf{A}_k$  – матриця Гессе, розрахована у точці  $\mathbf{x}_k$  (для КФ матриця Гессе не залежить від  $k$ ),  $\mathbf{A}_k = \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$ .

**Приклад 5.6.** Застосуємо метод найшвидшого спуску з мінімізацією уздовж прямої до функції  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + [1 \ 0] \mathbf{x}$  із прикл. 5.4, починаючи з початкового значення  $\mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ . Градієнт заданої функції

має вигляд  $\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix}$ . Отже, маємо  $\mathbf{p}_0 = -\mathbf{g}_0 =$   
 $= -\nabla F(\mathbf{x})|_{x=x_0} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$ .

Для першої ітерації з формули (5.11) маємо таку швидкість навчання:

$$\alpha_0 = -\frac{\begin{bmatrix} 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix}}{\begin{bmatrix} -3 & -3 \end{bmatrix} \cdot \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix}} = \frac{18}{90} = 0,2. \text{ Перший крок алгоритму най-}$$

швидшого спуску має вигляд  $\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} - 0,2 \cdot \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -0,1 \\ -0,1 \end{bmatrix}$ . Результат виконання перших п'яти ітерацій алгоритму зображено на рис. 5.7.

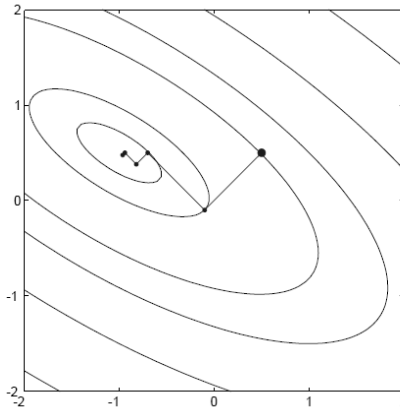


Рис. 5.7. Траєкторія найшвидшого спуску під час мінімізації уздовж прямої

Під час виконання алгоритму одержано ортогональні між собою вектори. Це сталося тому, що: 1) під час мінімізації уздовж прямої точка зупину завжди лежить у точці дотику прямої до контурної лінії; 2) оскільки градієнт завжди лежить під прямим кутом до контурної лінії, то кожний наступний вектор буде під прямим кутом до попереднього вектора.

Застосувавши до виразу (5.10) ітераційне перетворення, одержимо:

$$\begin{aligned} \frac{d}{d\alpha_k} F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &= \frac{d}{d\alpha_k} F(\mathbf{x}_{k+1}) = \nabla F(\mathbf{x})^T \Big|_{x=x_{k+1}} \frac{d}{d\alpha_k} [\mathbf{x}_k + \alpha_k \mathbf{p}_k] = \\ &= \nabla F(\mathbf{x})^T \Big|_{x=x_{k+1}} \mathbf{p}_k = \mathbf{g}_{k+1}^T \mathbf{p}_k. \end{aligned}$$

Отже, у точці мінімуму (де похідна дорівнює нулю) градієнт лежить під прямим кутом до попереднього напрямку пошуку. Оскільки наступний напрямок пошуку – це від’ємний градієнт поточного напрямку, то два послідовні напрямки пошуку є ортогональними. Зазначимо, що під час мінімізації у будь-якому напрямку, навіть якщо не використовувати метод найшвидшого спуску, градієнт у точці мінімуму буде ортогональним до напрямку пошуку. Цей висновок справедливий для методу спряжених градієнтів.

Метод найшвидшого спуску є гарантовано збіжним, якщо швидкість навчання мала, або якщо робити лінійну мінімізацію після кожного кроку. Далі покажемо, що отримані результати можна покращити, якщо обрати напрямок пошуку таким чином, щоб замість ортогональних векторів були спряжені.

**2. Метод Ньютона.** Основний принцип методу Ньютона полягає у визначенні стаціонарних точок квадратичної апроксимації  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$ . На відміну від алгоритму найшвидшого спуску, який ґрунтується на формулі розкладу в ряд Тейлора першого порядку (формула (Г.3) дод. Г), метод Ньютона використовує формулу розкладу в ряд Тейлора другого порядку:  $F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k + \frac{1}{2} \Delta \mathbf{x}_k^T \mathbf{A}_k \Delta \mathbf{x}_k$ . Прирівнявши градієнт квадратичної апроксимації  $F(\mathbf{x})$  у точці  $\mathbf{x}_k$  до нуля, одержимо  $\mathbf{g}_k + \mathbf{A}_k \Delta \mathbf{x}_k = 0$ , звідки  $\Delta \mathbf{x}_k = -\mathbf{A}_k^{-1} \mathbf{g}_k$ . Отже, метод Ньютона визначають за формулою

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k. \quad (5.12)$$

Метод Ньютона спочатку будує квадратичну апроксимацію заданої функції, а потім знаходить стаціонарні точки цієї апроксимації, тому, якщо початкова функція є квадратичною, то вона буде мінімізована за один крок. Зазначимо, що метод Ньютона не завжди є збіжним.

**Приклад 5.7.** Застосуємо метод Ньютона до функції з прикл. 5.4  $F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$ . Її градієнт і матриця Гессе мають вигляд

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix}, \quad \mathbf{A} = \nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}. \text{ Якщо розпочати алгоритм}$$

із початкового значення  $\mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ , то  $\mathbf{g}_0 = \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$  і перший

крок методу Ньютона має вигляд (рис. 5.8):  $\mathbf{x}_1 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} - \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0,5 \end{bmatrix}$ .

**Приклад 5.8.** Розглянемо функцію  $F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$ .  
 Задана функція має три стаціонарні точки:  $\mathbf{x}_1 = \begin{bmatrix} -0,42 \\ 0,42 \end{bmatrix}$ ;  $\mathbf{x}_2 = \begin{bmatrix} -0,13 \\ 0,13 \end{bmatrix}$ ;  
 $\mathbf{x}_3 = \begin{bmatrix} 0,55 \\ -0,55 \end{bmatrix}$ , де точка  $\mathbf{x}_1$  – це стійкий локальний мінімум; точка  $\mathbf{x}_2$  –  
 сідлова; а точка  $\mathbf{x}_3$  – стійкий глобальний мінімум.

На рис. 5.9 та 5.10, а зображено результати виконання першої ітерації методу Ньютона, починаючи з точок  $\mathbf{x}_0 = [1,5 \ 0]^T$  та  $\mathbf{x}_0 = [-1,5 \ 0]^T$  відповідно. Алгоритм збігається у точці локального мінімуму. Метод Ньютона не може розрізнити локальних та глобальних мінімумів. Оскільки функція апроксимації є квадратичною, вона має не більше одного мінімуму.

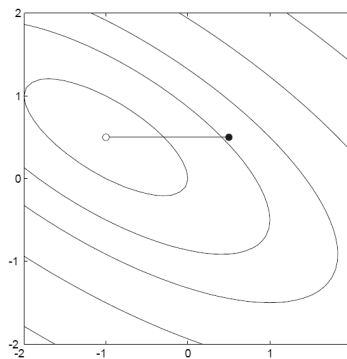


Рис. 5.8. Траєкторія методу Ньютона (до прикл. 5.7)

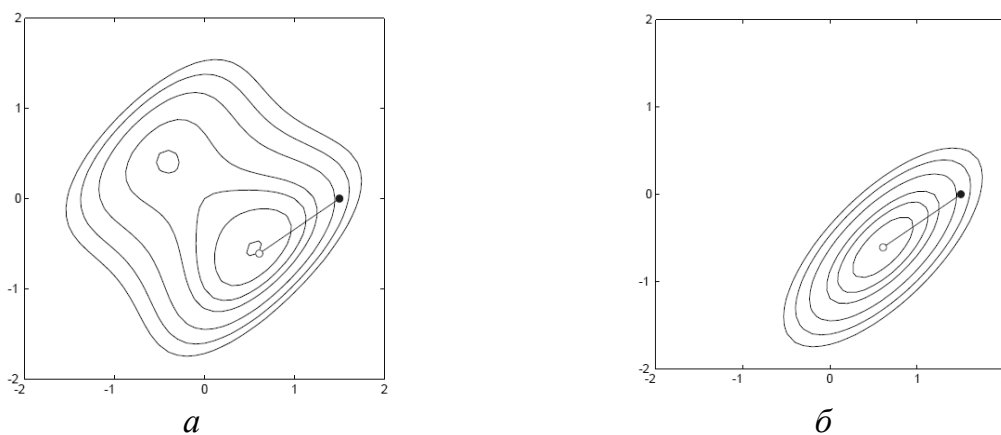


Рис. 5.9. Траєкторія методу Ньютона, одержана в результаті першої ітерації, починаючи з точки  $\mathbf{x}_0 = [1,5 \ 0]^T$  (до прикл. 5.8)

Метод Ньютона у багатьох випадках збігається швидко, оскільки аналітичні функції можна добре апроксимувати в околі сильного мінімуму. Чим ближче до точки мінімуму розміщена початкова точка, тим точніше метод Ньютона вкаже положення цього мінімуму. У цьому прикладі контурний рисунок квадратичної апроксимації збігається з контурним рисунком заданої функції біля початкової точки. На рис. 5.10, б зображено результат виконання першої ітерації методу Ньютона (а – контурні лінії функції  $F(\mathbf{x})$ ; б – квадратична апроксимація функції у початковій точці), починаючи з точки  $\mathbf{x}_0 = [0,75 \ 0,75]^T$ , якщо рухатися в напрямку сідлової точки.

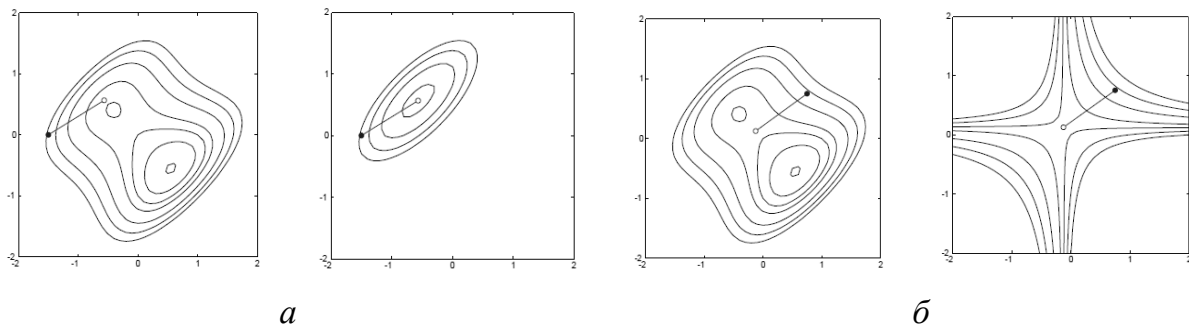


Рис. 5.10. Траєкторія методу Ньютона, одержана в результаті першої ітерації, починаючи з точки  $\mathbf{x}_0 = [-1,5 \ 0]^T$  (а) та точки  $\mathbf{x}_0 = [0,75 \ 0,75]^T$  (б)

Серед недоліків методу Ньютона можна виділити такі:

1) він визначає стаціонарні точки квадратичної апроксимації функції, починаючи із заданої початкової точки, і не розрізняє мінімумів, максимумів та сідлових точок;

2) він вимагає розрахунку, зберігання матриці Гессе та оберненої до неї матриці.

Якщо порівняти методи найшвидшого спуску (5.7) та Ньютона (5.12), то можна побачити, що їх напрямки пошуку збігаються, якщо  $\mathbf{A}_k = \mathbf{A}_k^{-1} = \mathbf{I}$ . Як результат цього спостереження виник цілий клас алгоритмів оптимізації, відомих як *квазіньютонівські* методи. У цих методах матриця  $\mathbf{A}$  замінюється додатно визначеною матрицею  $\mathbf{H}_k$ , яка оновлюється на кожній ітерації разом з оберненою матрицею. Алгоритм, розроблений таким чином, щоб матриці  $\mathbf{H}_k$  КФ збігалися до  $\mathbf{A}^{-1}$  (матриця Гессе є константою для КФ).

**3. Метод спряжених градієнтів.** Метод Ньютона має властивість, яку називають *квадратичною зупинкою*: він мінімізує КФ за скінченну кількість ітерацій. Це вимагає обчислення та зберігання похідних функції. Якщо кількість параметрів  $R$  надто велика, то обчислити всі другі похідні досить важко (нагадаємо: якщо градієнт має  $n$  елементів, то матриця Гессе –  $n^2$  елементів).

Нехай необхідно визначити мінімум КФ  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$ .

Набір векторів  $\{\mathbf{p}_k\}$  є взаємно-спряженим щодо додатно визначеної матриці Гессе  $\mathbf{A}$ , якщо:

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = 0, \quad k \neq j. \quad (5.13)$$

Як і з ортогональними векторами, існує нескінченна кількість взаємно спряжених векторів, які охоплюють заданий  $n$ -вимірний простір. Кожний набір спряжених векторів складається із власних векторів матриці  $\mathbf{A}$ . Нехай  $\{\lambda_1, \dots, \lambda_n\}$  та  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$  – власні значення та власні вектори матриці Гессе. Щоб побачити, що власні вектори є спряженими, замінимо у рівнянні (5.13)  $\mathbf{p}_k$  на  $\mathbf{z}_k$ , одержимо:  $\mathbf{z}_k^T \mathbf{A} \mathbf{z}_j = \lambda_j \mathbf{z}_k^T \mathbf{z}_j = 0, k \neq j$ , де остання рівність виконується, оскільки власні вектори симетричної матриці взаємно ортогональні. Таким чином, власні вектори одночасно спряжені й ортогональні. Не дивно, що можна мінімізувати КФ лише за допомогою пошуку власних векторів, оскільки вони утворюють основні осі для контурів функції. На жаль, для визначення власних векторів потрібно спочатку знайти матрицю Гессе. Розглянемо алгоритм, який не вимагає розрахунку других похідних.

Можна показати, що, якщо утворити послідовність ліній пошуку вздовж будь-якого набору спряжених напрямків  $\{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ , то мінімум будь-якої КФ із  $n$  параметрами буде досягнутий не більше, ніж за  $n$  кроків. Проблема полягає у визначенні таких спряжених напрямків пошуку. Для її вирішення необхідно переформулювати умови спряженості, наведені у формулі (5.13), без використання матриці Гессе. Нагадаємо, що для КФ  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$ :  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$ ;  $\nabla^2 F(\mathbf{x}) = \mathbf{A}$ . Завдяки комбінації цих виразів визначимо, що зміна градієнта на  $(k+1)$ -й ітерації має вигляд  $\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = (\mathbf{A} \mathbf{x}_{k+1} + \mathbf{d}) - (\mathbf{A} \mathbf{x}_k + \mathbf{d}) = \mathbf{A} \Delta \mathbf{x}_k$ , де за формулою (5.6) маємо  $\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$ , а  $\alpha_k$  вибрано так, щоб мінімізувати  $F(\mathbf{x})$  у напрямку  $\mathbf{p}_k$ . Тепер можна визначити нову умову спряженості:  $\alpha_k \mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{x}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{g}_k^T \mathbf{p}_j = 0, k \neq j$ , яка не вимагає визначення матриці Гессе. Зазначимо, що перший напрямок пошуку  $\mathbf{p}_0$  залежить від  $\mathbf{g}_0$ , а  $\mathbf{p}_1$  може бути будь-яким вектором, ортогональним до вектора  $\Delta \mathbf{g}_0$ . Таким чином, існує необмежений набір спряжених векторів.

Найчастіше пошук розпочинають у напрямку найшвидшого спуску:  $\mathbf{p}_0 = -\mathbf{g}_0$ . Потім на кожній ітерації треба визначити вектор  $\mathbf{p}_k$ , який буде ортогональним до  $\{\Delta \mathbf{g}_0, \dots, \Delta \mathbf{g}_{k-1}\}$ . Ця процедура аналогічна ортогона-

лізації Грамма–Шмідта, її можна описати у вигляді  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$ . Скаляри  $\beta_k$  можна визначити різними методами, які дають однакові результати для КФ. Найбільш поширені методи визначення  $\beta_k$  такі:

1) Хестенса–Штіфеля:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\Delta \mathbf{g}_{k-1}^T \mathbf{p}_{k-1}}; \quad (5.14)$$

2) Флетчера–Рівза:

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}; \quad (5.15)$$

3) Полака–Рібера:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}. \quad (5.16)$$

**Алгоритм методу спряжених градієнтів** складається з таких кроків:

*Крок 1.* Обирають напрямок пошуку, протилежний до градієнта функції:  $\mathbf{p}_0 = -\mathbf{g}_0$ .

*Крок 2.* Визначають коефіцієнт швидкості навчання  $\alpha_k$  таким чином, щоб мінімізувати функцію у напрямку пошуку. Для КФ можна використати формулу  $\alpha_k = -\frac{\nabla F(\mathbf{x})^T|_{x=x_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x})|_{x=x_k} \mathbf{p}_k} = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$ . Обчислюють

значення  $\Delta \mathbf{x}_k$ :

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k.$$

*Крок 3.* Обирають наступний напрямок пошуку за формулою  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$ , використовуючи формули (5.14), (5.15) або (5.16) для розрахунку  $\beta_k$ .

*Крок 4.* Якщо алгоритм не зійшовся, виконують *крок 2*. Квадратична функція мінімізується за  $n$  кроків.

**Приклад 5.9.** Розглянемо функцію  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + [1 \ 0] \mathbf{x}$  із початковим значенням  $\mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ . Градієнт функції має вигляд  $\nabla F(\mathbf{x}) =$

$$= \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix}. \text{ Як і в алгоритмі найшвидшого спуску,}$$



напрямок першого пошуку протилежний до градієнта:  $\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$ . За формулою (5.11) для першої ітерації (якщо

$$k = 0) \text{ швидкість навчання } \alpha_0 = -\frac{\nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k} = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k} =$$

$$= -\frac{\begin{bmatrix} 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix}}{\begin{bmatrix} -3 & -3 \end{bmatrix} \cdot \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix}} \approx 0,2.$$

Таким чином, перший крок алгоритму спряжених градієнтів має вигляд  $\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} - 0,2 \cdot \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -0,1 \\ -0,1 \end{bmatrix}$ , що еквівалентне першому кроку алгоритму найшвидшого спуску з мінімізацією вздовж напрямку вектора  $\mathbf{p}_0$ .

Визначимо другий напрямок пошуку. Для цього треба розрахувати градієнт у точці  $\mathbf{x}_1$ , маємо:  $\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -0,1 \\ -0,1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0,6 \\ -0,6 \end{bmatrix}$ .

$$\text{Знайдемо } \beta_1 \text{ за формулою (5.15), маємо: } \beta_1 = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} = \frac{\begin{bmatrix} 0,6 & -0,6 \end{bmatrix} \cdot \begin{bmatrix} 0,6 \\ -0,6 \end{bmatrix}}{\begin{bmatrix} 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 3 \end{bmatrix}} =$$

$$= \frac{0,72}{18} = 0,04. \text{ Другий напрямок обчислюємо за формулою } \mathbf{p}_1 = -\mathbf{g}_1 + \beta_1 \mathbf{p}_0 = \begin{bmatrix} -0,6 \\ 0,6 \end{bmatrix} + 0,04 \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} -0,72 \\ 0,48 \end{bmatrix}. \text{ Із формули (5.11) для другої ітерації маємо таку швидкість навчання:}$$

$$\alpha_1 = \frac{\begin{bmatrix} 0,6 & -0,6 \end{bmatrix} \cdot \begin{bmatrix} -0,72 \\ 0,48 \end{bmatrix}}{\begin{bmatrix} -0,72 & 0,48 \end{bmatrix} \cdot \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -0,72 \\ 0,48 \end{bmatrix}} = -\frac{-0,72}{0,576} = 1,25.$$

Другий крок методу спряжених градієнтів має вигляд

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = \begin{bmatrix} -0,1 \\ -0,1 \end{bmatrix} + 1,25 \cdot \begin{bmatrix} -0,72 \\ 0,48 \end{bmatrix} = \begin{bmatrix} -1 \\ 0,5 \end{bmatrix};$$

$$\mathbf{g}_2 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

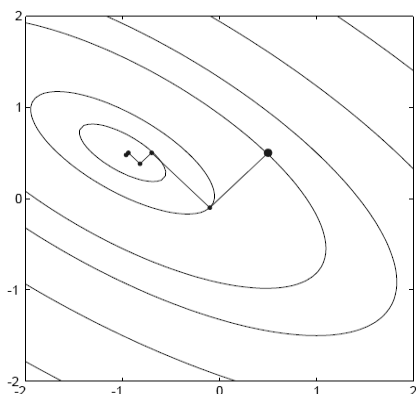


Рис. 5.11. Трасекторія виконання алгоритму спряжених градієнтів

Отже, оскільки це двовимірний КФ, то алгоритм збігається до мінімуму за дві ітерації (рис. 5.11). Алгоритм спряжених градієнтів змінює напрямок пошуку таким чином, щоб він пройшов через центр контурних кривих до мінімуму функції.

*Перевагою методу найшвидшого спуску* є простота – треба розрахувати лише градієнт, він гарантовано збігається до стаціонарної точки, якщо швидкість навчання досить мала. *Недоліком методу* є час його виконання, який залежить від швидкості навчання – він значно поступається іншим алгоритмам (особливо, якщо

власних значень матриці Гессе досить багато).

*Перевага методу Ньютона*: зазвичай він значно швидший за метод найшвидшого спуску (для КФ він знаходить стаціонарну точку за одну ітерацію). *Недолік*: вимагає розрахунку та зберігання прямої та оберненої матриць Гессе.

*Перевагами алгоритму спряжених градієнтів* є те, що він є компромісом між методами найшвидшого спуску та Ньютона. Він знаходить мінімум КФ за обмежену кількість ітерацій, не вимагає розрахунку та зберігання матриці Гессе. Його можна застосовувати до задач з великою кількістю параметрів.

### Приклади розв'язання задач\*

**Приклад 5.10.** Визначити розклад та апроксимації функції  $\cos(x)$  у ряд Тейлора в околі точки  $x^* = \pi/2$ .

*Розв'язання.* Розклад функції  $F(x) = \cos(x)$  у ряд Тейлора в околі точки  $x^* = \pi/2$  має вигляд

$$F(x) = \cos(x) = \cos\left(\frac{\pi}{2}\right) - \sin\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right) - \frac{1}{2}\cos\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right)^2 + \\ + \frac{1}{6}\sin\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right)^3 + \dots = -\left(x - \frac{\pi}{2}\right) + \frac{1}{6}\left(x - \frac{\pi}{2}\right)^3 - \frac{1}{120}\left(x - \frac{\pi}{2}\right)^5 + \dots$$

---

\* Задачі взято з посібника [14].

Апроксимації функції  $F(x)$  мають вигляд (рис. 5.12):

– нульового порядку:  $F_0(x) = 0$ ;

– першого та другого порядку:  $F_1(x) = F_2(x) = -\left(x - \frac{\pi}{2}\right) = \frac{\pi}{2} - x$ ;

– третього порядку:  $F_3(x) = -\left(x - \frac{\pi}{2}\right) + \frac{1}{6}\left(x - \frac{\pi}{2}\right)^3$ .

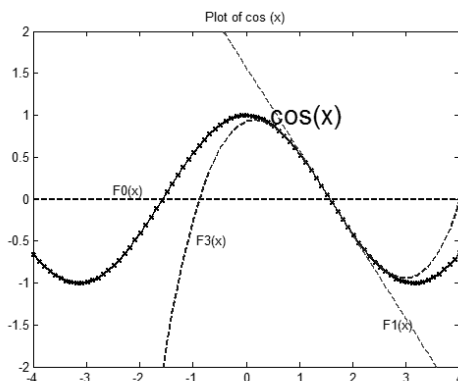


Рис. 5.12. Апроксимації функції  $\cos(x)$   
в околі точки  $x^* = \pi/2$

**Приклад 5.11.** Розглянемо функцію  $F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$ .  
Визначити розклад другого порядку цієї функції у ряд Тейлора в околі її мінімумів:  $\mathbf{x}_1 = [-0,42 \ 0,42]^T$ ;  $\mathbf{x}_2 = [0,55 \ -0,55]^T$ .

*Розв'язання.* Щоб визначити розклад другого порядку заданої функції в ряд Тейлора, необхідно визначити її градієнт і матрицю Гессе.

Градієнт функції  $F(\mathbf{x})$  має вигляд  $\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} =$

$= \begin{bmatrix} -4(x_2 - x_1)^3 + 8x_2 - 1 \\ 4(x_2 - x_1)^3 + 8x_2 + 1 \end{bmatrix}$ . Матриця Гессе функції  $F(\mathbf{x})$  має вигляд

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 12(x_2 - x_1)^2 & -12(x_2 - x_1)^2 + 8 \\ -12(x_2 - x_1)^2 + 8 & 12(x_2 - x_1)^2 \end{bmatrix}.$$

Усі власні значення матриці Гессе  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$  у точках  $\mathbf{x}_1$  і  $\mathbf{x}_2$  є додатними, тому матриця Гессе є додатно визначеною (що свідчить про

наявність сильного мінімуму). Розклад другого порядку функції  $F(\mathbf{x})$  у ряд Тейлора в околі точок  $\mathbf{x}_1$  і  $\mathbf{x}_2$  має вигляд:

– для точки  $\mathbf{x}_1$ :

$$\begin{aligned} F(\mathbf{x}_1) &= F(\mathbf{x}_1) + \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_1} (\mathbf{x} - \mathbf{x}_1) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_1)^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_1} (\mathbf{x} - \mathbf{x}_1) = \\ &= 2,93 + \frac{1}{2} \left( \mathbf{x} - \begin{bmatrix} -0,42 \\ 0,42 \end{bmatrix} \right)^T \cdot \begin{bmatrix} 8,42 & -0,42 \\ -0,42 & 8,42 \end{bmatrix} \cdot \left( \mathbf{x} - \begin{bmatrix} -0,42 \\ 0,42 \end{bmatrix} \right) = \\ &= 4,49 - [-3,7128 \quad 3,7128] \mathbf{x} + \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 8,42 & -0,42 \\ -0,42 & 8,42 \end{bmatrix} \mathbf{x}; \end{aligned}$$

$$\text{– для точки } \mathbf{x}_2: F(\mathbf{x}_2) = 7,41 - [11,781 \quad -11,781] \mathbf{x} + \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 14,71 & -6,71 \\ -6,71 & 14,71 \end{bmatrix} \mathbf{x}.$$

**Приклад 5.12.** Розглянемо функцію  $F(\mathbf{x}) = (2 + x_1)^2 + 5(1 - x_1 - x_2^2)^2$ .

Визначити рівняння прямої, яка є дотичною до контурної лінії заданої функції у точці  $\mathbf{x} = [0 \quad 0]^T$ .

*Розв'язання.* Оскільки уздовж контурної лінії функція  $F(\mathbf{x})$  не змінює свого значення, похідна функції  $F(\mathbf{x})$  уздовж контурної лінії повинна дорівнювати нулю. Тому, щоб одержати рівняння дотичної до контурної лінії, необхідно прирівняти похідну за напрямком до нуля. Нагадаємо, що рівняння похідної функції  $F(\mathbf{x})$  у напрямку вектора  $\mathbf{p}$  має вигляд  $\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$ . Тому, щоб одержати рівняння прямої, яка проходить через точ-

ку  $\mathbf{x}^* = [0 \quad 0]^T$  і вздовж якої похідна дорівнює нулю, необхідно прирівняти чисельник похідної в напрямку  $\Delta \mathbf{x}$  до нуля, тобто  $\Delta \mathbf{x}^T \nabla F(\mathbf{x}^*) = 0$ , де  $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}^*$ . Це і є рівняння дотичної.

Визначимо градієнт функції  $F(\mathbf{x})$ :

$$\begin{aligned} \nabla F(\mathbf{x}) &= \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2(2 + x_1) + 10(1 - x_1 - x_2^2)(-1) \\ 10(1 - x_1 - x_2^2)(-2x_2) \end{bmatrix} = \\ &= \begin{bmatrix} -6 + 12x_1 + 10x_2^2 \\ -20x_2 + 20x_1x_2 + 20x_2^3 \end{bmatrix}. \end{aligned}$$

Обчислимо значення градієнта в точці  $\mathbf{x}^* = [0 \quad 0]^T$ :  $\nabla F(\mathbf{x}^*) =$

$= \begin{bmatrix} -6 & 0 \end{bmatrix}^T$ . Отже, рівняння дотичної має вигляд  $\left( \mathbf{x} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)^T \begin{bmatrix} -6 \\ 0 \end{bmatrix} = 0$ , звідки одержимо  $\begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} -6 \\ 0 \end{bmatrix} = 0$  й, остаточно,  $x_1 = 0$ .

**Приклад 5.13.** Розглянемо функцію (рис. 5.13)  $F(x) = x^4 - \frac{2}{3}x^3 - 2x^2 + 2x + 4$ . Визначити всі стаціонарні точки функції  $F(x)$ , перевірити, чи вони є точками мінімуму. Визначити глобальний мінімум функції  $F(x)$ .

*Розв'язання.* Щоб визначити стаціонарні точки, прирівняємо похідну функції  $F(x)$  до нуля, одержимо:

$$\frac{d}{dx}F(x) = 4x^3 - 2x^2 - 4x + 2 = 4x(x^2 - 1) - 2(x^2 - 1) = 4(x^2 - 1) \cdot \left(x - \frac{1}{2}\right) = 0.$$

Звідси маємо точки:  $x_1 = 1$ ;  $x_2 = -1$ ;  $x_3 = 0,5$ . Щоб визначити корені багаточлена похідної функції  $F(x)$ , можна використати систему MatLab:

```
coef= [4 -2 -4 2];stapoints=roots(coef);
stapoints'
ans= 1.0000 -1.0000 0.5000
```

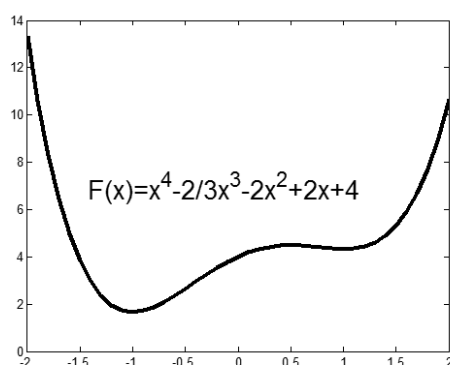


Рис. 5.13. Графік функції  $F(x)$  (до прикл. 5.13)

Перевіримо значення другої похідної у цих точках. Друга похідна функції  $F(x)$  має вигляд  $\frac{d^2}{dx^2}F(x) = 12x^2 - 4x - 4$ . Якщо обчислити її значення в

кожній стаціонарній точці, одержимо:  $\frac{d^2}{dx^2}F(1) = 4$ ;  $\frac{d^2}{dx^2}F(-1) = 12$ ;

$\frac{d^2}{dx^2}F(0,5) = -3$ . Отже, маємо сильний локальний мінімум у точках  $x_1 = 1$  і  $x_2 = -1$  (друга похідна є додатною), і сильний локальний максимум у точці  $x_3 = 0,5$  (друга похідна є від'ємною).

Щоб визначити глобальний мінімум, потрібно обчислити значення функції у двох точках локального мінімуму:  $F(1) = \frac{13}{3}$ ;  $F(-1) = \frac{5}{3}$ ; маємо  $F(1) > F(-1)$ . Отже, глобальний мінімум міститься в точці  $x_2 = -1$ .

Оскільки найвищий ступінь змінної  $x$  є парним і має додатний коефіцієнт, то функція  $F(x)$  прямує до нескінченності за  $x \rightarrow \pm\infty$ . Отже, точка  $x_2 = -1$  є точкою глобального мінімуму для всієї осі.

**Приклад 5.14.** Розглянемо функцію  $F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$ . Визначити, які із заданих стаціонарних точок є сильними мінімумами:  $\mathbf{x}_1 = [-0,42 \quad 0,42]^T$ ;  $\mathbf{x}_2 = [-0,13 \quad 0,13]^T$ ;  $\mathbf{x}_3 = [0,55 \quad -0,55]^T$ .

*Розв'язання.* Матриця Гессе для заданої функції має вигляд

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 F(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 \partial x_2} \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 F(\mathbf{x})}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 12(x_2 - x_1)^2 & -12(x_2 - x_1)^2 + 8 \\ -12(x_2 - x_1)^2 + 8 & 12(x_2 - x_1)^2 \end{bmatrix}.$$

Щоб перевірити визначеність цієї матриці, слід перевірити її власні значення:

1) якщо всі власні значення додатні, то матриця Гессе додатно визначена, що свідчить про наявність сильного мінімуму;

2) якщо власні значення невід'ємні, то матриця Гессе додатно напіввизначена, що свідчить про наявність сильного або слабкого мінімуму;

3) якщо одне власне значення є додатним, а друге – від'ємним, то матриця Гессе невизначена, що свідчить про наявність сідлової точки.

Обчислимо матрицю Гессе в точці  $\mathbf{x}_1 = [-0,42 \quad 0,42]^T$ , одержимо:

$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 8,42 & -0,42 \\ -0,42 & 8,42 \end{bmatrix} = \mathbf{A}$ . Тоді власні значення матриці  $\mathbf{A}$  можна визначити за формулою  $|\mathbf{A} - \lambda \mathbf{I}| = 0$ , звідки  $\lambda_1 = 8,84$ ;  $\lambda_2 = 8,0$ . Отже, оскільки  $\lambda_1, \lambda_2 > 0$ , то точка  $\mathbf{x}_1$  – точка сильного локального мінімуму.

Обчислимо матрицю Гессе у точці  $\mathbf{x}_2 = [-0,13 \quad 0,13]^T$ , одержимо:

$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 0,87 & 7,13 \\ 7,13 & 0,87 \end{bmatrix} = \mathbf{A}$ . Тоді власні значення матриці  $\mathbf{A}$ :  $\lambda_1 = -6,26$ ;

$\lambda_2 = 8$ . Отже, оскільки  $\lambda_1 < 0$ ,  $\lambda_2 > 0$ , то точка  $\mathbf{x}_2$  – сідлова точка: в одному напрямку викривлення функції  $F(\mathbf{x})$  додатне (у напрямку власного вектора  $\mathbf{z}_2 = [1 \quad 1]^T$ ), а в другому – від'ємне (у напрямку власного вектора  $\mathbf{z}_1 = [1 \quad -1]^T$ ).

Обчислимо матрицю Гессе у точці  $\mathbf{x}_3 = [0,55 \quad -0,55]^T$ , одержимо:  
 $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 14,7 & -6,71 \\ -6,71 & 14,7 \end{bmatrix} = \mathbf{A}$ . Тоді власні значення матриці  $\mathbf{A}$ :  $\lambda_1 = 21,42$ ;  $\lambda_2 = 8$ . Оскільки  $\lambda_1, \lambda_2 > 0$ , то точка  $\mathbf{x}_3$  – точка сильного локального мінімуму.

**Приклад 5.15.** Розглянемо лінійну мережу (рис. 5.14), що навчається на такій множині пар вхід–ціль:  $\{p_1 = 2, t_1 = 0,5\}$ ;  $\{p_2 = -1, t_2 = 0\}$ . Побудувати графік функції, яка описує ефективність функціонування заданої мережі за формулою  $F(\mathbf{x}) = (t_1 - y_1(\mathbf{x}))^2 + (t_2 - y_2(\mathbf{x}))^2$ .

*Розв’язання.* Нехай параметри мережі  $w, b$  утворюють вектор:  $\mathbf{x} = [w \quad b]^T$ . Щоб побудувати графік функції  $F(\mathbf{x})$ , яка описує показник ефективності функціонування заданої мережі, покажемо, що ця функція є квадратичною.

Визначимо власні вектори та значення матриці Гессе, нарисуємо контурні лінії функції  $F(\mathbf{x})$ .

Зобразимо функцію  $F(\mathbf{x})$ , яка описує показник ефективності функціонування заданої лінійної мережі, у вигляді функції, що залежить від вектора параметрів  $\mathbf{x}$ :  $F(\mathbf{x}) = e_1^2 + e_2^2$ , де  $e_1 = t_1 - (wp_1 + b)$ ,  $e_2 = t_2 - (wp_2 + b)$ .

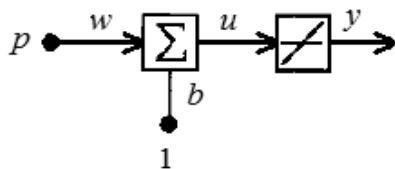


Рис. 5.14. Лінійний нейрон:  $y = \text{purelin}(wp + b)$

У матричній формі маємо:  $F(\mathbf{x}) = \mathbf{e}^T \mathbf{e}$ , де  $\mathbf{e} = \mathbf{t} - \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix} \mathbf{x} = \mathbf{t} - \mathbf{Gx}$ .

Тепер функцію  $F(\mathbf{x})$  можна записати у вигляді  $F(\mathbf{x}) = (\mathbf{t} - \mathbf{Gx})^T (\mathbf{t} - \mathbf{Gx})$ , або  $F(\mathbf{x}) = \mathbf{t}^T \mathbf{t} - 2\mathbf{t}^T \mathbf{Gx} + \mathbf{x}^T \mathbf{G}^T \mathbf{Gx}$ . Порівнявши цей вираз із загальним виглядом КФ:  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Ax} + \mathbf{d}^T \mathbf{x} + c$ , можна зробити висновок, що функція  $F(\mathbf{x})$  для заданої лінійної мережі є КФ, де  $c = \mathbf{t}^T \mathbf{t}$ ,  $\mathbf{d} = -2\mathbf{G}^T \mathbf{t}$ ,  $\mathbf{A} = 2\mathbf{G}^T \mathbf{G}$ . Градієнт КФ можна обчислити за формулою  $\nabla F(\mathbf{x}) = \mathbf{Ax} + \mathbf{d} = 2\mathbf{G}^T \mathbf{Gx} - 2\mathbf{G}^T \mathbf{t}$ .

Визначимо стаціонарну точку, яка є центром контурних ліній функції  $F(\mathbf{x})$ . Оскільки градієнт у стаціонарній точці дорівнює нулю, то її

можна визначити за формулою  $\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{t}$ , тому для

$$\mathbf{G} = \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \text{ і } \mathbf{t} = \begin{bmatrix} 0,5 \\ 0 \end{bmatrix} \text{ маємо } \mathbf{x}^* = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{t} = \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0,5 \end{bmatrix} = \begin{bmatrix} 0,167 \\ 0,167 \end{bmatrix}. \text{ Отже, оптимальними параметрами мережі є } w = 0,167, b = 0,167.$$

Матриця Гессе КФ має вигляд  $\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{G}^T\mathbf{G} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}$ . Щоб побудувати контурні лінії цієї функції, треба визначити власні значення та вектори матриці Гессе за формулою  $|\mathbf{A} - \lambda\mathbf{I}|\mathbf{z} = 0$ ,  $(10 - \lambda) \cdot (4 - \lambda) - 4 = 0$ , звідки  $\lambda^2 - 14\lambda + 36 = 0$ . Одержимо:  $\left\{ \lambda_1 = 10,6, \mathbf{z}_1 = \begin{bmatrix} 1 \\ 0,3 \end{bmatrix} \right\}$ ;  $\left\{ \lambda_2 = 34, \mathbf{z}_2 = \begin{bmatrix} 0,3 \\ -1 \end{bmatrix} \right\}$ . Отже, точка  $\mathbf{x}^*$  є точкою сильного мінімуму. Крім того, оскільки  $\lambda_2 > \lambda_1$ , то контурний графік буде мати форму еліпса, велика вісь якого спрямована в бік  $\mathbf{z}_2$ . Центром контурного графіка буде точка  $\mathbf{x}^*$ .

**Приклад 5.16.** Існують КФ, які не мають стаціонарних точок. Розглянемо приклад такої функції.

$$\text{Побудувати контурний графік функції } F(\mathbf{x}) = [1 \quad -1]\mathbf{x} + \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{x}.$$

*Розв'язання.* Для заданої функції  $F(\mathbf{x})$  матриця Гессе має вигляд  $\nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ , тому власні числа та власні вектори матриці Гессе

$$\left\{ \lambda_1 = 0, \mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}; \quad \left\{ \lambda_2 = 2, \mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}. \text{ Оскільки } \lambda_1 = 0, \text{ то кривизни}$$

уздовж власного вектора  $\mathbf{z}_1$  немає, а оскільки  $\lambda_2 > 0$ , то вздовж власного вектора  $\mathbf{z}_2$  існує додатна кривизна. Контурний графік і 3D частину заданої функції  $F(\mathbf{x})$  зображено на рис. 5.15. Оскільки одне із власних значень матриці Гессе дорівнює нулю, то немає можливості визначити:

- 1) стаціонарні точки КФ  $F(\mathbf{x})$  за формулою  $\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d}$ ;
- 2) обернену матрицю Гессе  $\mathbf{A}^{-1}$ .



Відсутність оберненої матриці означає, що наявні точки слабого мінімуму (рис. 5.15) або взагалі не існує стаціонарної точки.

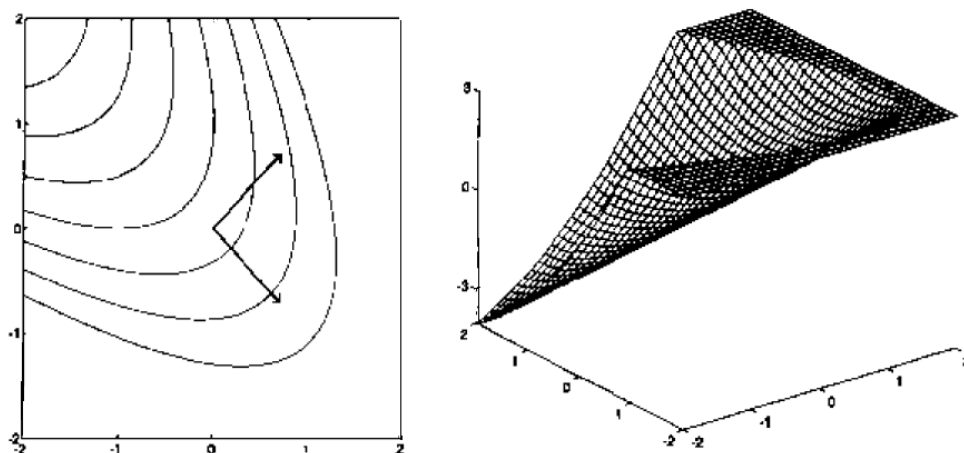


Рис. 5.15. Графіки функції  $F(\mathbf{x})$  та її контурних ліній (до прикл. 5.16)

**Приклад 5.17.** Визначити мінімум функції

$$F(\mathbf{x}) = 5x_1^2 - 6x_1x_2 + 5x_2^2 + 4x_1 + 4x_2 = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 4 & 4 \end{bmatrix} \mathbf{x}.$$

**I.** Накреслити контурний графік заданої функції.

**II.** Накреслити на контурному графіку траєкторію алгоритму найшвидшого спуску, який розпочинається з точки  $\mathbf{x}_0 = [-1 \ -2,5]^T$ . Припустити, що швидкість навчання дуже мала.

**III.** Визначити максимально можливе значення швидкості навчання.

*Розв'язання.* **I.** Визначимо матрицю Гессе. Для КФ стандартного

вигляду маємо:  $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 4 & 4 \end{bmatrix} \mathbf{x}.$

Градiєнт і матриця Гессе мають вигляд  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} +$

$+\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}.$  Визначимо власні значення та вла-

сні вектори матриці Гессе:  $|\mathbf{A} - \lambda \mathbf{I}| = \begin{vmatrix} 10 - \lambda & -6 \\ -6 & 10 - \lambda \end{vmatrix} = (10 - \lambda)^2 - 36 = 0,$

звідки  $\left\{ \lambda_1 = 4, \mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}; \left\{ \lambda_2 = 16, \mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}.$

Як відомо, контури КФ мають форму еліпса. Оскільки  $\lambda_2 > \lambda_1$ , то максимальна кривизна заданої функції  $F(\mathbf{x})$  лежить у напрямку  $\mathbf{z}_2$ ,

а мінімальна – у напрямку  $\mathbf{z}_1$  (великої осі еліпса). Визначимо центр контурних кривих (стаціонарну точку). З умови, що градієнт дорівнює нулю, одержимо:  $\nabla F(\mathbf{x}) = \mathbf{Ax} + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ , звідки

$\mathbf{x}^* = -\begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ . Отже, контурні криві (рис. 5.16) мають вигляд еліпса з центром в точці  $\mathbf{x}^*$  і великою віссю в напрямку  $\mathbf{z}_1$ .

**II.** Градієнт завжди лежить під прямим кутом до контурних ліній, тому траєкторія алгоритму найшвидшого спуску буде пролягати під прямим кутом до кожної контурної лінії, якщо вибрати досить малі значення кроків (швидкість навчання). Тому цю траєкторію можна накреслити без розрахунків (рис. 5.16).

**III.** Відомо, що максимально стійке значення швидкості навчання для КФ визначається з нерівності  $\alpha < \frac{2}{\lambda_{\max}}$ , де  $\lambda_{\max}$  – максимальне власне значення матриці Гессе, яке для цієї задачі  $\lambda_2 = 16$ . Тому умова стабільності алгоритму має вигляд  $\alpha < \frac{2}{16} = 0,125$ .

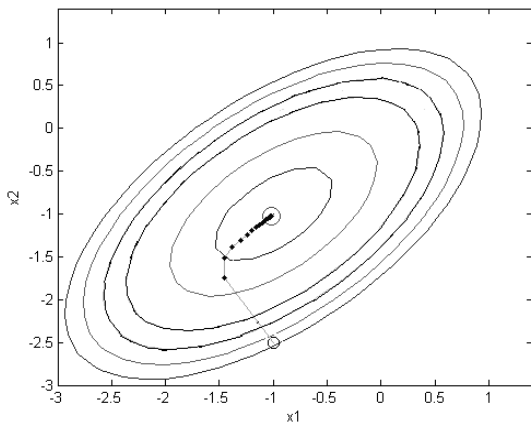


Рис. 5.16. Контурний графік функції, траєкторія алгоритму найшвидшого спуску

**Приклад 5.18.** Розглянемо КФ задачі з прикл. 5.17. Виконати два кроки алгоритму найшвидшого спуску зі змінним значенням швидкості навчання  $\alpha$ , мінімізуючи функцію на кожному кроці уздовж прямої та використовуючи початкове припущення  $\mathbf{x}_0 = [0 \ -2]^T$ .

**Розв'язання.** Градієнт цієї функції має вигляд  $\nabla F(\mathbf{x}) = \mathbf{Ax} + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 4 \\ 4 \end{bmatrix}$ . Обчисливши його

в точці  $\mathbf{x}_0$ , одержимо:

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{Ax}_0 + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -2 \end{bmatrix} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 16 \\ -16 \end{bmatrix}.$$

Отже, перший напрямок пошуку мінімуму має вигляд  $\mathbf{p}_0 = \mathbf{g}_0 = [-16 \ 16]^T$ .

Для мінімізації КФ уздовж прямої використаємо формулу для визначення коефіцієнта швидкості навчання  $\alpha_0$ :

$$\alpha_0 = -\frac{\mathbf{g}_0^T \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0} = -\frac{\begin{bmatrix} 16 & -16 \end{bmatrix} \cdot \begin{bmatrix} -16 \\ 16 \end{bmatrix}}{\begin{bmatrix} -16 & 16 \end{bmatrix} \cdot \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \cdot \begin{bmatrix} -16 \\ 16 \end{bmatrix}} = -\frac{-512}{8192} \approx 0,0625,$$

тому перший крок (ітерація) алгоритму найшвидшого спуску має вигляд

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0 \\ -2 \end{bmatrix} - 0,0625 \cdot \begin{bmatrix} 16 \\ -16 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}. \text{ На другому кроці необхідно}$$

визначити градієнт у точці  $\mathbf{x}_1$ :  $\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \mathbf{A}\mathbf{x}_1 + \mathbf{d} =$

$$= \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ точка } \mathbf{x}_1 \text{ є стаціонарною і } \mathbf{x}^* = \mathbf{x}_1. \text{ Із прикл.}$$

5.17 відомо, що точка  $\mathbf{x}_1$  – це точка мінімуму заданої квадратичної функції  $F(\mathbf{x})$ . На рис. 5.17 зображено контурний графік і відповідну траєкторію виконання алгоритму. Такий випадок є незвичним, оскільки алгоритм найшвидшого спуску за один крок одержав точку мінімуму. Це відбулося тому, що початкове припущення містилося в напрямку власного вектора матриці Гессе. Якщо напрямок руху алгоритму найшвидшого спуску збігається з напрямком власного вектора матриці Гессе, то він завжди визначить мінімум за одну ітерацію.

Розглянемо, що відбувається, якщо власні вектори замінити власними значеннями матриці Гессе (прикл. 5.19).

**Приклад 5.19.** Нехай задано лінійну мережу (див. рис. 5.14), яка навчається на множині пар вхід–ціль, вигляду  $\{p_1 = 2; t_1 = 0,5\}; \{p_2 = -1; t_2 = 0\}$ . Функціонал, який описує ефективність функціонування лінійної мережі, має вигляд

$$F(x) = (t_1 - y_1(x))^2 + (t_2 - y_2(x))^2.$$

**I.** Використовуючи алгоритм найшвидшого спуску, визначити оптимальні параметри заданої мережі  $\mathbf{x} = [w \ b]^T$  за умови, що початкове припущення  $\mathbf{x}_0 = [1 \ 1]^T$ , а швидкість навчання  $\alpha = 0,05$ .

**II.** Визначити максимальне значення швидкості навчання, за якого алгоритм збігається.

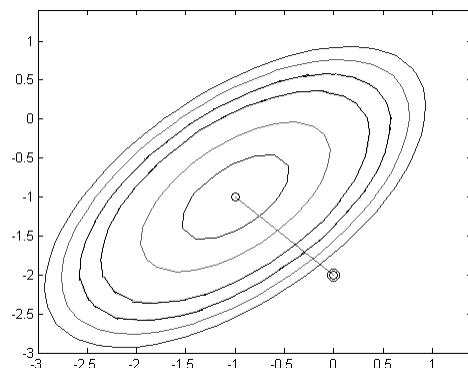


Рис. 5.17. Контурний графік функції, траєкторія алгоритму найшвидшого спуску за лінійною мінімізацією

*Розв'язання. І.* Ефективність функціонування заданої мережі можна описати такою КФ (прикл. 5.15):  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c = \mathbf{e}^T \mathbf{e}$ , де

$$\mathbf{e} = \mathbf{t} - \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix} \mathbf{x} = \mathbf{t} - \mathbf{G} \mathbf{x}. \text{ Тому маємо: } F(\mathbf{x}) = [\mathbf{t} - \mathbf{G} \mathbf{x}]^T [\mathbf{t} - \mathbf{G} \mathbf{x}] = \mathbf{t}^T \mathbf{t} - 2 \mathbf{t}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{G}^T \mathbf{G} \mathbf{x}, \text{ звідки } c = \mathbf{t}^T \mathbf{t} = \begin{bmatrix} 0,5 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0,5 \\ 0 \end{bmatrix} = 0,25; \quad \mathbf{G} = \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix};$$

$$\mathbf{d} = -2 \mathbf{G}^T \mathbf{t} = -2 \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0,5 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}; \quad \mathbf{A} = 2 \mathbf{G}^T \mathbf{G} = 2 \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}.$$

Градiєнт у точці  $\mathbf{x}_0$  має вигляд

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{A} \mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \end{bmatrix}.$$

Перша ітерація алгоритму найшвидшого спуску має вигляд

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0,05 \cdot \begin{bmatrix} 10 \\ 5 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 0,75 \end{bmatrix}.$$

Визначимо градієнт у точці  $\mathbf{x}_1$ , одержимо:

$$\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \mathbf{A} \mathbf{x}_1 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0,5 \\ 0,75 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 4,5 \\ 3 \end{bmatrix}.$$

$$\text{Друга ітерація має вигляд } \mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0,5 \\ 0,75 \end{bmatrix} - 0,05 \cdot \begin{bmatrix} 4,5 \\ 3 \end{bmatrix} = \begin{bmatrix} 0,275 \\ 0,6 \end{bmatrix}.$$

Визначимо градієнт у точці  $\mathbf{x}_2$ , одержимо:

$$\mathbf{g}_2 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_2} = \mathbf{A} \mathbf{x}_2 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0,275 \\ 0,6 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1,95 \\ 1,95 \end{bmatrix}.$$

Третя ітерація має вигляд

$$\mathbf{x}_3 = \mathbf{x}_2 - \alpha \mathbf{g}_2 = \begin{bmatrix} 0,275 \\ 0,6 \end{bmatrix} - 0,05 \cdot \begin{bmatrix} 1,95 \\ 1,95 \end{bmatrix} = \begin{bmatrix} 0,1775 \\ 0,5025 \end{bmatrix}.$$

Визначимо градієнт у точці  $\mathbf{x}_3$ , одержимо:

$$\mathbf{g}_3 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_3} = \mathbf{A} \mathbf{x}_3 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0,1775 \\ 0,5025 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 0,78 \\ 1,365 \end{bmatrix}.$$

Четверта ітерація має вигляд

$$\mathbf{x}_4 = \mathbf{x}_3 - \alpha \mathbf{g}_3 = \begin{bmatrix} 0,1775 \\ 0,5025 \end{bmatrix} - 0,05 \cdot \begin{bmatrix} 0,78 \\ 1,365 \end{bmatrix} = \begin{bmatrix} 0,1685 \\ 0,4325 \end{bmatrix} \text{ і т. д.}$$

Продовжуємо алгоритм найшвидшого спуску доти, поки він не зійдеться (рис. 5.18). Алгоритм збігається в точці мінімуму  $\mathbf{x}^* = [0,167 \ 0,167]^T$ .

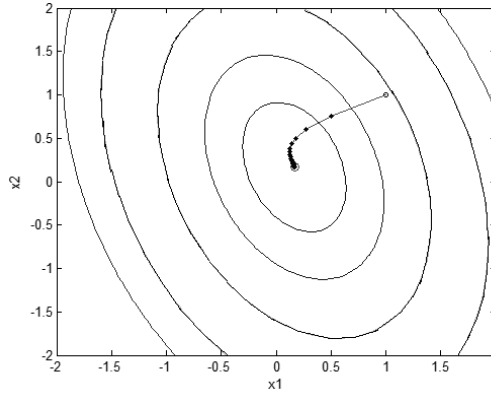


Рис. 5.18. Контурний графік функції та траєкторія алгоритму найшвидшого спуску за  $\alpha = 0,05$

Отже, оптимальні значення ваги та зсуву  $w = 0,167$ ,  $b = 0,167$ .

**II.** Щоб визначити максимальне значення швидкості навчання, треба визначити максимальне власне значення матриці Гессе.

$$\text{Маємо: } \nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{G}^T \mathbf{G} = 2 \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}; \quad \begin{vmatrix} 10-\lambda & 2 \\ 2 & 4-\lambda \end{vmatrix} = (10-\lambda) \cdot (4-\lambda) - 4 = \lambda^2 - 14\lambda + 36 = 0, \text{ звідки } \lambda_1 = 10,6; \lambda_2 = 3,4.$$

Отже, максимальне власне значення матриці Гессе – це  $\lambda_1 = 10,6$ , тому для збіжності алгоритму має виконуватись умова  $\alpha < 2/10,6 = 0,1887$ .

**Приклад 5.20.** Розглянемо функцію  $F(\mathbf{x}) = \exp(x_1^2 - x_1 + 2x_2^2 + 4)$ . Виконати одну ітерацію методу Ньютона з початковим припущенням  $\mathbf{x}_0 = [1 \ -2]^T$ . Наскільки віддалений одержаний результат від точки мінімуму функції  $F(\mathbf{x})$ ?

$$\text{Розв'язання. Визначимо градієнт } \nabla F(\mathbf{x}) = \exp(x_1^2 - x_1 + 2x_2^2 + 4) \begin{bmatrix} 2x_1 - 1 \\ 4x_2 \end{bmatrix}$$

$$\text{та матрицю Гессе } \nabla^2 F(\mathbf{x}) = \exp(x_1^2 - x_1 + 2x_2^2 + 4) \begin{bmatrix} 4x_1^2 - 4x_1 + 3 & (2x_1 - 1) \cdot 4x_2 \\ (2x_1 - 1) \cdot 4x_2 & 16x_2^2 + 4 \end{bmatrix}.$$

Обчислимо їх початкове наближення:

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = e^{12} \begin{bmatrix} 1 \\ -8 \end{bmatrix} = \begin{bmatrix} 0,163 \cdot 10^6 \\ -1,302 \cdot 10^6 \end{bmatrix};$$

$$\mathbf{A}_0 = \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 0,049 \cdot 10^7 & -0,13 \cdot 10^7 \\ -0,13 \cdot 10^7 & 1,107 \cdot 10^7 \end{bmatrix}.$$

Отже, перша ітерація методу Ньютона має вигляд

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{A}_0^{-1} \mathbf{g}_0 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} - \begin{bmatrix} 0,049 \cdot 10^7 & -0,13 \cdot 10^7 \\ -0,13 \cdot 10^7 & 1,107 \cdot 10^7 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0,163 \cdot 10^6 \\ -1,302 \cdot 10^6 \end{bmatrix} = \begin{bmatrix} 0,971 \\ -1,886 \end{bmatrix}.$$

Встановимо, наскільки віддалений одержаний результат від точки мінімуму функції  $F(\mathbf{x})$ . Зазначимо, що  $H(\mathbf{x}) = x_1^2 - x_1 + 2x_2^2 + 4$  – це КФ вигляду  $H(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{d}^T \mathbf{x} + c = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \mathbf{x} + [-1 \ 0]\mathbf{x} + 4$ . Таким чином,

мінімумом КФ  $H(\mathbf{x})$  є точка  $\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d} = -\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 0 \end{bmatrix}$ . Отже,

одна ітерація за методом Ньютона лише трохи наблизилася до точки мінімуму. Це відбулося тому, що функцію  $F(\mathbf{x})$  не можна точно апроксимувати КФ в околі точки  $\mathbf{x}_0$ . Для цієї задачі метод Ньютона зійдеться в точці реального мінімуму, але для цього необхідно виконати багато ітерацій. Траєкторію методу Ньютона зображено на рис. 5.19.

**Приклад 5.21.** Порівняти ефективність методів Ньютона та найшвидшого спуску для функції  $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{x}$  із початковим значенням у точці  $\mathbf{x}_0 = [1 \ 0]^T$ .

*Розв'язання.* Визначимо градієнт і матрицю Гессе заданої функції:  $\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{x}$ ;  $\nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ . За методом Ньютона маємо:  $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$ . Алгоритм Ньютона не можна виконати, оскільки визначник матриці Гессе дорівнює нулю і визначити  $\mathbf{A}^{-1}$  неможливо. Як відомо, така функція не має сильного мінімуму, але має слабкий мінімум уздовж прямої  $x_1 = x_2$ . Якщо розпочати алгоритм найшвидшого спуску з точки  $\mathbf{x}_0$  зі швидкістю навчання  $\alpha = 0,1$ , то перші дві ітерації набудуть вигляду

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0,9 \\ -0,1 \end{bmatrix}; \\ \mathbf{x}_2 &= \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0,9 \\ -0,1 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} 2 \\ -2 \end{bmatrix} = \begin{bmatrix} 0,8 \\ -0,2 \end{bmatrix}. \end{aligned}$$

Траєкторію алгоритму найшвидшого спуску зображено на рис. 5.20. Цей алгоритм надає кращі результати, ніж метод Ньютона: алгоритм найшвидшого спуску збігається в точці слабого мінімуму, а метод Ньютона – ні.

**Приклад 5.22.** Розглянемо функцію  $F(\mathbf{x}) = x_1^3 + x_1x_2 - x_1^2x_2^2$ .

**I.** Виконати одну ітерацію методу Ньютона, починаючи з точки  $\mathbf{x}_0 = [1 \ 1]^T$ .

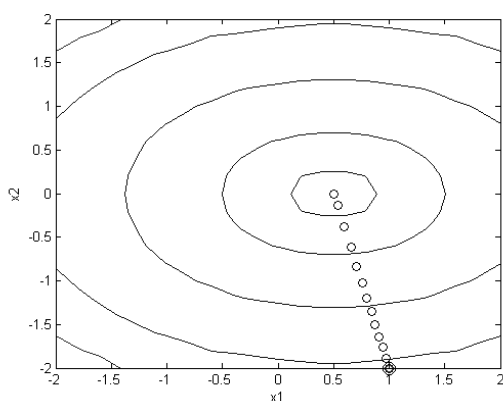


Рис. 5.19. Контурний графік функції та траєкторія методу Ньютона

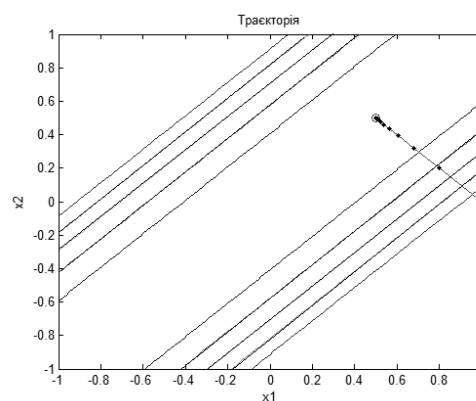


Рис. 5.20. Контурний графік функції та траєкторія алгоритму найшвидшого спуску за  $\alpha = 0,1$

**II.** Визначити розклад функції  $F(\mathbf{x})$  у ряд Тейлора другого порядку в околі точки  $\mathbf{x}_0$ . Встановити, чи має задана КФ мінімум.

*Розв'язання. I.* Градієнт функції  $F(\mathbf{x})$  та матриця Гессе

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 3x_1^2 + x_2 - 2x_1x_2^2 \\ x_1 - 2x_1^2x_2 \end{bmatrix}; \quad \nabla^2 F(\mathbf{x}) = \begin{bmatrix} 6x_1 - 2x_2^2 & 1 - 4x_1x_2 \\ 1 - 4x_1x_2 & -2x_1^2 \end{bmatrix}.$$

Обчислимо їхні значення у початковій точці  $\mathbf{x}_0$ , одержимо:

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}; \quad \mathbf{A}_0 = \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}.$$

Отже, перша ітерація методу Ньютона має вигляд

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{A}_0^{-1} \mathbf{g}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}^T \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 0,5882 \\ 1,1176 \end{bmatrix}.$$

**II.** Розклад функції  $F(\mathbf{x})$  у ряд Тейлора другого порядку в околі точки  $\mathbf{x}_0$  має вигляд  $F(\mathbf{x}) = F(\mathbf{x}_0 + \Delta \mathbf{x}) = F(\mathbf{x}_0) + \mathbf{g}_0^T \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x}^T \mathbf{A}_0 \Delta \mathbf{x}$ . Підста-

вивши значення  $\mathbf{x}_0$ ,  $\mathbf{g}_0$  та  $\mathbf{A}_0$ , одержимо:  $F(\mathbf{x}) = 1 + \begin{bmatrix} 2 & 1 \end{bmatrix} \cdot \left( \mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) + \frac{1}{2} \left( \mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^T \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}^T \left( \mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)$ , звідки  $F(\mathbf{x}) = -2 + \begin{bmatrix} 1 & 4 \end{bmatrix} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}^T \mathbf{x}$ ,

що відповідає загальному вигляду КФ. Тому функція  $F(\mathbf{x})$  має стаціонарну точку  $\mathbf{x}$ . Визначити, чи є ця точка сильним мінімумом, можна за допомогою власних значень матриці Гессе. Маємо: 1) сильний мінімум, якщо обидва власних значення додатні; 2) сильний максимум, якщо обидва власних значення від'ємні; 3) сідлову точку, якщо власні значення мають протилежні знаки.

Власні значення матриці  $\mathbf{A}_0$  мають вигляд

$$\begin{vmatrix} 4-\lambda & -3 \\ -3 & -2-\lambda \end{vmatrix} = \lambda^2 - 2\lambda - 17 = 0, \text{ звідки } \lambda_1 = 7,24; \lambda_2 = -3,24.$$

Отже, квадратична апроксимація функції  $F(\mathbf{x})$  у точці  $\mathbf{x}_0$  не має мінімуму, оскільки ця точка є сідловою.

**Приклад 5.23.** Нехай задано функцію  $F(\mathbf{x}) = 0,25 + [-2 \ -1]\mathbf{x} + \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x}$ , яка описує ефективність функціонування мережі. Визначити оптимальні параметри мережі, що мінімізують задану функцію, використовуючи алгоритм спряжених градієнтів та розпочинаючи з точки  $\mathbf{x}_0 = [1 \ 1]^T$ .

*Розв'язання.* Градієнт функції  $F(\mathbf{x})$  у точці  $\mathbf{x}_0$  має вигляд

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{A}\mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \text{ звідки } \mathbf{p}_0 = -\mathbf{g}_0 = \begin{bmatrix} -10 \\ -5 \end{bmatrix}.$$

Щоб мінімізувати КФ уздовж прямої, використаємо формулу

$$\alpha_0 = -\frac{\mathbf{g}_0^T \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0} = -\frac{\begin{bmatrix} 10 & 5 \end{bmatrix} \cdot \begin{bmatrix} -10 \\ -5 \end{bmatrix}}{\begin{bmatrix} -10 & -5 \end{bmatrix} \cdot \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -10 \\ -5 \end{bmatrix}} = -\frac{-125}{1300} \approx 0,0962. \text{ Тому перша}$$

$$\text{ітерація алгоритму спряжених градієнтів } \mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0,0962 \times \begin{bmatrix} -10 \\ -5 \end{bmatrix} = \begin{bmatrix} 0,038 \\ 0,519 \end{bmatrix}.$$

Щоб визначити напрям другого кроку, використаємо формулу  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$ . Визначимо градієнт у точці  $\mathbf{x}_1$ :

$$\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \mathbf{A}\mathbf{x}_1 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0,038 \\ 0,519 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} -0,577 \\ 1,154 \end{bmatrix}.$$

Використовуючи формулу Полака–Рібера, знайдемо  $\beta_1$ :

$$\Delta \mathbf{g}_0 = \mathbf{g}_1 - \mathbf{g}_0 = \begin{bmatrix} -0,577 \\ 1,154 \end{bmatrix} - \begin{bmatrix} 10 \\ 5 \end{bmatrix} = \begin{bmatrix} -10,577 \\ -3,846 \end{bmatrix};$$

$$\beta_1 = \frac{\Delta \mathbf{g}_0^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} = -\frac{\begin{bmatrix} -10,577 & -3,846 \end{bmatrix} \cdot \begin{bmatrix} -0,577 \\ 1,154 \end{bmatrix}}{\begin{bmatrix} 10 & 5 \end{bmatrix} \cdot \begin{bmatrix} 10 \\ 5 \end{bmatrix}} = \frac{1,665}{125} \approx 0,0133.$$



Отже, напрям наступного пошуку має такий вигляд:

$$\mathbf{p}_1 = -\mathbf{g}_1 + \beta_1 \mathbf{p}_0 = \begin{bmatrix} 0,577 \\ -1,154 \end{bmatrix} + 0,0133 \cdot \begin{bmatrix} -10 \\ -5 \end{bmatrix} = \begin{bmatrix} 0,444 \\ -1,220 \end{bmatrix}.$$

Визначимо швидкість навчання, одержимо:

$$\alpha_1 = -\frac{\mathbf{g}_1^T \mathbf{p}_1}{\mathbf{p}_1^T \mathbf{A} \mathbf{p}_1} = -\frac{\begin{bmatrix} -0,577 & -1,154 \end{bmatrix} \cdot \begin{bmatrix} 0,444 \\ -1,220 \end{bmatrix}}{\begin{bmatrix} 0,444 & -1,220 \end{bmatrix} \cdot \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0,444 \\ -1,220 \end{bmatrix}} = -\frac{-1,664}{5,758} \approx 0,2889.$$

Другий крок методу спряжених градієнтів має вигляд

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = \begin{bmatrix} 0,038 \\ 0,519 \end{bmatrix} + 0,2889 \cdot \begin{bmatrix} 0,444 \\ -1,220 \end{bmatrix} = \begin{bmatrix} 0,1667 \\ 0,1667 \end{bmatrix}.$$

Як і очікувалося, мінімум досягнуто за дві ітерації (рис. 5.21).

**Приклад 5.24.** Довести, що спряжені вектори є незалежними.

*Розв'язання.* Нехай маємо набір векторів  $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{n-1}\}$ , які спряжені відносно матриці  $\mathbf{A}$ . Якщо ці вектори залежні, то для деякого набору констант  $a_0, a_1, \dots, a_{n-1}$ , серед яких є хоча б одна ненульова, має виконуватися така рівність:  $\sum_{j=0}^{n-1} a_j \mathbf{p}_j = \mathbf{0}$ .

Якщо домножити обидві частини виразу на  $\mathbf{p}_k^T \mathbf{A}$ , отримаємо:  $\mathbf{p}_k^T \mathbf{A} \sum_{j=0}^{n-1} a_j \mathbf{p}_j = \sum_{j=0}^{n-1} a_j \mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = a_k \mathbf{p}_k^T \mathbf{A} \mathbf{p}_k = 0$  (друга рівність є наслідком визначення спряжених векторів). Якщо матриця  $\mathbf{A}$  додатно визначена, то  $\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k > 0$ , тому  $a_k$  має дорівнювати нулю для всіх  $k$ . Отже, спряжені вектори є незалежними.

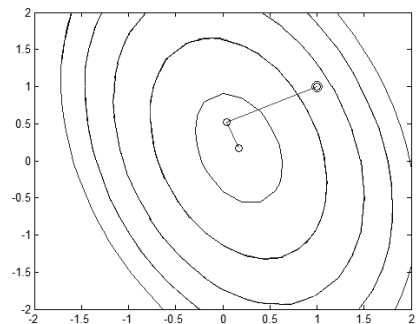


Рис. 5.21. Контурний графік функції та траєкторія алгоритму спряжених градієнтів

## Контрольні запитання

1. Опишіть сутність навчання з учителем на основі корекції похибок.
2. Визначте функціонал, який зображує ефективність функціонування лінійної мережі з одним входом.
3. Опишіть процедури виконання методу найшвидшого спуску.
4. Опишіть процедуру виконання методу Ньютона.
5. Опишіть процедуру виконання методу спряжених градієнтів.

## Задачі для самостійного розв'язання

**Задача 5.1.** Розгляньте функцію:  $F(x) = \left(x^3 - \frac{3}{4}x - \frac{1}{2}\right)^{-1}$ . Визначте апроксимацію функції  $F(x)$  у вигляді ряду Тейлора другого порядку в околі точки  $x = -0,5$ . Побудуйте графіки функції  $F(x)$  та її апроксимації.

**Задача 5.2.** Розгляньте функцію  $F(\mathbf{x}) = \frac{7}{2}x_1^2 - 6x_1x_2 - x_2^2$ . Визначте першу та другу похідні у точці  $\mathbf{x}_0 = [1 \ 1]^T$  у напрямку вектора  $\mathbf{p} = [-1 \ 1]^T$  для функції  $F(\mathbf{x})$ .

**Задача 5.3.** Розгляньте функцію  $F(x) = x^4 - \frac{1}{2}x^2 + 1$ . Визначте стаціонарні точки функції  $F(x)$ . Перевірте їх на мінімум і максимум. Побудуйте графік функції  $F(x)$  у середовищі системи MatLab.

**Задача 5.4.** Розгляньте функцію  $F(\mathbf{x}) = (x_1 + x_2)^4 - 12x_1x_2 + x_1 + x_2 + 1$ . **I.** Перевірте, чи є задані точки  $\mathbf{x}_1 = [-0,6504 \ -0,6504]^T$ ;  $\mathbf{x}_2 = [0,085 \ 0,085]^T$ ;  $\mathbf{x}_3 = [0,5655 \ 0,5655]^T$  стаціонарними точками функції  $F(\mathbf{x})$ . **II.** Визначте, яка з них є мінімумом, максимумом і сідловою. **III.** Визначте апроксимацію функції  $F(\mathbf{x})$  у вигляді ряду Тейлора другого порядку в околі стаціонарних точок. **IV.** Побудуйте графік функції  $F(\mathbf{x})$  та її апроксимацію у середовищі системи MatLab.

**Задача 5.5.** Для заданих функцій  $F(\mathbf{x}) = \frac{7}{2}x_1^2 - 6x_1x_2 - x_2^2$ ;  $F(\mathbf{x}) = 5x_1^2 - 6x_1x_2 + 5x_2^2 + 4x_1 + 4x_2$ ;  $F(\mathbf{x}) = \frac{9}{2}x_1^2 - 2x_1x_2 + 3x_2^2 + 2x_1 - x_2$ ;  $F(\mathbf{x}) = -\frac{1}{2}(7x_1^2 + 12x_1x_2 - 2x_2^2)$  визначте стаціонарні точки, перевірте, яка з них є мінімумом, максимумом або сідловою. Використовуючи власні вектори та власні значення матриці Гессе, побудуйте контурні лінії  $F(\mathbf{x})$ . Побудуйте графіки функцій  $F(\mathbf{x})$  у середовищі системи MatLab.

**Задача 5.6.** Визначте мінімум функції  $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix} \mathbf{x} + [-1 \ -1]\mathbf{x}$ . **I.** Накресліть контурний графік заданої функції, відобразіть на ньому траєкторію двох ітерацій алгоритму найшвидшого спуску, якщо швидкість навчання дуже мала ( $\alpha = 0,1$ ), а початкове припущення має вигляд точки  $\mathbf{x}_0 = [0 \ 0]^T$ . **II.** Визначте максимально можливе значення швидкості навчання. **III.** Напишіть програму (у вигляді *m*-файлу в середовищі системи MatLab) алгоритму найшвидшого спуску для цієї задачі. Перевірте з її допомогою відповіді до пунктів I–II.

**Задача 5.7.** Нехай задано КФ вигляду  $F(\mathbf{x}) = x_1^2 + 2x_2^2$ . **I.** Визначте мінімум функції уздовж прямої  $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \alpha \begin{bmatrix} -1 \\ -2 \end{bmatrix}$ . **II.** Перевірте, чи є градієнт функції  $F(\mathbf{x})$  у точці мінімуму (з п. I) ортогональним до прямої, уздовж якої відбувалася мінімізація.

**Задача 5.8.** Нехай задано функції  $F(\mathbf{x}) = 5x_1^2 - 6x_1x_2 + 5x_2^2 + 4x_1 + 4x_2$ ;  $F(\mathbf{x}) = \frac{7}{2}x_1^2 - 6x_1x_2 - x_2^2$ ;  $F(\mathbf{x}) = \frac{9}{2}x_1^2 - 2x_1x_2 + 3x_2^2 + 2x_1 - x_2$ ;  $F(\mathbf{x}) = -\frac{1}{2}(7x_1^2 + 12x_1x_2 - 2x_2^2)$ . Виконайте дві ітерації методу найшвидшого спуску з лінійною мінімізацією, починаючи з точки  $\mathbf{x}_0 = [1 \quad 1]^T$ . Перевірте виконання алгоритму в середовищі системи MatLab.

**Задача 5.9.** Розгляньте функцію  $F(\mathbf{x}) = (x_1 + x_2)^4 - 12x_1x_2 + x_1 + x_2 + 1$ . Визначте мінімум заданої функції методами найшвидшого спуску та Ньютона, починаючи з точки  $\mathbf{x}_0 = [10 \quad 10]^T$  у середовищі системи MatLab. Перевірте виконання алгоритму, змінюючи початкові точки.

**Задача 5.10.** Повторіть задачу 5.6 для алгоритму спряжених градієнтів, при цьому хоча б один раз використайте кожний із трьох методів визначення коефіцієнта  $\beta$  (формули (5.14)–(5.16)) алгоритму спряжених градієнтів.

**Задача 5.11.** Розгляньте функцію  $F(\mathbf{x}) = [1 + (x_1 + x_2 - 5)^2][1 + (3x_1 - 2x_2)^2]$ . **I.** Виконайте одну ітерацію методу Ньютона, починаючи з точки  $\mathbf{x}_0 = [10 \quad 10]^T$ . **II.** Повторіть п. I. починаючи з точки  $\mathbf{x}_0 = [2 \quad 2]^T$ . **III.** Визначте мінімум заданої функції і порівняйте його із результатами, одержаними в п. I, II.

## Розділ 6. ЛІНІЙНА ОДНОШАРОВА МЕРЕЖА

### 6.1. Однонеуронна мережа АДАЛІН

У 1960 р. Б. Відроу разом зі своїм студентом М. Хоффом запропонував мережу АДАЛІН (АДАптивний ЛІнійний Нейрон) та правило навчання, яке вони назвали алгоритмом LMS (Least Mean Square – найменше середньоквадратичне). Процедура навчання Відроу–Хоффа є наближеним алгоритмом найшвидшого спуску, в якому досліджується середньоквадратична похибка. Цей алгоритм передував алгоритму зворотного поширення похибки для багатошарових мереж. Обидві мережі – АДАЛІН та персептрон – мають однакові обмеження: вони можуть розв’язувати лише задачі, в яких дані можна розділити лінійно; однак алгоритм LMS більш потужний, ніж правило навчання персептрона. Оскільки вхідні дані часто наближені до граничного розв’язку, то мережа АДАЛІН чутлива до шумів. Найчастіше алгоритм LMS використовують для цифрової обробки сигналів (більшість телефонних ліній вико-

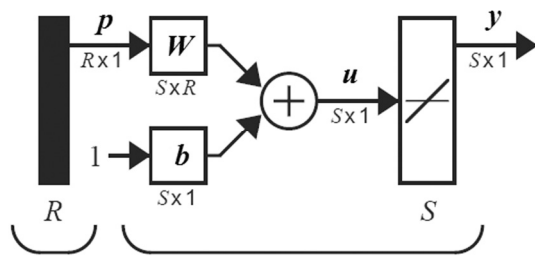


Рис. 6.1. Лінійна одношарова мережа (скорочена форма зображення):  
 $y = \text{purelin}(\mathbf{W}\mathbf{p} + \mathbf{b})$

ристовують мережу АДАЛІН для зменшення або повного усунення повторних сигналів).

У загальному випадку мережа АДАЛІН (рис. 6.1) – це лінійна мережа, що має  $S$  нейронів, пов’язаних з  $R$  входами через вагову матрицю  $\mathbf{W}$ . Вектор виходу мережі має розмірність  $[S \times 1]$  і в системі MatLab визначається за формулою  $y = \text{purelin}(\mathbf{W}\mathbf{p} + \mathbf{b}) =$

$= \mathbf{W}\mathbf{p} + \mathbf{b}$ . Нагадаємо, що  $i$ -й елемент вихідного вектора мережі можна записати як  $y_i = \text{purelin}(u_i) = {}_i\mathbf{w}^T \mathbf{p} + b_i$ , де  ${}_i\mathbf{w}$  складається з елементів  $i$ -го

рядка матриці  $\mathbf{W}$ :  ${}_i\mathbf{w} = \begin{bmatrix} w_{i1} \\ \dots \\ w_{iR} \end{bmatrix}$ .

Розглянемо мережу АДАЛІН, яка складається з одного нейрона із двома входами (рис. 6.2) [14]. У цьому випадку вагова матриця  $\mathbf{W}$  має тільки один рядок  ${}_1\mathbf{w}^T$  і вихід мережі визначають за формулою

$$y = \text{purelin}(u) = \text{purelin}({}_1\mathbf{w}^T \mathbf{p} + b) = {}_1\mathbf{w}^T \mathbf{p} + b = w_{11}p_1 + w_{12}p_2 + b.$$

Нагадаємо, що персептрон обчислює граничний розв’язок у вигляді прямої, яка визначається вхідними векторами  $\mathbf{p}$ :  ${}_1\mathbf{w}^T \mathbf{p} + b = 0$ . У мережі

АДАЛПН ця пряма теж визначає граничний розв'язок як лінію, що розділяє вхідні дані (рис. 6.3). Вектори входу, розміщені вище цієї лінії, відповідають додатним значенням виходу, а розміщені нижче – від'ємним. Отже, мережу АДАЛПН можна використовувати для вирішення задач класифікації, але тільки для лінійно подільних об'єктів, тому АДАЛПН має ті самі обмеження, що й персептрон.

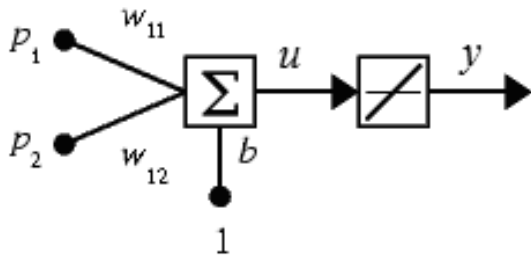


Рис. 6.2. Модель лінійного нейрона із двома входами  $y = \text{purelin}(\mathbf{w}^T \mathbf{p} + b)$

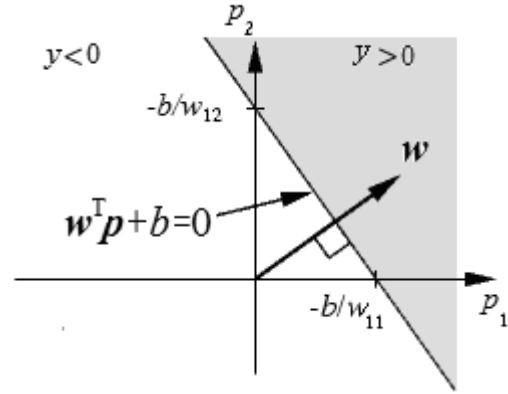


Рис. 6.3. Граничний розв'язок для мережі АДАЛПН із двома входами

## 6.2. Середньоквадратична похибка мережі АДАЛПН

Як і правило навчання персептрона, алгоритм LMS є прикладом керованого навчання мережі АДАЛПН, в якому правило навчання є набором прикладів правильної поведінки мережі:  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , де  $\mathbf{p}$  – вхідний сигнал мережі, а  $\mathbf{t}$  – цільовий сигнал, який йому відповідає.

Алгоритм LMS налаштовує вагові коефіцієнти та зсув мережі АДАЛПН таким чином, щоб мінімізувати середньоквадратичну похибку, яка визначається як різниця між цільовим і вихідним сигналами мережі. Розглянемо, як змінюється середньоквадратична похибка для одного нейрона із  $R$  входами. Для спрощення об'єднаємо всі параметри нейрона в один вектор:

$\mathbf{x} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$ , а також включимо до складу вектора  $\mathbf{z}$  вхід  $\mathbf{p}$  і вхідний сигнал, який дорівнює одиниці й відповідає зсуву  $b$ :  $\mathbf{z} = [\mathbf{p} \ 1]^T$ . Тепер вихідний сигнал мережі  $y = \mathbf{w}^T \mathbf{p} + b$  можна записати у вигляді  $y = \mathbf{x}^T \mathbf{z}$ , що дозволяє записати середньоквадратичну похибку мережі АДАЛПН як

$$\begin{aligned} F(\mathbf{x}) &= E[e^2] = E[(t - y)^2] = E\left[(t - \mathbf{x}^T \mathbf{z})^2\right] = E\left[t^2 - 2t\mathbf{x}^T \mathbf{z} + \mathbf{x}^T \mathbf{z} \mathbf{z}^T \mathbf{x}\right] = \\ &= E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z} \mathbf{z}^T] \mathbf{x}, \end{aligned}$$

де математичне сподівання  $E[\cdot]$  поширюється на всю множину пар вхід–ціль. Отриманий вираз можна записати у такому вигляді:

$$F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{B} \mathbf{x}, \quad (6.1)$$

де  $c = E[t^2]$ ;  $\mathbf{h} = E[t\mathbf{z}]$ ;  $\mathbf{B} = E[\mathbf{z}\mathbf{z}^T]$ , де вектор  $\mathbf{h}$  – кореляція між вектором входу та відповідним йому цільовим значенням;  $\mathbf{B}$  – вхідна кореляційна матриця, діагональні елементи якої дорівнюють середнім квадратичним значенням елементів вхідних векторів. Порівнявши вираз (6.1) із загальною формою квадратичної функції  $F(\mathbf{x}) = c + \mathbf{d}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$ , можна поба-

чити, що середньоквадратична похибка мережі АДАЛПН має вигляд квадратичної функції, якщо  $\mathbf{d} = -2\mathbf{h}$ ;  $\mathbf{A} = 2\mathbf{B}$ . Нагадаємо, що особливості квадратичної функції насамперед залежать від матриці Гессе  $\mathbf{A}$  (наприклад, якщо всі власні значення матриці Гессе додатні, то функція має один глобальний мінімум). У цьому випадку матриця Гессе  $\mathbf{A} = 2\mathbf{B}$ . Матриця кореляції  $\mathbf{B}$  має бути, принаймні, додатно напіввизначеною. Якщо  $\mathbf{B}$  має нульове власне значення, то функція або має слабкий мінімум, або взагалі не має стаціонарної точки, інакше вона матиме єдиний глобальний мінімум.

Можна показати, що матриця кореляції  $\mathbf{B}$  є або додатно визначеною, або додатно напіввизначеною, тобто  $\mathbf{B}$  не може мати від’ємних власних значень. Залишаються два можливих випадки:

1) якщо матриця кореляції має лише додатні власні значення, то середньоквадратична похибка має єдиний глобальний мінімум;

2) якщо матриця кореляції має принаймні одне нульове власне значення, то середньоквадратична похибка або має слабкий мінімум, або взагалі не має стаціонарної точки (залежно від вектора  $\mathbf{d} = -2\mathbf{h}$ ).

Визначимо стаціонарну точку середньоквадратичної похибки мережі АДАЛПН. Нагадаємо, що градієнт квадратичної функції має вигляд

$$\nabla F(\mathbf{x}) = \nabla \left( c + \mathbf{d}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} \right) = \mathbf{d} + \mathbf{A} \mathbf{x} = -2\mathbf{h} + 2\mathbf{B} \mathbf{x}. \quad (6.2)$$

Стаціонарну точку  $F(\mathbf{x})$  можна знайти, прирівнявши градієнт до нуля:  $-2\mathbf{h} + 2\mathbf{B} \mathbf{x} = 0$ , тому, якщо матриця кореляції  $\mathbf{B}$  додатно визначена, то існує єдина стаціонарна точка, яка є стійким мінімумом:

$$\mathbf{x}^* = \mathbf{B}^{-1} \mathbf{h}. \quad (6.3)$$

Отже, існування єдиного розв’язку залежить лише від матриці кореляції  $\mathbf{B}$ , тобто від особливостей вхідних векторів.

### 6.3. Алгоритм LMS

Розглянемо алгоритм визначення точки мінімуму середньоквадратичної похибки. Якщо б можна було визначити статистичні значення  $\mathbf{h}$  та  $\mathbf{B}$ , то точку мінімуму можна було б визначити безпосередньо з виразу (6.3). У загальному випадку важко обчислити значення вектора  $\mathbf{h}$  та матриці  $\mathbf{B}$ , тому використаємо наближений градієнт. Б. Відроу та М. Хофф оцінили середньоквадратичну похибку  $F(\mathbf{x})$ , замінивши математичне сподівання квадратом похибки на  $k$ -й ітерації:  $\bar{F}(\mathbf{x}) = e^2(k) = (t(k) - y(k))^2$ . Тоді на  $k$ -й ітерації маємо градієнт вигляду  $\nabla \bar{F}(\mathbf{x}) = \nabla e^2(k)$ . Перші  $R$  елементів вектора  $\nabla e^2(k)$  – це похідні за ваговими коефіцієнтами мережі, а  $(R + 1)$ -й елемент – похідна за зсувом. Таким чином, маємо: 
$$\left[ \nabla e^2(k) \right]_j =$$

$$= \frac{\partial e^2(k)}{\partial w_{1j}} = 2e(k) \frac{\partial e(k)}{\partial w_{1j}} \quad (j = 1, \dots, R); \quad \left[ \nabla e^2(k) \right]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k) \frac{\partial e(k)}{\partial b}.$$

Тепер розглянемо часткові похідні елементів на кінцях цих рівнянь. Спочатку оцінимо часткову похідну  $e(k)$  за вагою:

$$\begin{aligned} \frac{\partial e(k)}{\partial w_{1j}} &= \frac{\partial [t(k) - y(k)]}{\partial w_{1j}} = \frac{\partial}{\partial w_{1j}} \left[ t(k) - (\mathbf{w}^T \mathbf{p}(k) + b) \right] = \\ &= \frac{\partial}{\partial w_{1j}} \left[ t(k) - \left( \sum_{i=1}^R w_{1i} p_i(k) + b \right) \right], \end{aligned}$$

де  $p_i(k)$  –  $i$ -й елемент вектора входу на  $k$ -й ітерації. Цей вираз спрощується до вигляду  $\frac{\partial e(k)}{\partial w_{1j}} = -p_j(k)$ . Аналогічним способом можна отримати останній елемент градієнта:  $\frac{\partial e(k)}{\partial b} = -1$ . Оскільки  $\mathbf{p}_j(k)$  та  $1$  – елементи вектора входу  $\mathbf{z}$ , то градієнт середньоквадратичної похибки на  $k$ -й ітерації можна записати у вигляді

$$\nabla \bar{F}(\mathbf{x}) = \nabla e^2(k) = -2e(k) \mathbf{z}(k). \quad (6.4)$$

Це наближення до  $\nabla F(\mathbf{x})$  можна використати в алгоритмі найшвидшого спуску, для якого формула навчання з постійною швидкістю навчання  $\alpha$  має вигляд:  $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$ . Підставивши замість  $\nabla F(\mathbf{x})$  вираз (6.4), одержимо:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k) \mathbf{z}(k), \quad (6.5)$$

або  ${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$  і  $b(k+1) = b(k) + 2\alpha e(k)$ . Отже, алгоритм LMS навчання *одного нейрона* має вигляд методу найменших середніх квадратів, який також називають правилом дельти, або алгоритмом Відроу–Хоффа. Для багатьох нейронів (рис. 6.1)  $i$ -й рядок вагової матриці та  $i$ -й елемент вектора зсуву на  $(k+1)$ -й ітерації обчислюють за формулами  ${}_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i(k)\mathbf{p}(k)$ ;  $b_i(k+1) = b_i(k) + 2\alpha e_i(k)$ , де  $e_i(k)$  –  $i$ -й елемент похибки на  $k$ -й ітерації.

Алгоритм LMS можна також записати у матричній формі:  $\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k)\mathbf{p}^T(k)$ ;  $\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k)$ .

## 6.4. Аналіз збіжності алгоритму LMS для одного нейрона

Аналіз збіжності алгоритму найшвидшого спуску показав, що верхня межа швидкості навчання для КФ має вигляд  $\alpha < 2/\lambda_{\max}$ , де  $\lambda_{\max}$  – найбільше власне значення матриці Гессе. Дослідимо збіжність алгоритму LMS, який є наближенням до алгоритму найшвидшого спуску, і покажемо, що результат не зміниться. Зазначимо, що в алгоритмі LMS (формула (6.5))  $\mathbf{x}_k$  залежить лише від  $\mathbf{z}(0), \dots, \mathbf{z}(k-1)$ . Якщо припустити, що вхідні вектори статистично незалежні, то  $\mathbf{x}_k$  не залежать від  $\mathbf{z}(k)$ . Покажемо, що для входів, які задовольняють цю умову, математичне сподівання вагового вектора буде збігатися до мінімуму середньоквадратичної похибки  $\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{h}$ . Нагадаємо, що алгоритм LMS має вигляд  $\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$ . Взявши математичне сподівання від обох сторін, одержимо  $E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha E[e(k)\mathbf{z}(k)]$ . Замінивши  $e(k)$  на  $(t(k) - \mathbf{x}_k^T \mathbf{z}(k))$ , матимемо  $E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha (E[t(k)\mathbf{z}(k)] - E[(\mathbf{x}_k^T \mathbf{z}(k))\mathbf{z}(k)])$ . Замінивши  $\mathbf{x}_k^T \mathbf{z}(k) = \mathbf{z}^T(k)\mathbf{x}_k$ , одержимо  $E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha (E[t(k)\mathbf{z}(k)] - E[\mathbf{z}(k)\mathbf{z}^T(k)]\mathbf{x}_k)$ .

Оскільки  $\mathbf{x}_k$  не залежить від  $\mathbf{z}(k)$ , маємо  $E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha(\mathbf{h} - \mathbf{B}E[\mathbf{x}_k])$ . Це можна записати у вигляді  $E[\mathbf{x}_{k+1}] = [\mathbf{I} - 2\alpha\mathbf{B}]E[\mathbf{x}_k] + 2\alpha\mathbf{h}$ . Ця динамічна система буде стійкою, якщо всі власні значення матриці  $[\mathbf{I} - 2\alpha\mathbf{B}]$  потрапляють в інтервал  $(-1; 1)$ . Як відомо, власними значеннями матриці  $[\mathbf{I} - 2\alpha\mathbf{B}]$  є  $(1 - 2\alpha\lambda_i)$ , де  $\lambda_i$  – власні значення матриці  $\mathbf{B}$ , тому система буде стабільною, якщо  $(1 - 2\alpha\lambda_i) > -1$ . Оскільки  $\lambda_i > 0$ , то  $(1 - 2\alpha\lambda_i) < 1$ . Таким чином, умова стабільності має вигляд  $\alpha < 1/\lambda_i$  для будь-якого  $i$ , або  $0 < \alpha < 1/\lambda_{\max}$ . Ця умова еквівалентна умові, яку було отримано раніше для алгоритму найшвидшого спуску, хоча тоді використовувалися власні зна-



чення матриці Гессе  $\mathbf{A}$ . Тепер використовуються власні значення вхідної матриці кореляції  $\mathbf{B}$  (нагадаємо, що  $\mathbf{A} = 2\mathbf{B}$ ). У разі виконання умови стабільності системи маємо стійкий розв'язок, який обчислюють за формулою  $E[\mathbf{x}_s] = [\mathbf{I} - 2\alpha\mathbf{B}] \cdot E[\mathbf{x}_s] + 2\alpha\mathbf{h}$ , або  $E[\mathbf{x}_s] = \mathbf{B}^{-1}\mathbf{h} = \mathbf{x}^*$ . Таким чином, розв'язок алгоритму LMS збігається зі значенням точки мінімуму середньоквадратичної похибки  $\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{h}$ .

**Приклад 6.1.** Розглянемо задачу розпізнавання мережею АДАЛПН образів, які задано у вигляді векторів:  $\{\mathbf{p}_1 = [-1 \ 1 \ -1]^T, t_1 = -1\}$ ;  $\{\mathbf{p}_2 = [1 \ 1 \ -1]^T, t_2 = 1\}$ . Припустимо, що мережа має нульовий зсув. Алгоритм LMS оновлює вагу за формулою  $\mathbf{w}(k+1) = \mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$ . Обчислимо максимально можливе стійке значення швидкості навчання  $\alpha$ . Для цього визначимо власні значення вхідної матриці кореляції. Якщо припустити, що вхідні вектори згенерували випадковим чином з однаковою ймовірністю, то вхідна матриця кореляції має такий вигляд:

$$\begin{aligned}\mathbf{B} = E[\mathbf{p}\mathbf{p}^T] &= \frac{1}{2}\mathbf{p}_1\mathbf{p}_1^T + \frac{1}{2}\mathbf{p}_2\mathbf{p}_2^T = \frac{1}{2}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \cdot [-1 \ 1 \ -1] + \\ &+ \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \cdot [1 \ 1 \ -1] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}.\end{aligned}$$

Власні значення матриці  $\mathbf{B}$  такі:  $\lambda_1 = 1$ ;  $\lambda_2 = 0$ ;  $\lambda_3 = 2$ . Таким чином, максимально можливе стійке значення швидкості навчання  $\alpha < \frac{1}{\lambda_{\max}} = \frac{1}{2} = 0,5$ . Оберемо  $\alpha = 0,2$ . На практиці не зручно обчислювати матрицю  $\mathbf{B}$ , тому  $\alpha$  підбирають через проведення випробувань і обчислення похибки. Розпочнемо алгоритм зі значення  $\mathbf{w} = [0 \ 0 \ 0]^T$ . Вводимо вектори  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_1, \mathbf{p}_2$  і т. д. (у зазначеній послідовності), кожного разу обчислюючи нові значення ваги.

*Крок 1.* Уведемо вектор  $\mathbf{p}(0) = \mathbf{p}_1$  і цільове значення  $t(0) = t_1 = -1$ , одержимо:

$$\begin{aligned}y(0) &= \mathbf{w}^T(0)\mathbf{p}(0) = \mathbf{w}^T(0)\mathbf{p}_1 = [0 \ 0 \ 0] \cdot [-1 \ 1 \ -1]^T = 0; \\ e(0) &= t(0) - y(0) = -1 - 0 = -1.\end{aligned}$$

Обчислимо новий ваговий вектор:

$$\mathbf{w}(1) = \mathbf{w}(0) + 2\alpha e(0)\mathbf{p}(0) = [0 \ 0 \ 0]^T + 2 \cdot 0,2 \cdot (-1) \cdot \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = [0,4 \ -0,4 \ 0,4]^T.$$

*Крок 2.* Уведемо вектор  $\mathbf{p}(1) = \mathbf{p}_2$  і цільове значення  $t(1) = t_2 = 1$ , одержимо:

$$y(1) = \mathbf{w}^T(1)\mathbf{p}(1) = \mathbf{w}^T(1)\mathbf{p}_2 = [0,4 \quad -0,4 \quad 0,4] \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = -0,4;$$

$$e(1) = t(1) - y(1) = 1 - (-0,4) = 1,4.$$

Тепер обчислимо новий ваговий вектор:

$$\begin{aligned} \mathbf{w}(2) &= \mathbf{w}(1) + 2\alpha e(1)\mathbf{p}(1) = [0,4 \quad -0,4 \quad 0,4]^T + 2 \cdot 0,2 \cdot (1,4) \cdot \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \\ &= [0,96 \quad 0,16 \quad -0,16]^T. \end{aligned}$$

*Крок 3.* Знову введемо вектор  $\mathbf{p}_1$ , одержимо:

$$y(2) = \mathbf{w}^T(2)\mathbf{p}(2) = \mathbf{w}^T(2)\mathbf{p}_1 = [0,96 \quad 0,16 \quad -0,16] \cdot \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = -0,64;$$

$$e(2) = t(2) - y(2) = -1 - (-0,64) = -0,36.$$

Новий ваговий вектор має вигляд

$$\mathbf{w}(3) = \mathbf{w}(2) + 2\alpha e(2)\mathbf{p}(2) = [1,104 \quad 0,016 \quad -0,016]^T.$$

Якщо продовжимо цей процес, то алгоритм прямуватиме до  $\mathbf{w}(\infty) = [1 \quad 0 \quad 0]^T$ . Таким чином, мережа АДАЛІН отримала результат, який збігається з результатом навчання персептрона. Граничний розв'язок міститься посередині між двома входами:

1) правило навчання персептрона на основі множини навчання надає такий самий розв'язок, оскільки персептрон зупиняє процес навчання як тільки дані навчання правильно класифіковані (навіть якщо деякі з них містяться близько до границі поділу);

2) алгоритм LMS мінімізує середньоквадратичну похибку і намагається перемістити граничний розв'язок якнайдалі від вхідних даних.

## 6.5. Адаптивне фільтрування

Незважаючи на те, що мережа АДАЛПН може розв'язувати тільки лінійно подільні задачі, її використовують частіше, ніж мережу типу персептрон. Однією з найголовніших сфер застосування мережі АДАЛПН є адаптивне фільтрування. Для використання мережі АДАЛПН як адаптивного фільтра потрібно ввести новий складовий блок мережі – лінію затримки із  $R$  виходами (рис. 6.4, а). На виході лінії затримки маємо  $R$ -вимірний вектор, який складається із вхідного сигналу в поточний момент та затримок від 1 до  $R - 1$  часових кроків. Об'єднаємо лінію затримки з мережею АДАЛПН і створимо адаптивний фільтр (рис. 6.4, б). Вихід фільтра має вигляд

$$y(k) = \text{purelin}(\mathbf{W}\mathbf{p}(k) + b) = \sum_{i=1}^R w_{1i}v(k-i+1) + b,$$

де  $\mathbf{p}(k) = [p_1(k) \dots p_R(k)]$  і  $p_i(k) = v(k-i+1)$ ,  $1 \leq i \leq R$ .

Розглянемо приклад використання адаптивного фільтра для видалення шумів. Похибка виходу, яку мережа намагається мінімізувати, є, фактично, апроксимацією сигналу.

**Приклад 6.2.** Нехай лікар розглядає електроенцефалограму (ЕЕГ) аспіранта, якого відволікають, і бачить, що сигнал ЕЕГ забруднений шумовим джерелом на 60 Гц. Лікар досліджує пацієнта он-лайн і хоче побачити найкращий можливий сигнал. На рис. 6.5 зображено, як адаптивний фільтр можна використати для видалення шумового сигналу: зразок сигналу зі значенням 60 Гц подається на адаптивний фільтр, елементи якого пристосовані до мінімізації похибки  $e$ .

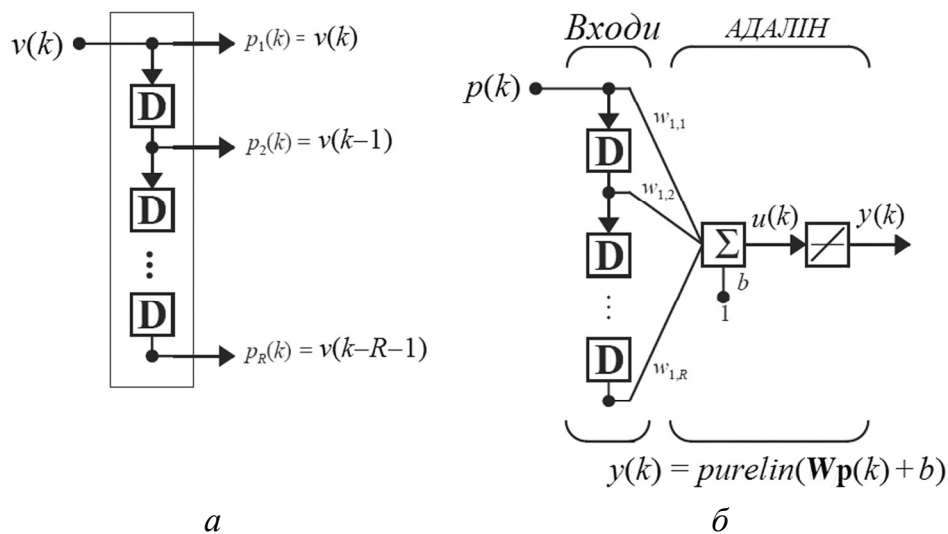


Рис. 6.4. Лінія затримки (а), адаптивний фільтр АДАЛПН (б)

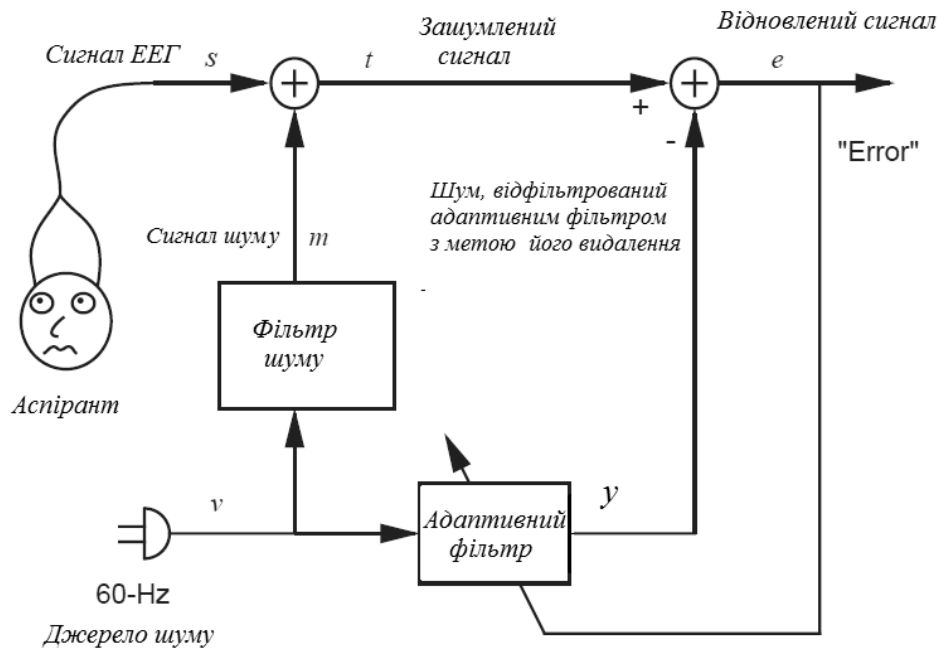


Рис. 6.5. Система видалення шумів (адаптивний фільтр мінімізує похибку, видаляючи шум значенням 60 Гц із забрудненого сигналу)

Цільовий вихід фільтра  $t$  – це забруднений сигнал ЕЕГ. Адаптивний фільтр намагається відтворити цей забруднений сигнал, але він знає тільки про джерело шуму  $v$ . Фільтр може лише відтворити частину  $t$  сигналу  $t$ , яка лінійно корельована з  $v$ . Насправді адаптивний фільтр намагається наслідувати фільтр шуму, так, щоб його вихід  $y$  був близьким за значенням до забрудненого сигналу  $t$ . Таким чином, похибка  $e$  буде близькою до незабрудненого сигналу ЕЕГ  $s$ .

Розглянемо простий випадок: наявність одного джерела шуму у вигляді синусоїдальної хвилі. Щоб реалізувати фільтр, достатньо мати нейрон із двома ваговими коефіцієнтами і нульовим зсувом. Входами у фільтр є поточне та попереднє значення джерела шуму. Такий фільтр із двома входами (рис. 6.6) може зменшити фази шуму  $v$  бажаним способом.

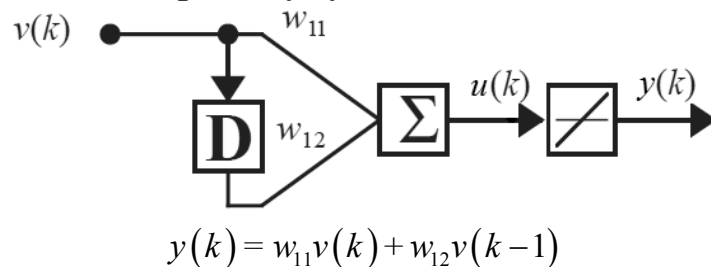


Рис. 6.6. Адаптивний фільтр АДАЛІН для видалення шумів

Щоб проаналізувати цю систему, застосуємо отримані раніше математичні співвідношення. Для цього спочатку знайдемо вхідну матрицю кореляції  $\mathbf{B} = [\mathbf{z}\mathbf{z}^T]$  і вектор кореляції  $\mathbf{h} = E[\mathbf{t}\mathbf{z}]$ . Вхідний вектор зада-

ний поточним і попереднім значеннями джерела шуму:  $\mathbf{z}(k) = \begin{bmatrix} v(k) \\ v(k-1) \end{bmatrix}$ ,

тоді як цільове значення – це сума поточного сигналу та фільтрованого шуму:  $t(k) = s(k) + m(k)$ . Врахувавши це, отримаємо такі вирази для  $\mathbf{B}$  та  $\mathbf{h}$ :

$$\mathbf{B} = \begin{bmatrix} E[v^2(k)] & E[v(k)v(k-1)] \\ E[v(k-1)v(k)] & E[v^2(k-1)] \end{bmatrix}; \quad \mathbf{h} = \begin{bmatrix} E[(s(k)+m(k))v(k)] \\ E[(s(k)+m(k))v(k-1)] \end{bmatrix}.$$

Щоб  $\mathbf{B}$  та  $\mathbf{h}$  набули певних значень, треба визначити шум  $v$ , сигнал ЕЕГ  $s$  і фільтрований шум  $m$ . Оберемо:

1) сигнал ЕЕГ  $s$  – це випадковий сигнал, однорідно розподілений між значеннями  $-0,2$  і  $0,2$  (некорельований від одного часового кроку до наступного);

2) джерело шуму (синусоїдальна хвиля 60 Гц, збільшена до 180 Гц):  

$$v(k) = 1,2 \sin\left(\frac{2\pi k}{3}\right);$$

3) фільтрований шум, який забруднює ЕЕГ і є джерелом шуму з коефіцієнтом 0,1 і зміщенням за фазою  $\frac{\pi}{2}$ :  $m(k) = 0,12 \sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right)$ .

Тепер обчислимо елементи вхідної матриці кореляції  $\mathbf{B}$ :

$$E[v^2(k)] = (1,2)^2 \frac{1}{3} \sum_{k=1}^3 \left( \sin\left(\frac{2\pi k}{3}\right) \right)^2 = (1,2)^2 0,5 = 0,72;$$

$$E[v^2(k-1)] = E[v^2(k)] = 0,72;$$

$$\begin{aligned} E[v(k)v(k-1)] &= \frac{1}{3} \sum_{k=1}^3 \left( 1,2 \sin\frac{2\pi k}{3} \right) \left( 1,2 \sin\frac{2\pi(k-1)}{3} \right) = \\ &= (1,2)^2 0,5 \cos\left(\frac{2\pi}{3}\right) = -0,36. \end{aligned}$$

Отже, матриця  $\mathbf{B}$  набуває такого вигляду:  $\mathbf{B} = \begin{bmatrix} 0,72 & -0,36 \\ -0,36 & 0,72 \end{bmatrix}$ .

Визначимо елементи вектора  $\mathbf{h}$ .

Для першого елемента маємо:  $E[(s(k)+m(k))v(k)] = E[s(k)v(k)] + E[m(k)v(k)]$ , де  $E[s(k)v(k)] = 0$ , оскільки  $s(k)$  і  $v(k)$  незалежні та мають нульові середні; другий доданок:  $E[m(k)v(k)] = \frac{1}{3} \sum_{k=1}^3 \left( 0,12 \sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right) \right) \times$

$\times \left(1,2 \sin \frac{2\pi k}{3}\right) = 0$ . Таким чином, перший елемент вектора  $\mathbf{h}$  дорівнює нулю.

Для другого елемента маємо:  $E[(s(k) + m(k))v(k-1)] = E[s(k)v(k-1)] + E[m(k)v(k-1)]$ , де  $E[s(k)v(k-1)] = 0$ , оскільки  $s(k)$  і  $v(k-1)$  незалежні та мають нульові середні. Другий доданок оцінимо таким чином:  $E[m(k)v(k-1)] = \frac{1}{3} \sum_{k=1}^3 \left(0,12 \sin \left(\frac{2\pi k}{3} + \frac{\pi}{2}\right)\right) \left(1,2 \sin \frac{2\pi(k-1)}{3}\right) = -0,0624$ .

Отже,  $\mathbf{h}$  набуває такого вигляду:  $\mathbf{h} = \begin{bmatrix} 0 \\ -0,0624 \end{bmatrix}$ . Точка мінімуму серед-

ньоквадратичної похибки:  $\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{h} = \begin{bmatrix} 0,72 & -0,36 \\ -0,36 & 0,72 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -0,0624 \end{bmatrix} = \begin{bmatrix} -0,0578 \\ -0,1156 \end{bmatrix}$ .

Визначимо, яка похибка відповідає такому значенню. Для цього нагадаємо вираз середньоквадратичної похибки:

$$F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{B} \mathbf{x}. \quad (6.7)$$

Оскільки значення  $\mathbf{x}^*$ ,  $\mathbf{h}$  і  $\mathbf{B}$  вже відомі, то залишилося знайти  $c$ :

$$c = E[t^2(k)] = E[(s(k) + m(k))^2] = E[s^2(k)] + 2E[s(k)m(k)] + E[m^2(k)].$$

Доданок  $2E[s(k)m(k)] = 0$ , оскільки  $s(k)$  і  $m(k)$  незалежні та мають нульові середні. Елемент  $E[s^2(k)]$  можна обчислити таким чином:

$$E[s^2(k)] = \frac{1}{0,4} \int_{-0,2}^{0,2} s^2 ds = \frac{1}{1,2} s^3 \Big|_{-0,2}^{0,2} = 0,0133. \text{ Елемент } E[m^2(k)] - \text{серед-}$$

ньоквадратичне значення фільтрованого шуму,  $E[m^2(k)] =$

$$= \frac{1}{3} \sum_{t=1}^3 \left(0,12 \sin \left(\frac{2\pi}{3} + \frac{\pi}{2}\right)\right)^2 = 0,0072. \text{ Отже, маємо } c = 0,0133 + 0,0072 = 0,0205.$$

Підставивши значення  $\mathbf{x}^*$ ,  $c$ ,  $\mathbf{h}$  і  $\mathbf{B}$  у вираз (6.7), одержимо значення мінімальної середньоквадратичної похибки:  $F(\mathbf{x}^*) = 0,0205 - 2(0,0072) + 0,0072 = 0,0133$ . Мінімальна середньоквадратична похибка – це те саме, що і середньоквадратичне значення сигналу ЕЕГ. Це фактично відновлений сигнал ЕЕГ.

Траєкторію алгоритму LMS у ваговому просторі з кроком навчання  $\alpha = 0,1$  зображено на рис. 6.7. За такого моделювання вага  $w_{11}$  і  $w_{12}$  набуває значень від «0» до «-2» та від «1» до «-2» відповідно. Траєкторія LMS схожа на траєкторію найшвидшого спуску за наявності шуму.

Зазначимо, що контурний графік функції відповідає власним значенням і власним векторам матриці Гессе ( $\mathbf{A} = 2\mathbf{B}$ ), які мають такий вигляд:

$$\lambda_1 = 2,16, \mathbf{z}_1 = \begin{bmatrix} -0,7071 \\ 0,7071 \end{bmatrix}; \lambda_2 = 0,72, \mathbf{z}_2 = \begin{bmatrix} -0,7071 \\ -0,7071 \end{bmatrix}.$$

Максимально стабільне значення швидкості навчання  $\alpha < \frac{2}{2,16} = 0,926$ .

Щоб оцінити продуктивність адаптивного фільтра щодо видалення шуму, розглянемо рис. 6.8, який ілюструє процес видалення шуму.

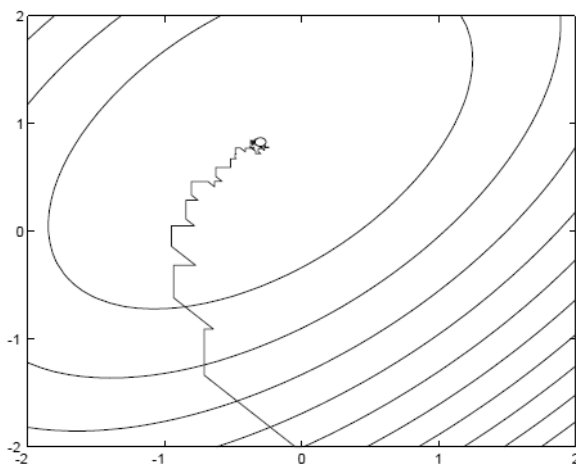


Рис. 6.7. Траєкторія виконання алгоритму LMS (якщо  $\alpha = 0,1$ )

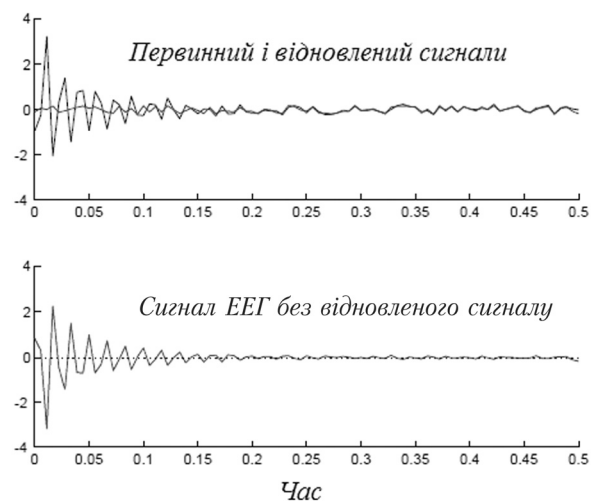


Рис. 6.8. Адаптивний фільтр видаляє шум

На *верхньому графіку* зображено відновлений і первинний сигнали ЕЕГ. На вхід фільтра подається первинний сигнал і бажаний відгук, фільтр визначає відновлений сигнал виходу і середньоквадратичну похибку (як різницю між первинним і відновленим сигналами). Спочатку відновлений сигнал є поганою апроксимацією первинного, але приблизно через 0,2 секунди (якщо  $\alpha = 0,1$ ) фільтр видає правильний відновлений сигнал. На *нижньому графіку* зображено відмінність між первинним і відновленим сигналами. Зазначимо, що алгоритм LMS реалізує приблизний метод найшвидшого спуску: для відновлення значення ваги він використовує оцінку градієнта (а не його реальне значення). Алгоритм LMS є сильнішим за алгоритм персептрона, оскільки мінімізує середньоквадратичну похибку і може визначити граничні розв'язки, на які майже не впливає шум.

Визначимо *основні властивості лінійних мереж*.

1. Одношарові лінійні мережі здатні розв'язувати задачі лінійної апроксимації функцій і розпізнавання образів. Архітектура мережі

визначається завданням. Мережі можна навчити за допомогою правила навчання Відроу–Хоффа.

2. Лінійну мережу можна успішно навчити тільки у випадку, коли її входи та виходи лінійно пов'язані між собою. Проте навіть якщо лінійна мережа не може визначити точного розв'язку, вона може побудувати приблизний розв'язок (у сенсі мінімуму середньоквадратичної похибки) за умови, що швидкість навчання  $\alpha$  досить мала. Така мережа відповідно до своєї лінійної структури визначить найбільш точний розв'язок. Це зумовлено тим, що графік поверхні похибки навчання (багатовимірний параболоїд) має єдиний мінімум, і алгоритм градієнтного спуску повинен звести розв'язок до цього мінімуму.

### Приклади розв'язання задач\*

**Приклад 6.3.** Нехай задано структуру та параметри фільтра АДАЛІН (рис. 6.9), в якому  $w_{11} = 2$ ;  $w_{12} = -1$ ;  $w_{13} = 3$ . Послідовність входів заданого фільтра така:  $\{p_i\} = \{\dots, 0, 0, 0, 5, -4, 0, 0, 0, \dots\}$ , де  $p_0 = 5$ ;  $p_1 = -4$  і т. д.

**I.** Що буде на виході фільтра безпосередньо перед  $k = 0$ ?

**II.** Що буде на виході фільтра, якщо  $k$  набуває значення від «0» до «5»?

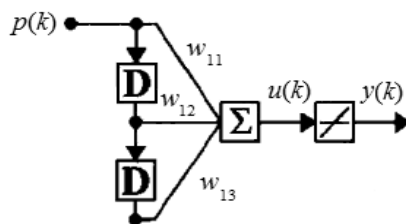


Рис. 6.9. Фільтр АДАЛІН:  $y(k) = \text{purelin}(\mathbf{w}^T \mathbf{p}(k) + b)$

**Розв'язання. I.** Безпосередньо перед  $k = 0$  маємо таку ситуацію: три нулі  $[0 \ 0 \ 0]^T$  було подано на вхід фільтра, на виході одержали значення «0».

**II.** Якщо  $k = 0$ , цифра «5» подається на вхід фільтра, а потім множиться на  $w_{11} = 2$ . Отже, маємо  $y(0) = 10$ .

Цю дію можна зобразити в матричному вигляді:  $y(0) = \mathbf{w}^T \mathbf{p}(0) =$

$$= [w_{11} \ w_{12} \ w_{13}] \cdot \begin{bmatrix} p_0 \\ p_{-1} \\ p_{-2} \end{bmatrix} = [2 \ -1 \ 3] \cdot \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} = 10. \text{ Аналогічним чином можна}$$

\* Задачі взято з посібника [14].



обчислити такі виходи:

$$y(1) = \mathbf{w}^T \mathbf{p}(1) = [2 \ -1 \ 3] \cdot \begin{bmatrix} -4 \\ 5 \\ 0 \end{bmatrix} = -13; \quad y(2) = \mathbf{w}^T \mathbf{p}(2) = [2 \ -1 \ 3] \cdot \begin{bmatrix} 0 \\ -4 \\ 5 \end{bmatrix} = 19;$$
$$y(3) = \mathbf{w}^T \mathbf{p}(3) = [2 \ -1 \ 3] \cdot \begin{bmatrix} 0 \\ 0 \\ -4 \end{bmatrix} = -12; \quad y(4) = \mathbf{w}^T \mathbf{p}(4) = [2 \ -1 \ 3] \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 0.$$

Усі інші виходи набувають значення «0».

**Приклад 6.4.** Спроектувати мережу АДАЛПН для класифікації таких двох груп вхідних векторів:

група I:  $\mathbf{p}_1 = [1 \ 1]^T$ ,  $\mathbf{p}_2 = [-1 \ -1]^T$ ; група II:  $\mathbf{p}_3 = [2 \ 2]^T$ .

**I.** Чи існує мережа АДАЛПН, здатна розв'язати поставлену задачу?

**II.** Якщо відповідь на запитання I «так», то які значення ваги та зсуву можна використати?

**III.** Розглянути також інші групи векторів:

група III:  $\mathbf{p}_1 = [1 \ 1]^T$ ;  $\mathbf{p}_2 = [1 \ -1]^T$ ; група IV:  $\mathbf{p}_3 = [1 \ 0]^T$ .

Чи існує мережа АДАЛПН, яка може класифікувати задані вектори?

**IV.** Якщо відповідь на запитання III «так», то які значення ваги та зсуву можна використати?

*Розв'язання.* **I.** На рис. 6.10, а зображено вхідні вектори. Оскільки наведені групи векторів є лінійно подільними, то існує мережа АДАЛПН, яка здатна їх успішно класифікувати й одержати відповідний граничний розв'язок.

**II.** Граничний розв'язок проходить через точки (3, 0) і (0, 3) і має вигляд  $-3p_1 = 3p_2 - 9$ . Тому рівняння прямої  $\mathbf{w}^T \mathbf{p} + b = 0$  набуває вигляду  $-p_1 - p_2 + 3 = 0$ , звідки  $w_{11} = -1$ ;  $w_{12} = -1$ ;  $b = 3$ . Зазначимо, що якщо  $\mathbf{w}^T \mathbf{p} + b \geq 0$ , мережа відносить вектор входу до групи I, а якщо  $\mathbf{w}^T \mathbf{p} + b < 0$  – до групи II. Одержаний розв'язок також передбачає похибку, оскільки граничний розв'язок ділить відстань між векторами  $\mathbf{p}_1$  та  $\mathbf{p}_3$  навпіл.

**III.** На рис. 6.10, б зображено вхідні вектори, які необхідно класифікувати. Ці вектори не є лінійно подільними, тому не існує мережі АДАЛПН, яка може їх розрізнити.

**IV.** Як зазначено у п. III, не існує мережі АДАЛПН, яка може розрізнити задані групи векторів, тому не існує таких значень ваги та зсуву.

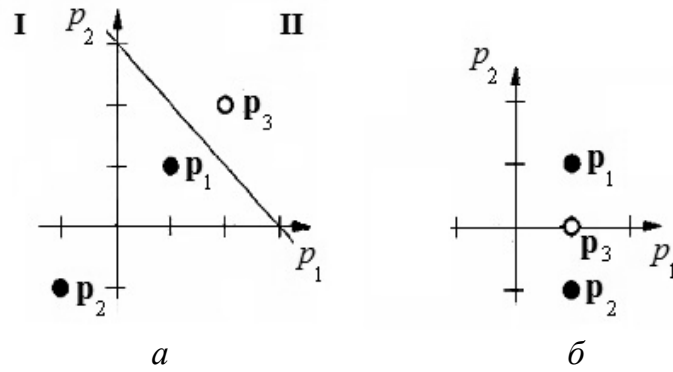


Рис. 6.10. Вхідні вектори до прикл. 6.4

**Приклад 6.5.** Нехай маємо такі пари вхід-ціль:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}$ ;  $\left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1 \right\}$ . Задані вектори подаються на вхід з однаковою ймовірністю і використовуються для навчання мережі АДАЛПН без зсуву.

Який вигляд має функція середньоквадратичної похибки?

*Розв'язання.* Спочатку треба обчислити елементи КФ, яка описує ефективність функціонування мережі АДАЛПН і має вигляд  $F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{B} \mathbf{x}$ , тобто слід визначити значення  $c$ ,  $\mathbf{h}$ ,  $\mathbf{B}$ . Ймовірність появи на вході кожного з векторів навчання дорівнює 0,5, тому ймовірність отримання кожного з цільових значень  $t$  також дорівнює 0,5. Отже, математичне сподівання квадрата цільових значень

$$c = E[t^2] = (t_1)^2 (0,5) + (t_2)^2 (0,5) = (1)^2 (0,5) + (-1)^2 (0,5) = 1.$$

Обчислимо кореляцію між входом і ціллю, одержимо:

$$\mathbf{h} = E[t\mathbf{z}] = (0,5)t_1\mathbf{p}_1 + (0,5)t_2\mathbf{p}_2 = 0,5 \cdot 1 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0,5 \cdot (-1) \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Вхідна матриця кореляції  $\mathbf{B}$  має вигляд

$$\mathbf{B} = E[\mathbf{z}\mathbf{z}^T] = \mathbf{p}_1\mathbf{p}_1^T (0,5) + \mathbf{p}_2\mathbf{p}_2^T (0,5) = 0,5 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} + 0,5 \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Отже, за  $\mathbf{x} = [w_{11} \ w_{12}]^T$  середньоквадратична похибка має вигляд

$$F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{B} \mathbf{x} = 1 - 2[w_{11} \ w_{12}] \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} + [w_{11} \ w_{12}] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = 1 - 2w_{12} + w_{11}^2 + w_{12}^2.$$

Визначимо власні значення матриці Гессе:  $\nabla^2 F(\mathbf{x}) = 2\mathbf{B} = 2 \times$

$\times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ , звідки одержимо  $\lambda_{1,2} = 2$ . Контури поверхні показника ефективності функціонування мережі будуть мати форму кола (рис. 6.11), центром якого є точка мінімуму середньоквадратичної похибки:

$$\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{h} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

**Приклад 6.6.** Нехай маємо такі пари вхід-ціль:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1 \right\}$ . Задані вектори подаються на вхід з однаковою ймовірністю і використовуються для навчання мережі АДАЛІН без зсуву.

Навчити мережу АДАЛІН за допомогою алгоритму LMS, якщо  $\mathbf{w}(0) = [0 \ 0]^T$ , швидкість навчання  $\alpha = 0,25$  (під час навчання використати кожний вхідний образ тільки один раз і визначити граничний розв'язок на кожному кроці).

*Розв'язання.* Нехай вхідний вектор  $\mathbf{p}_1$  першим подається на вхід мережі. Тоді вихід, похибка та оновлена вага мають вигляд

$$y(0) = \text{purelin} \left( [0 \ 0] \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = 0; \quad e(0) = t(0) - y(0) = 1 - 0 = 1;$$

$$\mathbf{w}(1) = \mathbf{w}(0) + 2\alpha e(0)\mathbf{p}(0) = [0 \ 0]^T + 2 \left( \frac{1}{4} \right) \cdot 1 \cdot [1 \ 1]^T = [0,5 \ 0,5]^T.$$

Граничний розв'язок, пов'язаний з вагою  $\mathbf{w}(1)$ , зображено на рис. 6.12, а. Використаємо другий вхідний вектор  $\mathbf{p}_2$ , одержимо:

$$y(1) = \text{purelin} \left( [0,5 \ 0,5] \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = 0; \quad e(1) = t(1) - y(1) = -1 - 0 = -1;$$

$$\mathbf{w}(2) = \mathbf{w}(1) + 2\alpha e(1)\mathbf{p}(1) = [0,5 \ 0,5]^T + 2 \left( \frac{1}{4} \right) (-1) [1 \ -1]^T = [0 \ 1]^T.$$

Граничний розв'язок, пов'язаний з вагою  $\mathbf{w}(2)$ , зображено на рис. 6.12, б.

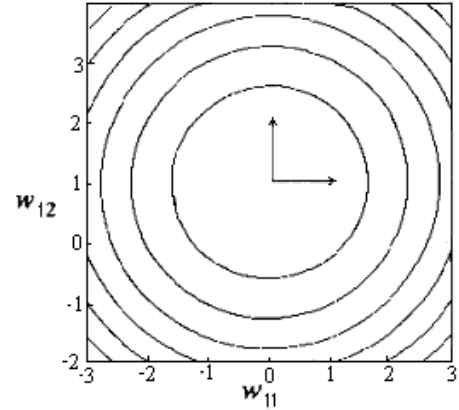


Рис. 6.11. Контурний графік функції  $F(\mathbf{x})$

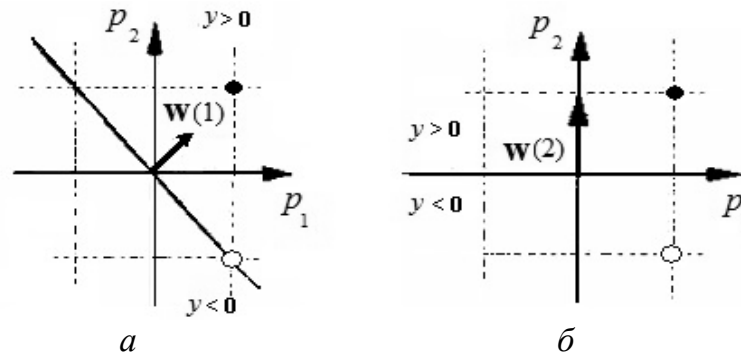


Рис. 6.12. Розв'язок до прикл. 6.6

**Приклад 6.7.** Нехай задано такі пари вхід–ціль:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}$ ;  $\left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1 \right\}$ . Задані вектори подаються на вхід з однаковою ймовірністю і використовуються для навчання мережі АДАЛПН без зсуву.

Навчити мережу АДАЛПН за допомогою алгоритму LMS, якщо  $\mathbf{w}(0) = [0 \ 0]^T$  і швидкість навчання  $\alpha = 0,25$  (під час навчання використати кожний вхідний образ тільки один раз і визначити граничний розв'язок на кожному кроці). Визначити максимально стійке значення швидкості навчання  $\alpha$ .

*Розв'язання.* Збіжність алгоритму LMS визначається коефіцієнтом швидкості навчання  $\alpha$ . Під час розв'язання задачі 6.5 було показано, що вхідна матриця кореляції  $\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ . Визначимо власні значення матриці  $\mathbf{B}$  за допомогою системи MatLab:

$$\begin{aligned} [\mathbf{V}, \mathbf{D}] &= \text{eig}(\mathbf{B}) \\ \mathbf{V} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \mathbf{D} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

Діагональні елементи матриці  $\mathbf{D}$  зображують такі власні значення матриці  $\mathbf{B}$ :  $\lambda_1 = 1$  та  $\lambda_2 = 1$ , а стовпці матриці  $\mathbf{V}$  – її власні вектори. Зазначимо, що власні вектори матриці  $\mathbf{B}$  мають той самий напрям, що й вектори, зображені на рис. 6.11. Найбільше власне значення  $\lambda_{\max} = 1$  встановлює верхню межу для коефіцієнта швидкості навчання:  $\alpha < \frac{1}{\lambda_{\max}} = 1$ .

(У прикл. 6.6 коефіцієнт навчання  $\alpha = 0,25$  й алгоритм LMS збігався швидко. Що буде, якщо швидкість навчання  $\alpha$  дорівнює одиниці або більшому значенню?)

**Приклад 6.8.** Розглянемо адаптивний фільтр АДАЛІН (рис. 6.13), який прогнозує значення вхідного сигналу на основі двох попередніх його значень. Нехай вхідний сигнал – це стаціонарний випадковий процес із функцією автокореляції вигляду  $C_v(n) = E[v(k)v(k+n)]$ ;  $C_v(0) = 3$ ;  $C_v(1) = -1$ ;  $C_v(2) = -1$ .

**I.** Накреслити графік контурних ліній показника ефективності функціонування мережі (середньоквадратичної похибки).

**II.** Визначити, яке значення коефіцієнта навчання  $\alpha$  є максимально стійким для алгоритму LMS.

**III.** Накреслити траєкторію виконання алгоритму LMS. Розпочати його з початкового припущення щодо значення ваги  $\mathbf{w}(0) = [0,75 \ 0]^T$ , припускаючи, що використовується дуже мале значення  $\alpha$ .

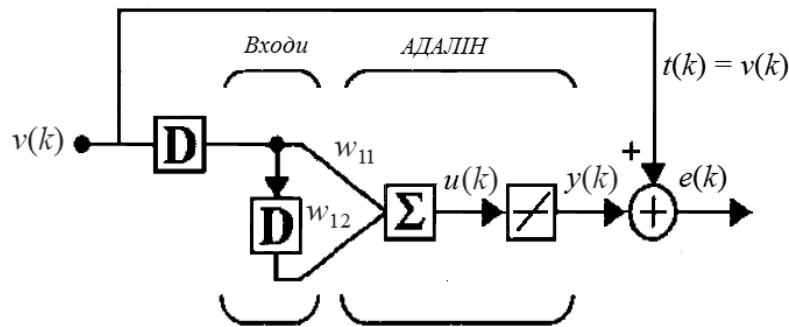


Рис. 6.13. Адаптивний фільтр АДАЛІН, який прогнозує значення вхідного сигналу на основі двох попередніх:  $y(k) = w_{11}v(k-1) + w_{12}v(k-2)$

**Розв'язання. I.** Для побудови контурних ліній поверхні показника ефективності функціонування мережі необхідно визначити середньоквадратичну похибку, а також власні значення та власні вектори матриці Гессе. Позначимо вхідний вектор у вигляді  $\mathbf{z}(k) = \begin{bmatrix} v(k-1) \\ v(k-2) \end{bmatrix}$ . Середньоквадратичну похибку мережі АДАЛІН можна зобразити як  $F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{B} \mathbf{x}$ , де  $c$ ,  $\mathbf{h}$  і  $\mathbf{B}$  мають такий вигляд:

$$c = E[t^2(k)] = E[v^2(k)] = C_v(0) = 3;$$

$$\mathbf{B} = E[\mathbf{z}^T \mathbf{z}] = E \begin{bmatrix} v^2(k-1) & v(k-1)v(k-2) \\ v(k-1)v(k-2) & v^2(k-2) \end{bmatrix} = \begin{bmatrix} C_v(0) & C_v(1) \\ C_v(1) & C_v(0) \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix};$$

$$\mathbf{h} = E[t\mathbf{z}] = E \begin{bmatrix} v(k)v(k-1) \\ v(k)v(k-2) \end{bmatrix} = \begin{bmatrix} C_v(1) \\ C_v(2) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

Отже, оптимальне значення ваги та матриця Гессе

$$\mathbf{x}^* = \mathbf{B}^{-1}\mathbf{h} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} \frac{3}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{3}{8} \end{bmatrix} \cdot \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix};$$

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{B} = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}.$$

Тепер можна визначити власні значення:  $|\mathbf{A} - \lambda \mathbf{I}| = \begin{vmatrix} 6-\lambda & -2 \\ -2 & 6-\lambda \end{vmatrix} = 0$ .

Маємо:  $\lambda^2 - 12\lambda + 32 = (\lambda - 8)(\lambda - 4) = 0$ , звідки  $\lambda_1 = 4$ ,  $\lambda_2 = 8$ .

Визначимо власні вектори за формулою  $[\mathbf{A} - \lambda \mathbf{I}]\mathbf{z} = 0$ : для  $\lambda_1 = 4$  маємо  $\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \mathbf{z}_1 = 0$ , звідки  $\mathbf{z}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ ; для  $\lambda_2 = 8$  маємо  $\begin{bmatrix} -2 & -2 \\ -2 & -2 \end{bmatrix} \mathbf{z}_2 = 0$ , звідки  $\mathbf{z}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ . Отже, контурні лінії функції  $F(\mathbf{x})$  мають вигляд еліпса

(рис. 6.14) із центром в точці  $\mathbf{x}^*$  і довгою віссю вздовж власного вектора  $\mathbf{z}_1$  (оскільки значення  $\lambda_1$  є найменшим).

**II.** Найбільше власне значення  $\lambda_{\max} = 8$  встановлює верхню межу для швидкості навчання (максимально стійку швидкість навчання):

$$\alpha < \frac{2}{\lambda_{\max}} = 0,25.$$

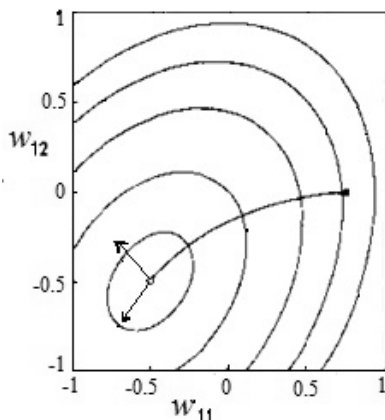


Рис. 6.14. Траєкторія виконання LMS

**III.** Оскільки алгоритм LMS є наближенням методу найшвидшого спуску, то його траєкторія за малого значення швидкості навчання ( $\alpha < 0,25$ ) буде перпендикулярною до контурних ліній (рис. 6.14).

**Приклад 6.9.** Пілот літака говорить у мікрофон своєї кабіни. Голосовий сигнал, отриманий авіадиспетчером на вежі, спотворений звуком двигуна.

Запропонуйте адаптивний фільтр АДАЛПН, який міг би зменшити шум у сигналі, що надходить до контрольно-диспетчерського пункту.

**Розв'язання.** Шум двигуна можна мінімізувати за допомогою адаптивної системи фільтрування (рис. 6.15).

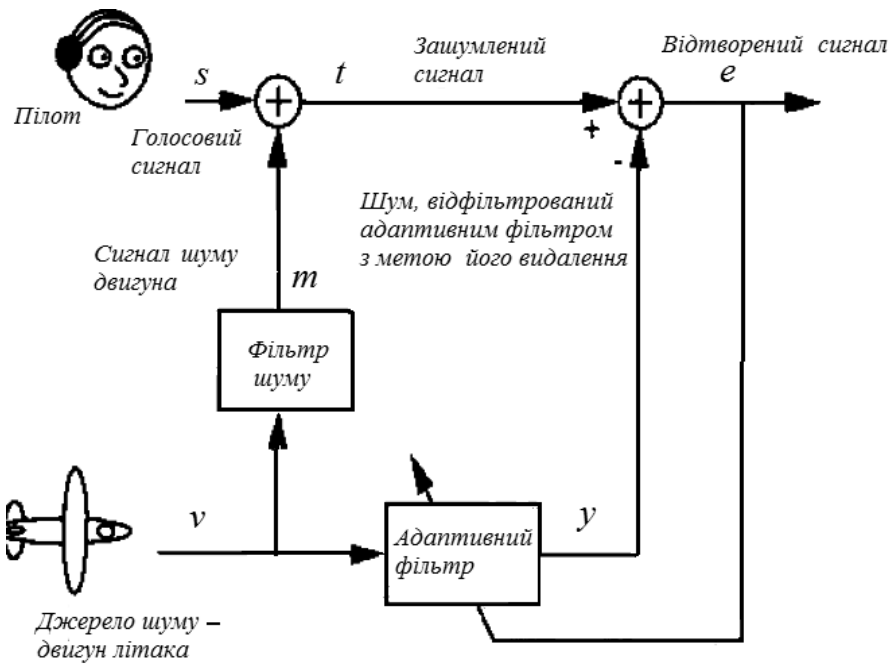


Рис. 6.15. Фільтрування шуму двигуна

Зразок шуму двигуна надходить на адаптивний фільтр через мікрофон у кабіні пілота. Бажаний вихід фільтра має вигляд зашумленого сигналу, який надходить з мікрофона. Фільтр намагається мінімізувати «похибку» сигналу (шум). Це можна зробити тільки через вилучення компоненти, яка зашумлює сигнал і є лінійно корельованою з шумом двигуна та некорельованою з голосом пілота. Результатом фільтрування є чіткий голосовий сигнал, який передається до контрольно-диспетчерського пункту, попри те, що шум двигуна потрапив до мікрофона разом з голосовим сигналом пілота.

**Приклад 6.10.** Визначити мережу АДАЛПН, яка використовує алгоритм навчання LMS і здатна класифікувати вхідні вектори на чотири класи (за умови, що кожний вхідний вектор з'являється із ймовірністю  $\frac{1}{8}$ ):

клас 1:  $\mathbf{p}_1 = [1 \ 1]^T$ ,  $\mathbf{p}_2 = [1 \ 2]^T$ ;

клас 2:  $\mathbf{p}_3 = [2 \ -1]^T$ ,  $\mathbf{p}_4 = [2 \ 0]^T$ ;

клас 3:  $\mathbf{p}_5 = [-1 \ 2]^T$ ,  $\mathbf{p}_6 = [-2 \ 1]^T$ ;

клас 4:  $\mathbf{p}_7 = [-1 \ -1]^T$ ,  $\mathbf{p}_8 = [-2 \ -2]^T$ .

Навчання мережі відбувається зі швидкістю навчання  $\alpha = 0,04$  і розпочинається з таких значень ваги та зсуву:  $\mathbf{W}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ,  $\mathbf{b}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ .

*Розв'язання.* Зобразимо задані вхідні вектори таким чином (рис. 6.16): світлі кола – вектори 1-го класу, світлі квадрати – вектори 2-го класу, темні кола – вектори 3-го класу, темні квадрати – вектори 4-го класу. Використаємо цільові вектори з елементами «1» та «-1». Тоді множина навчання має такий вигляд:

$$\text{група I: } \left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\};$$

$$\text{група II: } \left\{ \mathbf{p}_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\};$$

$$\text{група III: } \left\{ \mathbf{p}_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \mathbf{t}_5 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\};$$

$$\text{група VI: } \left\{ \mathbf{p}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{t}_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Перша ітерація:

$$\mathbf{y}(0) = \text{purelin}(\mathbf{W}(0)\mathbf{p}(0) + \mathbf{b}(0)) = \text{purelin}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 2 \\ 2 \end{bmatrix};$$

$$\mathbf{e}(0) = \mathbf{t}(0) - \mathbf{y}(0) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \end{bmatrix};$$

$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha\mathbf{e}(0)\mathbf{p}^T(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + 2 \cdot 0,04 \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix} \cdot [1 \ 1] = \begin{bmatrix} 0,76 & -0,24 \\ -0,24 & 0,76 \end{bmatrix};$$

$$\mathbf{b}(1) = \mathbf{b}(0) + 2\alpha\mathbf{e}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \cdot 0,04 \cdot \begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} 0,76 \\ 0,76 \end{bmatrix}.$$

Друга ітерація:

$$\mathbf{y}(1) = \text{purelin}(\mathbf{W}(1)\mathbf{p}(1) + \mathbf{b}(1)) =$$

$$= \text{purelin}\left(\begin{bmatrix} 0,76 & -0,24 \\ -0,24 & 0,76 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0,76 \\ 0,76 \end{bmatrix}\right) = \begin{bmatrix} 1,04 \\ 1,04 \end{bmatrix};$$

$$\mathbf{e}(1) = \mathbf{t}(1) - \mathbf{y}(1) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1,04 \\ 1,04 \end{bmatrix} = \begin{bmatrix} -2,04 \\ -3,04 \end{bmatrix};$$

$$\begin{aligned} \mathbf{W}(2) &= \mathbf{W}(1) + 2\alpha\mathbf{e}(1)\mathbf{p}^T(1) = \begin{bmatrix} 0,76 & -0,24 \\ -0,24 & 0,76 \end{bmatrix} + \\ &+ 2 \cdot 0,04 \cdot \begin{bmatrix} -2,04 \\ -3,04 \end{bmatrix} \cdot [1 \ 2] = \begin{bmatrix} 0,5968 & -0,5664 \\ -0,4832 & 0,2736 \end{bmatrix}; \end{aligned}$$



$$\mathbf{b}(2) = \mathbf{b}(1) + 2\alpha\mathbf{e}(1) = \begin{bmatrix} 0,76 \\ 0,76 \end{bmatrix} + 2 \cdot 0,04 \cdot \begin{bmatrix} -2,04 \\ -3,04 \end{bmatrix} = \begin{bmatrix} 0,5968 \\ 0,5168 \end{bmatrix}.$$

Якщо продовжити виконувати цей алгоритм, одержимо (рис. 6.16):

$$\mathbf{W}(\infty) = \begin{bmatrix} -0,5948 & -0,0523 \\ 0,1667 & -0,6667 \end{bmatrix}; \mathbf{b}(\infty) = \begin{bmatrix} 0,0131 \\ 0,1667 \end{bmatrix}.$$

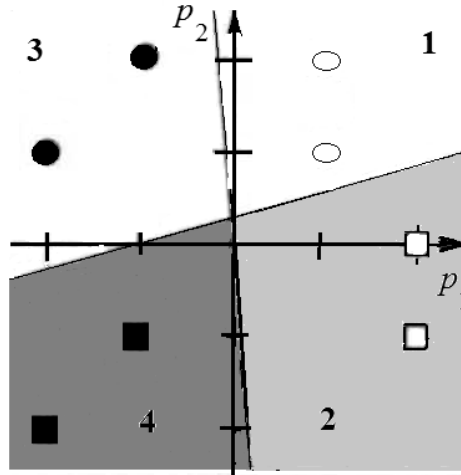


Рис. 6.16. Вхідні вектори та граничний розв'язок

### Контрольні запитання

1. Опишіть одношарову мережу АДАЛПН та її особливості.
2. Наведіть приклад одношарової мережі АДАЛПН. Який вигляд при цьому має функція середньоквадратичної похибки?
3. Опишіть алгоритм LMS.
4. Опишіть процес фільтрації та прогнозування на основі мережі АДАЛПН.

### Задачі для самостійного розв'язання

**Задача 6.1.** Задано адаптивний фільтр АДАЛПН (рис. 6.17), в якому:  $w_{11} = 1$ ;  $w_{12} = -4$ ;  $w_{13} = 2$ . Послідовність входів фільтра має вигляд  $\{p_i\} = \{\dots, 0, 0, 0, 1, 1, 2, 0, 0, \dots\}$ , де  $p_0 = 1$ ,  $p_1 = 1, \dots$ .

Визначити послідовність виходів фільтра  $\{y(k)\}$ .

**Задача 6.2.** На рис. 6.18 зображено два класи образів.

Використавши алгоритм LMS, навчити мережу АДАЛПН розрізняти образи класів 1 та 2 (мережа повинна ідентифікувати горизонтальні та вертикальні лінії). Пояснити, чому мережа АДАЛПН має труднощі при розв'язанні цієї задачі.

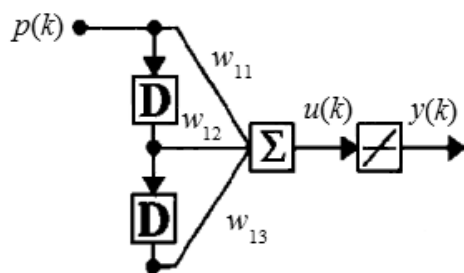


Рис. 6.17. Адаптивний фільтр АДАЛІН:  
 $y(k) = \text{purelin}(\mathbf{w}^T \mathbf{p}(k) + b)$

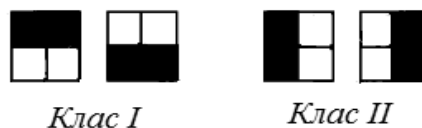


Рис. 6.18. Задача класифікації образів

**Задача 6.3.** Нехай задано такі пари вхід–ціль:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}$ ;  $\left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1 \right\}$ . Задані вектори подаються на вхід мережі АДАЛІН з різною ймовірністю: ймовірність потрапляння на вхід вектора  $\mathbf{p}_1 = 0,75$ , а вектора  $\mathbf{p}_2 = 0,25$ .

Визначити функціонал середньоквадратичної похибки та максимально стійке значення коефіцієнта швидкості навчання  $\alpha$ .

**Задача 6.4.** Нехай задано такі пари вхід–ціль:  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1 \right\}$ ;  $\left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = -1 \right\}$ . Задані вектори подаються на вхід з однаковою ймовірністю і використовуються для навчання мережі АДАЛІН без зсуву.

**I.** Визначити середньоквадратичну похибку та максимально стійке значення коефіцієнта швидкості навчання  $\alpha$ .

**II.** Написати програму (у вигляді *m*-файла) реалізації алгоритму LMS. Виконати 40 кроків алгоритму з використанням стійкого значення коефіцієнта швидкості навчання. Як початкове припущення використати нульовий вектор. Нарисувати графік кінцевого граничного розв'язку.

**III.** Виконати 40 кроків алгоритму LMS, встановивши початкові значення обох параметрів рівними одиниці. Накреслити графік кінцевого граничного розв'язку.

**IV.** Порівняти значення параметрів мережі, одержані в п. II і III.

**Задача 6.5.** Нехай задано такі пари вхід–ціль:  $\left\{ \mathbf{p}_1 = [1 \ 1]^T, t_1 = 1 \right\}$ ;  $\left\{ \mathbf{p}_2 = [1 \ -1]^T, t_2 = -1 \right\}$ . Задані вектори подаються на вхід з однаковою

ймовірністю і використовуються для навчання мережі АДАЛПН зі зсувом, яка має такі параметри:  $w_{11}$ ,  $w_{12}$  і  $b$ .

**I.** Визначити середньоквадратичну похибку та максимально стійке значення коефіцієнта швидкості навчання  $\alpha$ .

**II.** Написати програму реалізації алгоритму LMS (у вигляді  $m$ -файлу). Виконати 40 кроків алгоритму з використанням стійкого значення коефіцієнта швидкості навчання. Як початкове припущення – значення параметрів мережі – використати вектори спочатку з нульовими елементами, а потім з елементами, рівними одиниці. Накреслити відповідні графіки кінцевого граничного розв’язку.

**III.** Порівняти кінцеві значення параметрів мережі та граничні розв’язки, одержані в п. II.

**Задача 6.6.** Розглянемо адаптивний фільтр АДАЛПН (рис. 6.13), який прогнозує наступне значення вхідного сигналу на основі двох попередніх його значень. Відомо, що вхідний сигнал  $v(k)$  – це стаціонарний випадковий процес із функцією автокореляції вигляду  $C_v(n) = E[v(k)v(k+n)]$ .

**I.** Визначити середньоквадратичну похибку в термінах  $C_v(n)$  та за  $v(k) = \sin\left(\frac{k\pi}{5}\right)$ .

**II.** Визначити власні значення, власні вектори матриці Гессе (максимальне стійке значення швидкості навчання  $\alpha$ ), мінімальну точку середньоквадратичної похибки. Побудувати приблизний контурний графік середньоквадратичної похибки.

**III.** Виконати три кроки алгоритму LMS, використовуючи постійне значення  $\alpha$  та нульовий вектор як початкове припущення. Написати програму реалізації алгоритму LMS.

## Розділ 7. БАГАТОШАРОВИЙ ПЕРСЕПТРОН

### 7.1. Можливості багатошарового персептрона

Правило навчання персептрона Ф. Розенблатта та алгоритм LSM Б. Відроу і М. Хоффа були спроектовані для навчання одношарового персептрона, який здатен вирішити тільки лінійно подільні задачі класифікації. Як Розенблатт, так і Відроу запропонували використовувати багатошарові мережі, здатні подолати цей недолік, але вони не змогли узагальнити свої алгоритми навчання. У середині 1980-х років незалежно один від одного ряд вчених (серед них Девід Румельхарт, Джефрі Е. Хінт і Рональд Дж. Вільямс) запропонували алгоритм зворотного поширення (метод навчання БП). Цей алгоритм був популяризований завдяки книзі психологів Д. Румельхарта і Д. Маккелланда «Паралельно розподілена обробка».

У наш час алгоритм зворотного поширення для БП широко використовують на практиці. Його можна розглядати як реалізацію рекурсивної технології, яку в статистиці називають стохастичною апроксимацією. Діаграму тришарового персептрона, який одержано в результаті об'єднання трьох мереж одношарового персептрона, зображено на рис. 1.9. При цьому вихід першої мережі – це вхід другої мережі, вихід другої мережі – це вхід третьої мережі. Кожен шар може мати різний набір нейронів та різні функції активації.

Розглянемо можливості багатошарових мереж щодо класифікації образів та апроксимації функцій. Для ідентифікації структури тришарової мережі будемо використовувати такий запис:  $R - S^1 - S^2 - S^3$ , де  $R$  – кількість входів, а  $S^i$  – кількість нейронів в  $i$ -му шарі.

**Класифікація образів.** Для ілюстрації можливостей використання БП для класифікації образів розглянемо схему, зображену на рис. 7.1, а. Існують різні способи реалізації функції виключна диз'юнкція на основі багатошарових мереж, один із них – двошаровий персептрон із структурою 2-2-1 (рис. 7.1, б) з такими параметрами:  $\mathbf{W}^1 = \begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix}$ ;  $\mathbf{b}^1 = \begin{bmatrix} -1 \\ 1,5 \end{bmatrix}$ ;

$\mathbf{w}^2 = [1 \ 1]^T$ ;  $b^2 = -1,5$ .

Розглянемо іншу задачу (рис. 7.2, а): процедуру побудови цієї мережі можна використати для розв'язання задач класифікації з довільною кількістю границь поділу. При цьому перший шар використовують для формування необхідної кількості лінійних граничних розв'язків, які можуть бути об'єднані нейронами другого шару за допомогою логічної

операції «І» та нейронами третього шару за допомогою логічної операції «АБО». Існують різні способи вирішення цієї задачі на основі багатшарових мереж, один із них – тришаровий персептрон із структурою 2-4-2-1. Перша підмережа БП (рис. 7.2, б) виконує такі дії: перший нейрон першого шару визначає значення першого граничного розв’язку:  $y_1^1 = \text{hardlim}([-1 \ 0]\mathbf{p} + 0,5)$ , другий нейрон першого шару – значення другого граничного розв’язку:  $y_2^1 = \text{hardlim}([0 \ -1]\mathbf{p} + 0,75)$ , третій нейрон першого шару – значення третього граничного розв’язку:  $y_3^1 = \text{hardlim}([1 \ 0]\mathbf{p} - 1,5)$ , четвертий нейрон першого шару – значення четвертого граничного розв’язку:  $y_4^1 = \text{hardlim}([0 \ 1]\mathbf{p} - 0,25)$ . Тришаровий персептрон (рис. 7.2, б) має такі параметри:

$$\mathbf{W}^1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix};$$

$$\mathbf{b}^1 = \begin{bmatrix} 0,5 \\ 0,75 \\ -1,5 \\ -0,25 \end{bmatrix}; \quad \mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}; \quad \mathbf{b}^2 = \begin{bmatrix} -1,5 \\ -1,5 \end{bmatrix}; \quad \mathbf{w}^3 = [1 \ 1]^T; \quad \mathbf{b}^3 = -0,5.$$

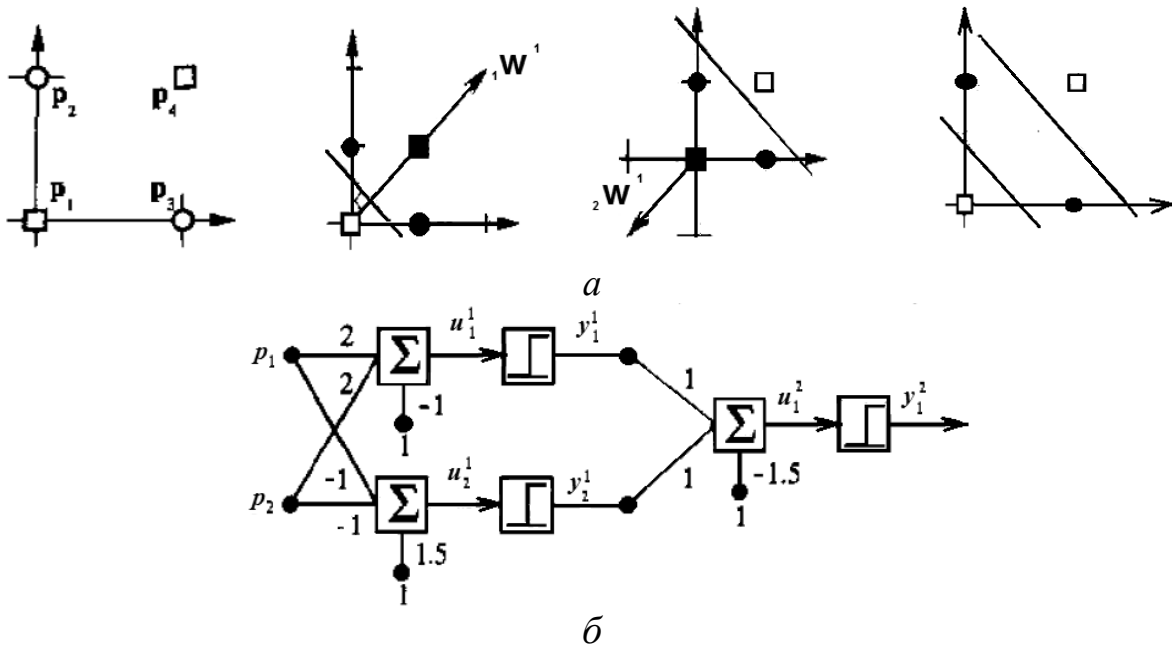


Рис. 7.1. Вхідні дані (логічна функція виключна диз’юнкція), лінійні граничні розв’язки й остаточний розв’язок (а); схема двошарового персептрона, що реалізує цю функцію (б)

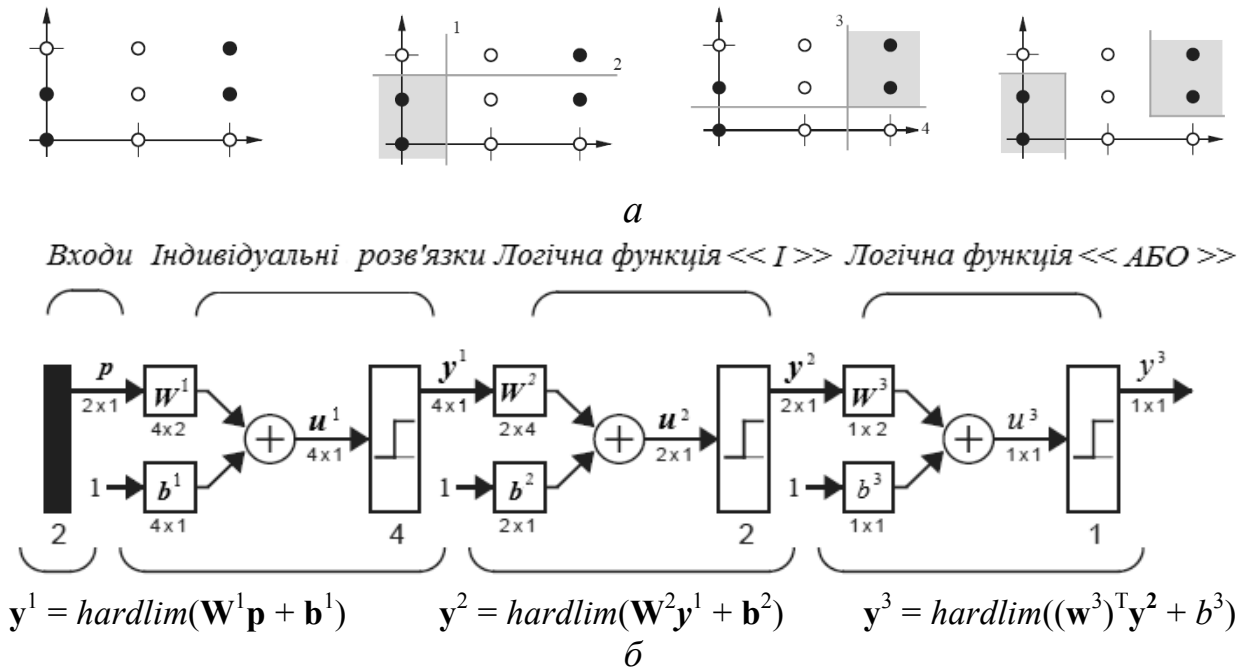


Рис. 7.2. Вхідні дані, границі поділу та остаточний розв'язок (а) і загальна схема (б) тришарового персептрона, що класифікує дані

**Апроксимація функцій.** У системах контролю зазвичай постає проблема визначення функції апроксимації, яка встановлює відповідність між входом і виходом. У розглянутому прикл. 7.1 проілюстровано ефективність застосування БП для апроксимації функції  $y = g(p)$  на основі двошарової мережі зі структурою 1-2-1, ФА якої мають вигляд

$$f^1(u) = \text{logsig}(u) = \frac{1}{1 + e^{-u}} \quad \text{та} \quad f^2(u) = \text{purelin}(u) = u.$$

Двошарові мережі із сигмоїдною ФА у прихованому шарі та лінійною ФА у вихідному шарі можуть апроксимувати будь-яку функцію з будь-якою точністю.

Розглянемо алгоритм навчання БП.

## 7.2. Алгоритм зворотного поширення похибки

Для багатошарових мереж вихід попереднього шару є входом наступного:

$$y^{m+1} = f^{m+1}(W^{m+1} y^m + b^{m+1}), \quad m = 0, 1, \dots, M-1, \quad (7.1)$$

де  $M$  – кількість шарів мережі. Нейрони першого шару отримують зовнішні входи:  $y^0 = p$ . Вихід нейронів останнього шару вважають виходом мережі:  $y = y^M$ .

**Показник ефективності функціонування мережі.** Алгоритм зворотного поширення похибки для багатошарових мереж – це узагальнення

алгоритму LMS. Обидва алгоритми використовують показник ефективності функціонування мережі – середньоквадратичну похибку – та вимагають наявності множини прикладів навчання мережі:  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , де  $\mathbf{p}_i$  – вхід мережі,  $\mathbf{t}_i$  – цільове значення. Алгоритм налаштовує параметри мережі таким чином, щоб мінімізувати середньоквадратичну похибку:

$F(\mathbf{x}) = E[e^2] = E[(t - y)^2]$ , якщо мережа має один вихід. Якщо мережа має множину виходів, це рівняння узагальнюється до вигляду  $F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{y})^T (\mathbf{t} - \mathbf{y})]$ . Як і в алгоритмі LMS, середньоквадратична

похибка визначається за формулою  $\bar{F}(\mathbf{x}) = \mathbf{e}^T(k) \mathbf{e}(k) = (\mathbf{t}(k) - \mathbf{y}(k))^T (\mathbf{t}(k) - \mathbf{y}(k))$ .

Алгоритм найшвидшого спуску для обчислення середньоквадратичної похибки зі швидкістю навчання  $\alpha$  має вигляд

$$w_{ij}^m(k+1) = w_{ij}^m(k) - \alpha \frac{\partial \bar{F}}{\partial w_{ij}^m}; \quad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \bar{F}}{\partial b_i^m}. \quad (7.2)$$

**Правило ітерації.** Для багатошарової мережі похибка має вигляд неявної вагової функції у прихованому шарі, тому часткові похідні обчислити не так легко. У цьому випадку будемо використовувати рекурентне правило обчислення похідних.

Припустимо, що функція  $f$  явно залежить від змінної  $u$ . Необхідно визначити похідну функції  $f$  за другою змінною  $w$ . Одержимо таке рекурентне правило:  $\frac{df(u(w))}{dw} = \frac{df(u)}{du} \cdot \frac{du(w)}{dw}$ . Використаємо це правило щоб знайти похідні в (7.2):

$$\frac{\partial \bar{F}}{\partial w_{ij}^m} = \frac{\partial \bar{F}}{\partial u_i^m} \cdot \frac{\partial u_i^m}{\partial w_{ij}^m}; \quad \frac{\partial \bar{F}}{\partial b_i^m} = \frac{\partial \bar{F}}{\partial u_i^m} \cdot \frac{\partial u_i^m}{\partial b_i^m}. \quad (7.3)$$

Оскільки  $u_i^m$  є явно заданою функцією від ваги та зсуву, то другий елемент у кожному із цих рівнянь можна легко обчислити:

$u_i^m = \sum_{j=1}^{S^{m-1}} w_{ij}^m y_j^{m-1} + b_i^m$ , маємо  $\frac{\partial u_i^m}{\partial w_{ij}^m} = y_j^{m-1}$ ;  $\frac{\partial u_i^m}{\partial b_i^m} = 1$ . Якщо позначити

через  $s_i^m = \frac{\partial \bar{F}}{\partial u_i^m}$  чутливість функції  $\bar{F}$  до зміни  $i$ -го елемента входу

$m$ -го шару, то рівняння (7.3) можна записати у вигляді  $\frac{\partial \bar{F}}{\partial w_{ij}^m} =$

$= s_i^m y_j^{m-1}$ ;  $\frac{\partial \bar{F}}{\partial b_i^m} = s_i^m$ . Отже, алгоритм найшвидшого спуску має вигляд

$w_{ij}^m(k+1) = w_{ij}^m(k) - \alpha s_i^m y_j^{m-1}$ ;  $b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$ , або в матричній формі:  $\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{y}^{m-1})^T$ ,  $\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$ , де

$$\mathbf{s}^m = \frac{\partial \bar{F}}{\partial \mathbf{u}^m} = \left[ \frac{\partial \bar{F}}{\partial u_1^m} \cdots \frac{\partial \bar{F}}{\partial u_{s^m}^m} \right]^T.$$

### 7.3. Матриця Якобі. Чутливість мереж зі зворотним зв'язком

Розрахунок значення  $\mathbf{s}^m$  вимагає визначення рекурентного правила, яке описує співвідношення між значеннями чутливості  $m$ -го і  $(m+1)$ -го шарів. Для цього використаємо матрицю Якобі вигляду  $\frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} =$

$$= \begin{bmatrix} \frac{\partial u_1^{m+1}}{\partial u_1^m} & \cdots & \frac{\partial u_1^{m+1}}{\partial u_{s^m}^m} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_{s^{m+1}}^{m+1}}{\partial u_1^m} & \cdots & \frac{\partial u_{s^{m+1}}^{m+1}}{\partial u_{s^m}^m} \end{bmatrix}. \text{ Визначимо } (i, j)\text{-й елемент цієї матриці:}$$

$$\frac{\partial u_i^{m+1}}{\partial u_j^m} = \frac{\partial \left( \sum_{l=1}^{s^m} w_{il}^{m+1} y_j^m + b_i^{m+1} \right)}{\partial u_j^m} = w_{ij}^{m+1} \frac{\partial y_j^m}{\partial u_j^m} = w_{ij}^{m+1} \frac{\partial f^m(u_j^m)}{\partial u_j^m} = w_{ij}^{m+1} \dot{f}^m(u_j^m),$$

де  $\dot{f}^m(u_j^m) = \frac{\partial f^m(u_j^m)}{\partial u_j^m}$ . Тому матрицю Якобі можна записати у вигляді

$$\frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{u}^m), \text{ де } \dot{\mathbf{F}}^m(\mathbf{u}^m) = \begin{bmatrix} \dot{f}^m(u_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(u_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(u_{s^m}^m) \end{bmatrix}.$$

Таким чином, маємо:

$$\begin{aligned} \mathbf{s}^m &= \frac{\partial \bar{F}}{\partial \mathbf{u}^m} = \left( \frac{\partial \mathbf{u}^{m+1}}{\partial \mathbf{u}^m} \right)^T \frac{\partial \bar{F}}{\partial \mathbf{u}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{u}^m) (\mathbf{W}^{m+1})^T \frac{\partial \bar{F}}{\partial \mathbf{u}^{m+1}} = \\ &= \dot{\mathbf{F}}^m(\mathbf{u}^m) (\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}. \end{aligned} \quad (7.4)$$



Одержане рівняння показує, чому алгоритм зворотного зв'язку (або зворотного поширення похибки) має таку назву: чутливість поширюється назад від останнього шару до першого:  $\mathbf{s}^M \rightarrow \dots \rightarrow \mathbf{s}^1$ . Варто зазначити, що алгоритм зворотного поширення похибки використовує той самий алгоритм найшвидшого спуску, як і алгоритм LMS. Єдиною складністю є обчислення градієнта для визначення першого значення чутливості. Виконання алгоритму зворотного поширення похибки вимагає визначення початкового значення  $\mathbf{s}^M$  для використання рекурентного співвідношення (7.4).

У вихідного шару одержимо:

$$s_i^M = \frac{\partial \bar{F}}{\partial u_i^M} = \frac{\partial (t - y)^T (t - y)}{\partial u_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - y_i)^2}{\partial u_i^M} = -2(t_i - y_i) \frac{\partial y_i}{\partial u_i^M}. \quad (7.5)$$

Коефіцієнт чутливості – це похідна від квадрату похибки по відношенню до входів мережі. Оскільки  $\frac{\partial y_i}{\partial u_i^M} = \frac{\partial y_i^M}{\partial u_i^M} = \frac{\partial f^M(u_i^M)}{\partial u_i^M} = \dot{f}^M(u_i^M)$ , то можна записати:  $s_i^M = -2(t_i - y_i) \dot{f}^M(u_i^M)$ , або в матричній формі:  $\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{u}^M)(\mathbf{t} - \mathbf{y})$ .

Отже, **алгоритм зворотного поширення** складається з таких етапів.

*Крок 1. Ініціалізація ваги.* Зазвичай початкове значення ваги (як і зсуву) дорівнює випадковим числам, рівномірно розподіленим на відрізках  $[-1/2R, 1/2R]$  для першого шару та  $[-1/2S^i, 1/2S^i]$  для  $(i + 1)$ -го шару (де  $\mathbf{p} \in \mathbb{R}^R$ ,  $S^i$  – кількість нейронів у  $i$ -му шарі). Зсув використовують як значення ваги, що відповідає з'єднанню з фіктивним входом, який завжди дорівнює одиниці.

*Крок 2. Прямий прохід.* На входи всіх шарів мережі подаються відповідні значення:  $\mathbf{y}^0 = \mathbf{p}$ ;  $\mathbf{y}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{y}^m + \mathbf{b}^{m+1})$ ,  $m = 0, 1, \dots, M-1$ ;  $\mathbf{y} = \mathbf{y}^M$ .

*Крок 3. Зворотний прохід.* Зворотне поширення похибки вздовж всієї мережі у вигляді чутливості функції мережі  $F(\mathbf{x})$  до змін елементів вагового вектора:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{u}^M)(\mathbf{t} - \mathbf{y}); \quad \mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{u}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, \quad m = M-1, \dots, 2, 1.$$

*Крок 4. Налаштування значень ваги та зсуву за допомогою правила найшвидшого спуску вигляду*

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{y}^{m-1})^T; \quad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m. \quad (7.6)$$

*Крок 5. Перевірка виконання критерію зупинки алгоритму:* значення  $F(\mathbf{x})$ , де  $\mathbf{x} = [\mathbf{w} \ \mathbf{b}]^T$ , не змінюється на множині навчання (або різниця між виходом мережі та бажаним виходом  $t$  на всій множині навчання досягає деякого прийняттого значення). Якщо критерій виконується, то виконання алгоритму закінчують, інакше змінюють значення входу  $\mathbf{p}$  і переходять до кроку 2.

Якщо мережа не збігається до розв'язку, то може бути потрібна більша або менша кількість нейронів прихованого шару. Вага деяких нейронів рідко змінюється, тому ці зв'язки (або нейрони) можуть не брати участі у процесі навчання – їх можна видалити, і кількість прихованих нейронів виявиться значно меншою.

Щодо даних навчання, то немає єдиного визначення (яке стосується усіх випадків) поняття «достатня і правильна множина навчання» для мереж зворотного поширення. У загальному випадку для навчання мережі можна використовувати стільки даних, скільки є в наявності, хоча не потрібно використовувати їх усі. Щоб успішно навчити мережу, часто достатньо невеликої підмножини наявних даних. Інші дані можуть бути використані для тестування мережі: процесу перевірки, чи виконує мережа потрібне відображення на вхідних векторах, з якими вона ніколи не працює під час навчання. Переконайтеся в тому, що навчальні дані охоплюють весь очікуваний вхідний простір. У процесі навчання вибір пари із множини навчання виконується випадковим чином (не потрібно навчати мережу повністю на вхідних векторах одного класу, а потім переходити на елементи іншого класу: мережа буде забувати оригінальність навчання). Якщо ФА вихідного шару є сигмоїдною, то потрібно масштабувати вихідні значення. У цьому випадку виходи мережі ніколи не будуть дорівнювати нулю або одиниці. Тому для зображення найменших і найбільших вихідних значень потрібно використовувати такі значення, як 0,1 і 0,9.

Мережа зворотного поширення добре виконує узагальнення. Під узагальненням розумітимемо те, що, враховуючи декілька вхідних векторів, які належать до одного класу, мережа визначає подібність між вхідними векторами. Непотрібні дані будуть проігноровані. Наприклад, якщо треба навчити мережу визначати, чи є бінарне число завдовжки 5 бітів парним або непарним, то тільки невелика множина прикладів із множини навчання буде коригувати значення ваги, так що класифікація буде здійснюватися винятково на основі значення наймолодшого біта числа (мережа навчається ігнорувати непотрібні дані в інших бітах).

**Приклад 7.1.** Щоб проілюструвати роботу алгоритму зворотного поширення, виберемо мережу зі структурою 1-2-1 (рис. 7.3) і застосуємо

її для вирішення завдання апроксимації функції. Нехай задано функцію  $g(p) = 1 + \sin\left(\frac{\pi}{4}p\right)$ ,  $-2 \leq p \leq 2$ . Виберемо початкові значення ваги і зсуву (у загальному випадку вага та зсув набувають невеликих випадкових початкових значень):  $\mathbf{w}^1(0) = \begin{bmatrix} -0,27 \\ -0,41 \end{bmatrix}$ ;  $\mathbf{b}^1(0) = \begin{bmatrix} -0,48 \\ -0,13 \end{bmatrix}$ ;  $\mathbf{w}^2(0) = \begin{bmatrix} 0,09 \\ -0,17 \end{bmatrix}$ .  $b^2(0) = 0,48$ . Відповідь мережі для вказаних початкових значень разом із графіком функції  $g(p)$ , яку необхідно апроксимувати, зображено на рис. 7.4. Виконання алгоритму розпочнемо з початковим значенням входу  $p = 1$ , маємо:  $y^0 = p = 1$ .

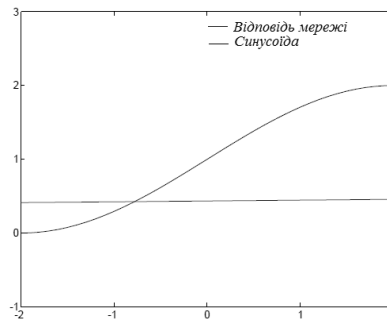
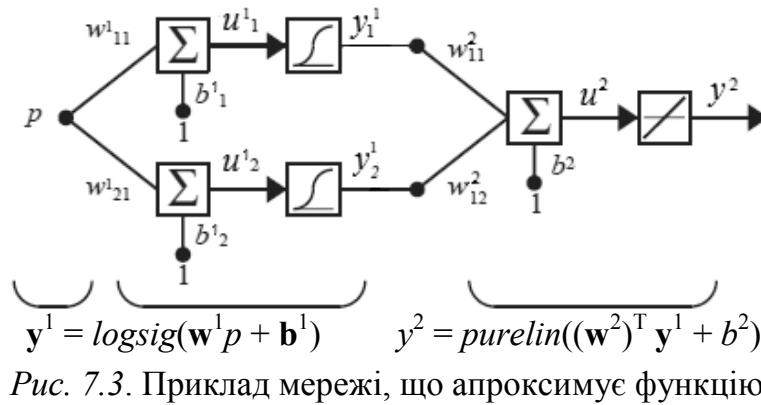


Рис. 7.4. Початкова відповідь мережі (залежність  $y^2$  від  $p$ )

Виконаємо прямий прохід. Вихід першого шару:

$$\begin{aligned}
 \mathbf{y}^1 &= \mathbf{f}^1(\mathbf{w}^1 y^0 + \mathbf{b}^1) = \text{logsig}\left(\begin{bmatrix} -0,27 \\ -0,41 \end{bmatrix} \cdot 1 + \begin{bmatrix} -0,48 \\ -0,13 \end{bmatrix}\right) = \\
 &= \text{logsig}\left(\begin{bmatrix} -0,75 \\ -0,54 \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{1 + e^{0,75}} \\ \frac{1}{1 + e^{0,54}} \end{bmatrix} = \begin{bmatrix} 0,321 \\ 0,368 \end{bmatrix}.
 \end{aligned}$$

Вихід другого шару:

$$y^2 = f^2 \left( (\mathbf{w}^2)^T y^1 + b^2 \right) = \text{purelin} \left( [0,09 \quad -0,17] \cdot \begin{bmatrix} 0,321 \\ 0,368 \end{bmatrix} + 0,48 \right) = 0,446.$$

Відповідна похибка має вигляд

$$e = t - y = \left\{ 1 + \sin \left( \frac{\pi}{4} p \right) \right\} - y^2 = \left\{ 1 + \sin \left( \frac{\pi}{4} \cdot 1 \right) \right\} - 0,446 = 1,261.$$

*Виконаємо зворотний прохід.* Для обчислення коефіцієнта чутливості зворотного зв'язку потрібні похідні функцій активації  $f^1(u)$  і  $f^2(u)$ . Для першого шару маємо

$$\frac{d}{du} f^1(u) = \frac{d}{du} \left( \frac{1}{1 + e^{-u}} \right) = \frac{e^{-u}}{(1 + e^{-u})^2} = \left( 1 - \frac{1}{1 + e^{-u}} \right) \left( \frac{1}{1 + e^{-u}} \right) = (1 - y^1)(y^1),$$

і для другого шару  $\frac{d}{du} f^2(u) = \frac{d}{du}(u) = 1.$

Використавши рівняння (7.5), для другого шару маємо значення чутливості функції мережі  $F(\mathbf{x})$  до змін елементів вагового вектора:  $s^2 = -2 \dot{f}^2(u^2)(t - y) = -2 \cdot 1 \cdot 1,261 = -2,522.$

Використавши рівняння (7.4), одержимо значення чутливості для першого шару:

$$\begin{aligned} \mathbf{s}^1 &= \dot{\mathbf{f}}^1(\mathbf{u}^1)(\mathbf{w}^2)^T s^2 = \begin{bmatrix} (1 - y_1^1)(y_1^1) & 0 \\ 0 & (1 - y_2^1)(y_2^1) \end{bmatrix} \cdot \begin{bmatrix} 0,09 \\ -0,17 \end{bmatrix} \cdot (-2,522) = \\ &= \begin{bmatrix} (1 - 0,321)(0,321) & 0 \\ 0 & (1 - 0,368)(0,368) \end{bmatrix} \cdot \begin{bmatrix} 0,09 \\ -0,17 \end{bmatrix} \cdot (-2,522) = \begin{bmatrix} -0,0495 \\ 0,0997 \end{bmatrix}. \end{aligned}$$

*Виконаємо налаштування ваги та зсуву для  $\alpha = 0,1$ .* Із рівнянь (7.6) маємо:

$$\begin{aligned} \mathbf{w}^2(1) &= \mathbf{w}^2(0) - \alpha s^2 \mathbf{y}^1 = [0,09 \quad -0,17]^T - 0,1 \cdot (-2,522) \cdot [0,321 \quad 0,368]^T = \\ &= [0,171 \quad -0,0772]^T; \end{aligned}$$

$$b^2(1) = b^2(0) - \alpha s^2 = 0,48 - 0,1 \cdot (-2,522) = 0,732,$$

$$\mathbf{w}^1(1) = \mathbf{w}^1(0) - \alpha s^1 \mathbf{y}^0 = \begin{bmatrix} -0,27 \\ -0,41 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} -0,0495 \\ 0,0997 \end{bmatrix} \cdot 1 = \begin{bmatrix} -0,265 \\ -0,420 \end{bmatrix};$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha \mathbf{s}^1 = \begin{bmatrix} -0,48 \\ -0,13 \end{bmatrix} - 0,1 \cdot \begin{bmatrix} -0,0495 \\ 0,0997 \end{bmatrix} = \begin{bmatrix} -0,475 \\ -0,140 \end{bmatrix}.$$

На цьому завершується перша ітерація алгоритму зворотного поширення. Потім вибирають інше значення входу  $p$  й алгоритм повторюється доти, доки різниця між виходом мережі та цільовою функцією не досягне деякого прийнятного значення на множині прикладів навчання.

#### 7.4. Багатошаровий персептрон з одним прихованим шаром як універсальний апроксиматор

Середньоквадратична похибка для багатошарових мереж набагато складніша, ніж для одношарових, і має багато локальних мінімумів. Якщо алгоритм зворотного поширення збігається, то це ще не означає, що отриманий розв'язок є оптимальним. Щоб гарантувати оптимальність отриманого розв'язку, необхідно виконати декілька спроб з різними початковими умовами. У більшості випадків багатошарова мережа пов'язується із множиною прикладів її навчання:  $\{p_1, t_1\}, \dots, \{p_Q, t_Q\}$ , яка, зазвичай, подана великою кількістю можливих пар вхід–ціль. Важливо, щоб мережа вміла узагальнювати одержані дані. Наприклад, припустимо, що множину навчання отримали для апроксимації функції  $g(p) = 1 + \sin\left(\frac{\pi}{4}p\right)$  у точках

$p = -2, -1,6, -1,2, \dots, 1,6, 2$  (разом 11 пар вхід–ціль). На рис. 7.5, а зображено відповідь мережі зі структурою 1-2-1, яка була навчена на цих даних (відповідь точно відтворює функцію  $g(p)$ : товста лінія – функцію  $g(p)$ , тонка – відповідь мережі, символи «+» – множину навчання). Відповідь мережі зі структурою 1-9-1 (рис. 7.5, б) точно моделює значення функції  $g(p)$  у точках навчання. Проте у точках, які не належать множині навчання, мережа може одержати вихід, далекий від значення  $g(p)$ .

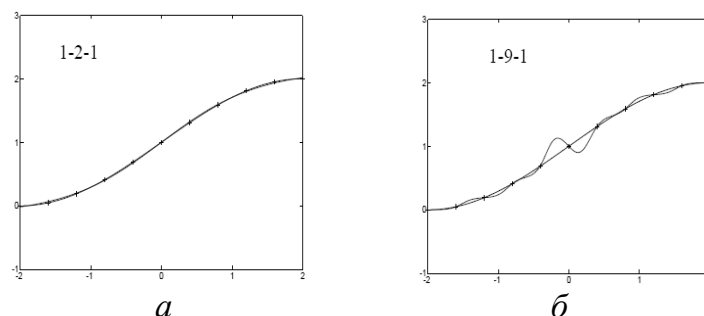


Рис. 7.5. Апроксимації функції  $g(p)$  мережею зі структурою 1-2-1 (а), та зі структурою 1-9-1 (б)

**Теорема про універсальну апроксимацію для нелінійного відображення вхід–вихід.** Багатошаровий персептрон, який навчається за алгоритмом зворотного поширення, можна розглядати як загальний механізм

реалізації нелінійного відображення вхід–вихід [12]. Наприклад, якщо  $R$  – кількість входів багатошарового персептрона, а  $M$  – кількість нейронів вихідного шару, то відношення вхід–вихід для такої мережі визначає неперервно диференційоване нескінченну кількість разів відображення  $\mathfrak{R}^R \rightarrow \mathfrak{R}^M$  ( $R$ -вимірний простору Евкліда вхідних даних у  $M$ -вимірний простір Евкліда вихідних даних), якщо цю умову задовольняють і ФА. Запитання «Яку мінімальну кількість прихованих шарів має БП, який забезпечує апроксимацію деякого неперервного відображення?» пояснює теорема про універсальну апроксимацію для нелінійного відображення вхід–вихід. Вона є природним розширенням теореми Вейерштрасса, яка стверджує, що будь-яка неперервна функція на замкненому інтервалі дійсної осі може бути зображена рядом поліномів, який збігається абсолютно й рівномірно.

*Теорема про універсальну апроксимацію.* Нехай  $\varphi(\cdot)$  – обмежена, монотонно зростаюча неперервна функція,  $I_R$  –  $R$ -вимірний одинарний гіперкуб  $[0, 1]^R$ ,  $C(I_R)$  – простір неперервних на  $I_R$  функцій. Тоді для будь-якої функції  $f \in C(I_R)$  і  $\varepsilon > 0$  існують такі ціле число  $S^1$  і множина дійсних констант  $\alpha_i, b_i, w_{ij}$  ( $i = 1, \dots, S^1; j = 1, \dots, R$ ), що

$$F(p_1, \dots, p_R) = \sum_{i=1}^{S^1} \alpha_i \varphi \left( \sum_{j=1}^R w_{ij} p_j + b_i \right) \quad (7.7)$$

є реалізацією апроксимації функції  $f(\cdot)$ , тобто  $|F(p_1, \dots, p_R) - f(p_1, \dots, p_R)| < \varepsilon$  для всіх  $(p_1, \dots, p_R)$ , які належать вхідному простору.

Теорему про універсальну апроксимацію безпосередньо можна використати до БП:

1) в його моделі як ФА використовується обмежена, монотонно зростаюча логістична функція *logsig*, яка задовольняє умови теореми щодо функції  $\varphi(\cdot)$ ;

2) вираз (7.7) описує вихідний сигнал персептрона такого вигляду:

а) мережа містить  $R$  входів  $(p_1, \dots, p_R)$  та один прихований шар, який складається із  $S^1$  нейронів;

б) прихований  $i$ -й нейрон має синаптичну вагу  $(w_{i1}, \dots, w_{iR})$  і поріг  $b_i$ ;

в) вихід мережі зображує лінійну комбінацію вихідних сигналів прихованих нейронів, зважених синаптичною вагою вихідного нейрона –  $(\alpha_1, \dots, \alpha_{S^1})$ .

Теорема про універсальну апроксимацію є теоремою існування, тобто математичним доказом можливості апроксимації будь-якої неперервної функції. Вираз (7.7) узагальнює опис апроксимації функції скінченим рядом Фур'є. Теорема стверджує, що БП з одним прихованим

шаром є достатнім для побудови рівномірної апроксимації з точністю  $\epsilon$  для будь-якої множини навчання, яка має вигляд  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ , де  $\mathbf{t}_i = f(p_1^i, \dots, p_R^i)$ . Але з теореми не випливає, що один прихований шар є оптимальним у сенсі часу навчання, простоти реалізації та якості узагальнення. Отже, однією з основних проблем алгоритму зворотного поширення є тривалий час навчання. У розд. 8 розглянемо деякі методи підвищення ефективності його роботи.

### Приклади розв'язання задач\*

**Приклад 7.2.** Розглянемо дві групи образів (рис. 7.6): групу 1 зображено вертикальними лініями, групу 2 – горизонтальними.



Рис. 7.6. Образи до прикл. 7.2

**I.** Визначити, чи є ці групи образів лінійно подільними.

**II.** Описати БП, здатний класифікувати ці образи.

**Розв'язання. I.** Спочатку перетворимо образи у вектори-прототипи: кожний білий квадрат зобразимо числом «-1», а кожний чорний квадрат – числом «1». Тоді вектори-прототипи образів першої групи мають вигляд:  $\mathbf{p}_1 = [1 \ 1 \ -1 \ -1]^T$ ;  $\mathbf{p}_2 = [-1 \ -1 \ 1 \ 1]^T$ , а другої групи:  $\mathbf{p}_3 = [1 \ -1 \ 1 \ -1]^T$ ;  $\mathbf{p}_4 = [-1 \ 1 \ -1 \ 1]^T$ . Щоб ці групи образів були лінійно подільними, мають існувати вага  $\mathbf{W}$  та зсув  $b$  такі, що:  $\mathbf{W}\mathbf{p}_1 + b \geq 0$ ;  $\mathbf{W}\mathbf{p}_2 + b \geq 0$ ;  $\mathbf{W}\mathbf{p}_3 + b < 0$ ;  $\mathbf{W}\mathbf{p}_4 + b < 0$ , тобто:

$$w_{11} + w_{12} - w_{13} - w_{14} + b \geq 0; \quad w_{11} - w_{12} + w_{13} - w_{14} + b < 0;$$

$$-w_{11} - w_{12} + w_{13} + w_{14} + b \geq 0; \quad -w_{11} + w_{12} - w_{13} + w_{14} + b < 0.$$

Перші дві умови скорочуються до нерівностей вигляду  $\begin{cases} w_{11} + w_{12} + b \geq w_{13} + w_{14}; \\ w_{13} + w_{14} + b \geq w_{11} + w_{12}, \end{cases}$  які не є сумісними. Останні дві умови скорочуються до нерівностей  $\begin{cases} w_{11} + w_{13} + b > w_{12} + w_{14}; \\ w_{12} + w_{14} + b > w_{11} + w_{13}, \end{cases}$  які теж не є сумісними.

Тому не існує гіперплощини, яка може поділити задані дві групи образів.

\* Задачі взято з посібника [14].

II. Існують різні види БП, здатні розв'язати цю задачу. Спроекуємо мережу (рис. 7.7).

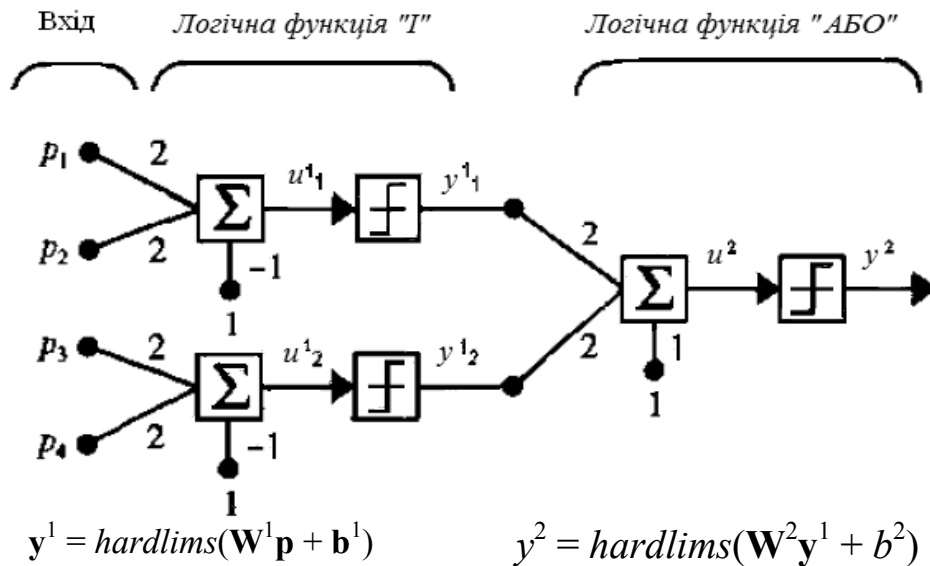


Рис. 7.7. Мережа, яка класифікує образи (до прикл. 7.2)

*Перший шар* її включає два нейрони, які виконують операцію «І» – перший нейрон досліджує перші два елементи вектора входу (якщо обидва елементи дорівнюють «1», він видає «1», інакше – «-1»); другий нейрон досліджує останні два елементи вектора входу таким самим чином. *Другий шар* мережі перевіряє, чи дорівнює один із виходів першого шару «1» (реалізуючи операцію «АБО»). Таким чином, мережа виведе «1», якщо перші (або останні) два елементи вхідного вектора одночасно дорівнюють одиниці.

**Приклад 7.3.** Задачу класифікації (світлі кола – образи класу I, а темні – образи класу II) зображено на рис. 7.8. Ці класи образів не є лінійно подільними. Спроекувати БП, здатний правильно класифікувати задані образи.

*Розв'язання.* Вирішуватимемо це завдання за допомогою тришарової мережі. У першому шарі визначимо набір з одинадцяти ліній, які відділяють образи класу I від образів класу II (рис. 7.9). Кожний стовпець вагової матриці першого шару відповідає одній границі поділу. Вагова матриця та вектор зсуву першого шару

$$\mathbf{W}^1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix};$$

$$\mathbf{b}^1 = [-2 \quad 3 \quad 0,5 \quad 0,5 \quad -1,75 \quad 2,25 \quad -3,25 \quad 3,75 \quad 6,25 \quad -5,75 \quad -4,75]^T.$$



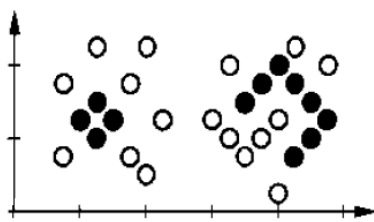


Рис. 7.8. Образи до прикл. 7.3

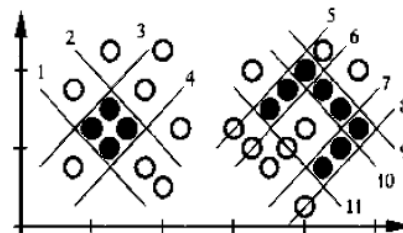


Рис. 7.9. Границі поділу першого шару

Можна об'єднати виходи 11 нейронів першого шару з нейронами другого шару, використавши функцію «І». Вагова матриця та вектор зсуву другого шару  $\mathbf{W}^2$  =

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}; \quad \mathbf{b}^2 = \begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix}.$$

Чотири границі поділу для другого шару зображено на рис. 7.10: для другого нейрона граничний розв'язок одержимо, об'єднавши границі 5, 6, 9 і 11 першого шару, які відповідають другому рядку вагової матриці  $\mathbf{W}^2$ . Використовуючи функцію «АБО», третій шар мережі об'єднує чотири області розв'язків другого шару мережі в одну область. Вага та зсув третього шару мають такий вигляд:  $\mathbf{w}^3 = [1 \ 1 \ 1 \ 1]^T$ ;  $b^3 = 3$ . Мережу, яка класифікує задані образи, зображено на рис. 7.11.

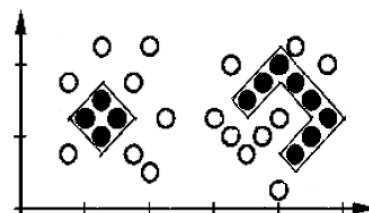
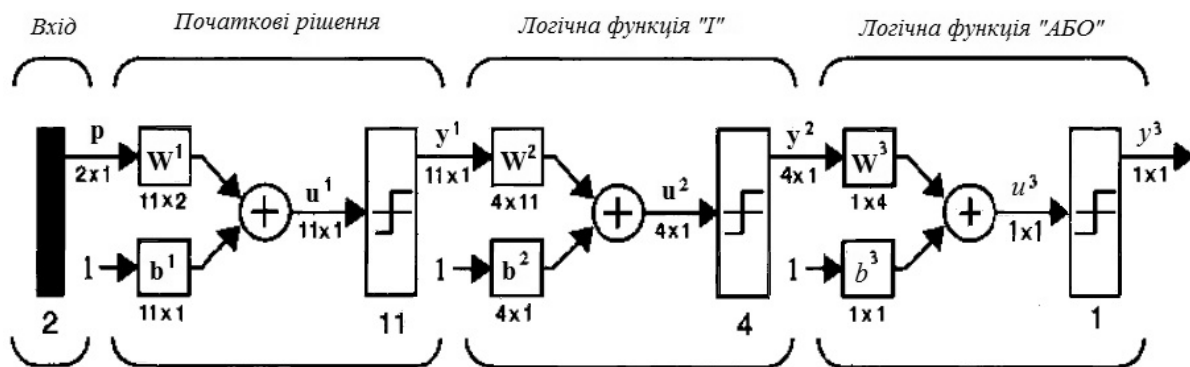


Рис. 7.10. Границі поділу другого шару



$$y^1 = \text{hardlims}(\mathbf{W}^1 \mathbf{p} + \mathbf{b}^1) \quad y^2 = \text{hardlims}(\mathbf{W}^2 \mathbf{y}^1 + \mathbf{b}^2) \quad y^3 = \text{hardlims}((\mathbf{w}^1)^T \mathbf{y} + b^3)$$

Рис. 7.11. Мережа, яка класифікує образи (до прикл. 7.3)

Якщо ввести в мережу будь-який вектор із затемнених областей, її виходом буде значення «1» (вектор класу 2), усім іншим вхідним векторам буде відповідати вихід «-1» (вектор класу 1).

**Приклад 7.4.** Довести, що багатошарова мережа з лінійною ФА еквівалентна одношаровій лінійній мережі.

*Розв'язання.* Для багатошарової мережі з лінійною ФА рівняння граничних розв'язків мають такий вигляд:

$$y^1 = W^1 p + b^1; \quad y^2 = W^2 y^1 + b^2 = W^2 W^1 p + [W^2 b^1 + b^2];$$

$$y^3 = W^3 y^2 + b^3 = W^3 W^2 W^1 p + [W^3 W^2 b^1 + W^3 b^2 + b^3].$$

Якщо продовжити ці розрахунки, то можна побачити, що  $M$ -шарова лінійна мережа із ФА *purelin* еквівалентна одношаровій лінійній мережі з такими значеннями ваги та зсуву:  $W = W^M W^{M-1} \dots W^1$ ;  $b = [W^M W^{M-1} \dots W^2] b^1 + [W^M W^{M-1} \dots W^3] b^2 + \dots + b^M$ .

**Приклад 7.5.** Розглянемо двошарову мережу (рис. 7.12), для якої початкові значення ваги та зсуву мають такий вигляд:  $w^1 = 1$ ;  $b^1 = 1$ ;  $w^2 = -2$ ;  $b^2 = 1$ . На вхід подається пара вхід-ціль  $\{p = 1, t = 1\}$ .

**I.** Визначити квадрат похибки  $(e)^2$  у вигляді явної функції від ваги та зсуву.

**II.** Використовуючи результат п. I, обчислити значення  $\partial(e)^2 / \partial w^1$  для початкових значень ваги та зсуву.

**III.** Виконати п. II, використовуючи алгоритм зворотного поширення. Порівняти отримані результати.

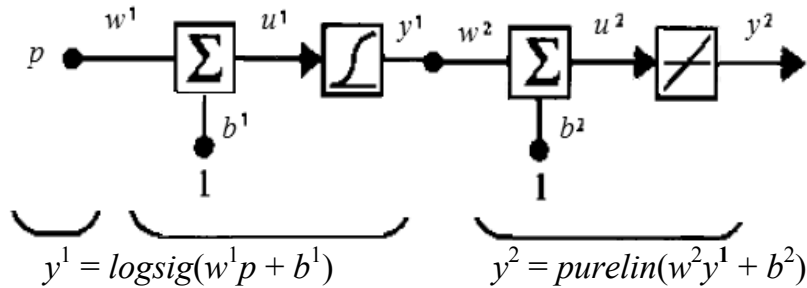


Рис. 7.12. Двошарова мережа

*Розв'язання. I.* Квадрат похибки можна визначити за формулою

$$(e)^2 = (t - y^2)^2 = \left( t - \left( w^2 \frac{1}{1 + \exp(-(w^1 p + b^1))} + b^2 \right) \right)^2.$$

**II.** Похідна квадрата похибки

$$\frac{\partial(e)^2}{\partial w^1} = 2e \frac{\partial e}{\partial w^1} = 2e \cdot \left( w^2 \frac{1}{\left( 1 + \exp(-(w^1 p + b^1)) \right)^2} \exp(-(w^1 p + b^1)) (-p) \right).$$

Підставивши початкові значення ваги та зсуву, одержимо:

$$y^1 = \frac{1}{1 + \exp\left(-\left(w^1 p + b^1\right)\right)} = \frac{1}{1 + \exp\left(-(1 \cdot 1 + 1)\right)} = 0,8808;$$

$$y^2 = w^2 y^1 + b = -2 \cdot 0,8808 + 1 = -0,7616;$$

$$e = (t - y^2) = 1 - (-0,7616) = 1,7616;$$

$$\frac{\partial(e)^2}{\partial w^1} = 2e \left( w^2 \frac{1}{\left(1 + \exp\left(-\left(w^1 p + b^1\right)\right)\right)^2} \exp\left(-\left(w^1 p + b^1\right)\right)(-p) \right) =$$

$$= 2 \cdot 1,7616 \cdot \left( -2 \frac{1}{\left(1 + \exp\left(-(1 \cdot 1 + 1)\right)\right)^2} \exp\left(-(1 \cdot 1 + 1)\right)(-1) \right) = 0,7398.$$

**III.** Щоб реалізувати алгоритм зворотного поширення, слід визначити коефіцієнти чутливості 1-го та 2-го шарів. Маємо:

$$s^2 = -2\dot{F}^2(u^2)(t - y^2) = -2 \cdot 1 \cdot 1,7616 = -3,5232;$$

$$s^1 = \dot{F}^1(u^1)w^2 s^2 = y^1(1 - y^1)w^2 s^2 =$$

$$= 0,8808 \cdot (1 - 0,8808) \cdot (-2) \cdot (-3,5232) = 0,7398.$$

Тепер можна обчислити значення  $\partial(e)^2/\partial w^1$ :

$$\frac{\partial(e)^2}{\partial w^1} = s^1 p = 0,7398 \cdot 1 = 0,7398,$$

що збігається з результатом, одержаним у п. II.

**Приклад 7.6.** Відомо, що похідну сигмоїдної ФА нейрона  $y = f(u) = \text{logsig}(u) = \frac{1}{1 + e^{-u}}$  обчислюють за формулою  $\dot{f}(u) = y(1 - y)$ .

Визначити похідну ФА у вигляді гіперболічного тангенса:  $y = f(u) = \text{tansig}(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$ .

*Розв'язання.* Розрахунок похідної гіперболічного тангенса має вигляд

$$\dot{f}(u) = \frac{df(u)}{du} = \frac{d}{du} \left( \frac{e^u - e^{-u}}{e^u + e^{-u}} \right) = - \frac{e^u - e^{-u}}{(e^u + e^{-u})^2} (e^u - e^{-u}) +$$

$$+ \frac{e^u - e^{-u}}{e^u + e^{-u}} = 1 - \frac{(e^u - e^{-u})^2}{(e^u + e^{-u})^2} = 1 - (y)^2.$$

**Приклад 7.7.** Нехай задано мережу (рис. 7.13), початкові значення ваги та зсуву якої мають такий вигляд:  $w^1(0) = -1$ ,  $b^1(0) = 1$ ;  $w^2(0) = -2$ ;  $b^2(0) = 1$ . На вхід мережі подається пара вхід-ціль  $\{p = -1, t = 1\}$ . Виконати одну ітерацію алгоритму зворотного поширення, якщо  $\alpha = 1$ .

*Розв'язання.* Спочатку обчислимо вихід мережі:

$$u^1 = w^1(0)p + b^1(0) = -1 \cdot (-1) + 1 = 2;$$

$$y^1 = \text{tansig}(u^1) = \frac{\exp(u^1) - \exp(-u^1)}{\exp(u^1) + \exp(-u^1)} = \frac{\exp(2) - \exp(-2)}{\exp(2) + \exp(-2)} = 0,964;$$

$$u^2 = w^2(0)y^1 + b^2(0) = -2 \cdot 0,964 + 1 = -0,928;$$

$$y^2 = \text{tansig}(u^2) = \frac{\exp(u^2) - \exp(-u^2)}{\exp(u^2) + \exp(-u^2)} = \frac{\exp(-0,928) - \exp(0,928)}{\exp(-0,928) + \exp(0,928)} = -0,7297;$$

$$e = (t - y^2) = 1 - (-0,7297) = 1,7297.$$

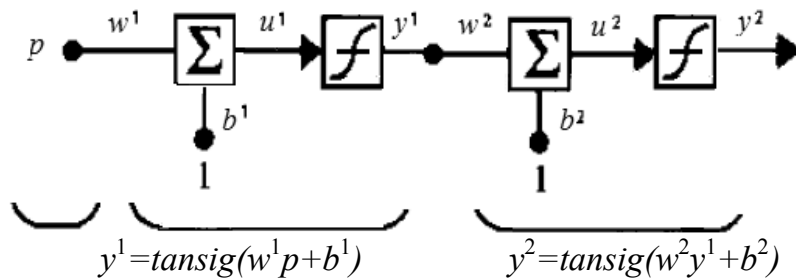


Рис. 7.13. Мережа із двома сигмоїдними шарами

Тепер визначимо коефіцієнти чутливості зворотного зв'язку для обох шарів:

– для другого:  $s^2 = -2\dot{F}^2(u^2)(t - y) = -2 \cdot \left(1 - (y^2)^2\right) \cdot e = -2 \cdot \left(1 - (-0,7297)^2\right) \times 1,7297 = -1,6175;$

– для першого:  $s^1 = \dot{F}^1(u^1)w^2 s^2 = \left(1 - (y^1)^2\right)w^2 s^2 = \left(1 - (0,964)^2\right) \cdot (-2) \times (-1,6175) = 0,2285.$

Після першої ітерації алгоритму зворотного поширення знайдемо оновлені значення ваги та зсуву:

$$w^2(1) = w^2(0) - \alpha s^2 y^1 = -2 - 1 \cdot (-1,6175) \cdot 0,964 = -0,4407;$$

$$w^1(1) = w^1(0) - \alpha s^1 p = -1 - 1 \cdot 0,2285 \cdot (-1) = -0,7715;$$

$$b^2(1) = b^2(0) - \alpha s^2 = 1 - 1 \cdot (-1,6175) = 2,6175;$$

$$b^1(1) = b^1(0) - \alpha s^1 = 1 - 1 \cdot (0,2285) = 0,7715.$$

**Приклад 7.8.** Визначити алгоритм зворотного поширення, який можна використати в лінійній рекурентній мережі (рис. 7.14) для оновлення значень ваги  $\mathbf{w} = [w_1 \ w_2]^T$ .

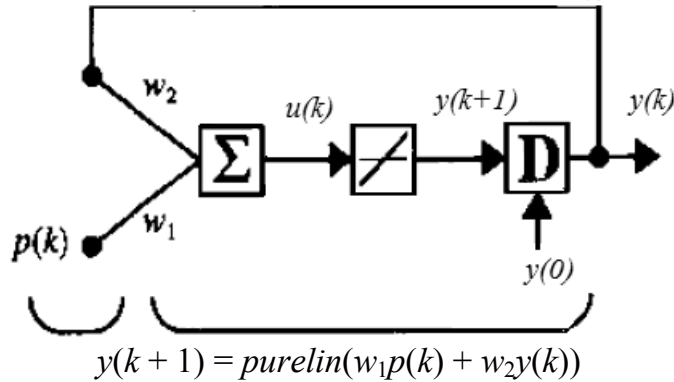


Рис. 7.14. Лінійна рекурентна одношарова мережа

*Розв'язання.* Визначимо функцію, яка описує ефективність функціонування цієї мережі. Як і для багатошарової мережі, використаємо функцію квадрату похибки:  $F(x) = (e(k))^2 = (t(k) - y(k))^2$ . Для оновлення значень ваги використаємо алгоритм найшвидшого спуску, отримаємо:

$$\Delta w_i = -\alpha \frac{\partial}{\partial w_i} F(x) = -\alpha \frac{\partial}{\partial w_i} (t(k) - y(k))^2 = -2\alpha(t(k) - y(k)) \left( -\frac{\partial y(k)}{\partial w_i} \right).$$

Отже, слід знайти  $\partial y(k) / \partial w_i$ , що вимагає визначення виходу мережі:  $y(k+1) = \text{purelin}(w_1 p(k) + w_2 y(k)) = w_1 p(k) + w_2 y(k)$ .

Візьмемо похідну від обох сторін цього рівняння за коефіцієнтами ваги:

$$\frac{\partial y(k+1)}{\partial w_1} = p(k) + w_2 \frac{\partial y(k)}{\partial w_1}; \quad \frac{\partial y(k+1)}{\partial w_2} = y(k) + w_2 \frac{\partial y(k)}{\partial w_2}.$$

Нехай маємо такі початкові значення:  $y(0) = 0$ ;  $\frac{\partial y(0)}{\partial w_1} = 0$ ;  $\frac{\partial y(0)}{\partial w_2} = 0$ .

Тоді перша дія мережі  $y(1) = w_1 p(0) + w_2 y(0) = w_1 p(0)$ , а перша похідна

$$\frac{\partial y(1)}{\partial w_1} = p(0) + w_2 \frac{\partial y(0)}{\partial w_1} = p(0); \quad \frac{\partial y(1)}{\partial w_2} = y(0) + w_2 \frac{\partial y(0)}{\partial w_2} = 0.$$

Перше оновлення ваги

$$\Delta w_i = -\alpha \frac{\partial}{\partial w_i} F(x) = -2\alpha(t(1) - y(1)) \cdot \left( -\frac{\partial y(1)}{\partial w_i} \right),$$

звідки

$$\Delta w_1 = -2\alpha(t(1) - y(1)) \cdot (-p(0)); \Delta w_2 = -2\alpha(t(1) - y(1)) \cdot 0 = 0.$$

**Приклад 7.9.** Показати, що алгоритм зворотного поширення для одношарової лінійної мережі АДАЛІН збігається з алгоритмом LMS.

*Розв'язання.* Алгоритм зворотного поширення обчислює коефіцієнт чутливості, який для одношарової лінійної мережі має вигляд  $s^1 = -2\dot{F}^1(u^1)(t - y) = -2(t - y) = -2e$ , тому оновлення значень ваги та зсуву відбувається таким чином:

$$\mathbf{W}^1(k+1) = \mathbf{W}^1(k) - \alpha s^1 (y^0)^T = \mathbf{W}^1(k) - \alpha(-2e)\mathbf{p}^T = \mathbf{W}^1(k) + 2\alpha e \mathbf{p}^T;$$

$$\mathbf{b}^1(k+1) = \mathbf{b}^1(k) - \alpha s^1 = \mathbf{b}^1(k) - \alpha(-2e) = \mathbf{b}^1(k) + 2\alpha e,$$

що збігається з алгоритмом LMS.

### Контрольні запитання

1. Опишіть тришаровий персептрон.
2. Опишіть алгоритм зворотного поширення.
3. Опишіть формулу визначення коефіцієнтів чутливості мереж із зворотним зв'язком.
4. Визначте теорему про універсальну апроксимацію для нелінійного, неперервного відображення вхід–вихід, пов'язану з БП.

### Задачі для самостійного розв'язання

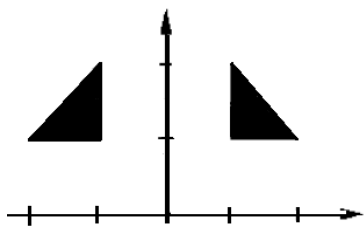


Рис. 7.15. Области класифікації

**Задача 7.1.** Спроектувати багатошарову мережу, яка виконує класифікацію заданих областей (рис. 7.15): має на виході «1», якщо вхідний вектор належить затемненій області (чи перебуває на її межі), й «-1» в іншому випадку.

**Задача 7.2.** Визначити одношарову мережу, яка має той самий показник ефективності функціонування, що й мережа, зображена на рис. 7.16.

**Задача 7.3.** Визначити вагу та зсув для мережі зі структурою 1-2-1, зображеною на рис. 7.17 так, щоб вихід мережі проходив через точки, зображені на рис. 7.18.

**Задача 7.4.** Задано мережу (рис. 7.17), для якої:

- 1) початкові значення ваги та зсуву мають вигляд  $w^1(0) = 1$ ;  
 $b^1(0) = -2$ ;  $w^2(0) = 1$ ;  $b^2(0) = 1$ ;

2) ФА мають вигляд  $f^1(u) = (u)^2$ ,  $f^2(u) = \frac{1}{u}$ ;

3) пара навчання вхід–ціль має вигляд  $\{p = 1, t = 1\}$ .

Виконати одну ітерацію алгоритму зворотного поширення, якщо  $\alpha = 1$ .

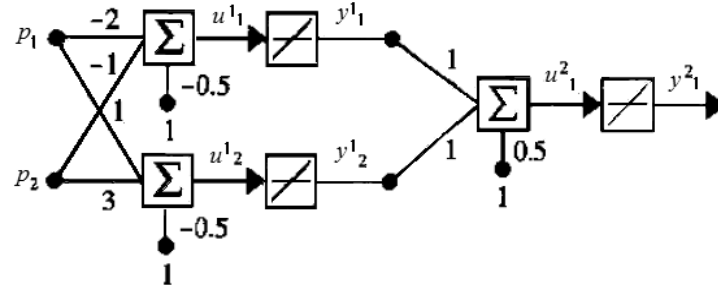


Рис. 7.16. Двошарова лінійна мережа

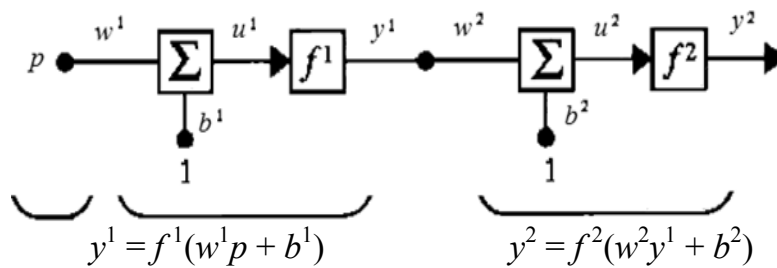


Рис. 7.17. Двошарова мережа

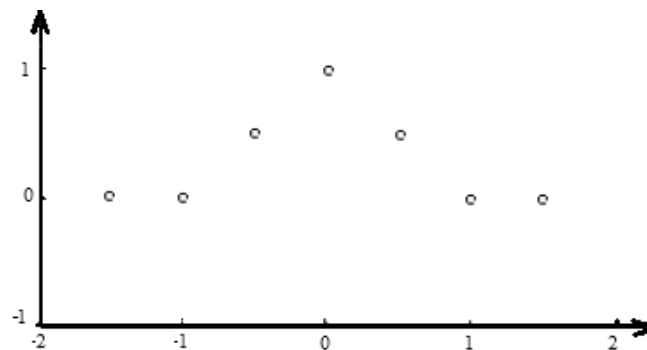


Рис. 7.18. Точки апроксимації функції

**Задача 7.5.** Нехай задано мережу (рис. 7.19), нейрон якої має ФА  $f^1(u) = (u)^2$ , а пара вхід–ціль  $\{\mathbf{p} = [1 \ 1]^T, \mathbf{t} = [8 \ 2]^T\}$ .

Виконати одну ітерацію алгоритму зворотного поширення, якщо  $\alpha = 1$ .

**Задача 7.6.** Мережа, зображена на рис. 7.20, не використовує стандартного формату нейрона і має вихід  $y = w_1 p_1 + w_{12} p_1 p_2 + w_2 p_2 + b$ .

Визначити правило навчання параметрів  $w_1, w_{12}, w_2$  і  $b$ , використавши алгоритм найшвидшого спуску (як це робиться з використанням зворотного зв'язку).

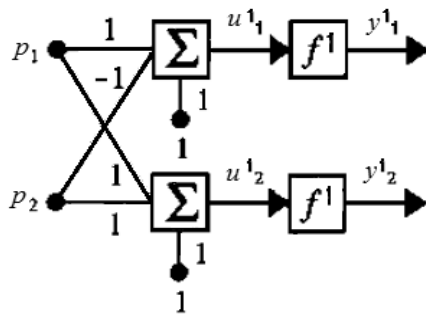
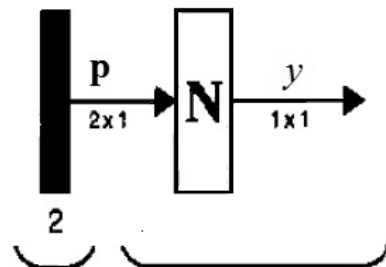


Рис. 7.19. Одношарова НМ



$$y = (w_1 p_1 + w_{12} p_1 p_2 + w_2 p_2 + b)$$

Рис. 7.20. Нейрон векторного добутку

**Задача 7.7.** Двошарову НМ зображено на рис 7.21, другий шар мережі безпосередньо пов'язаний із входом мережі.

Визначити алгоритм зворотного поширення для цієї мережі.

**Задача 7.8.** Нехай задано функцію  $g(p) = 1 + \sin\left(\frac{\pi}{8} p\right)$ ,  $-2 \leq p \leq 2$ .

Написати програму (у вигляді *m*-файлу), яка реалізує алгоритм зворотного поширення для БП зі структурою 1-2-1, що апроксимує цю функцію. Як початкові значення ваги та зсуву вибрати випадкові числа, рівномірно розподілені на відріжку  $[-0,5; 0,5]$ . Навчати НМ, використовуючи різні значення швидкості навчання  $\alpha$  та різні початкові умови.

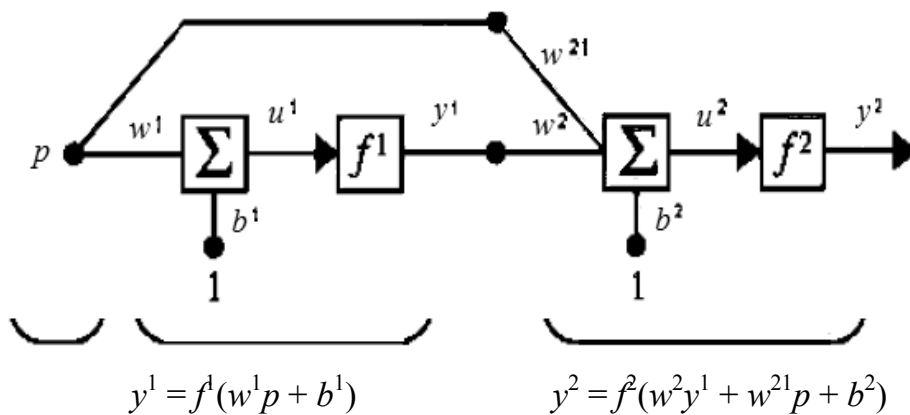


Рис. 7.21. Двошарова мережа з обхідним зв'язком



## Розділ 8. МОДИФІКАЦІЇ МЕТОДУ ЗВОРОТНОГО ПОШИРЕННЯ

### 8.1. Недоліки методу зворотного поширення

Алгоритм зворотного поширення (описаний у розд. 7) був головним досягненням у дослідженні багатосарових НМ, проте у більшості практичних додатків він виконується дуже повільно, тому було розроблено модифікації цього методу для прискорення його збіжності. Нагадаємо, що метод зворотного поширення є наближенням алгоритму найшвидшого спуску. У розд. 5 було показано, що метод найшвидшого спуску – найпростіший і, зазвичай, найповільніший метод мінімізації, а алгоритми спряжених градієнтів і Ньютона забезпечують більш швидку збіжність. Ці більш швидкі алгоритми можна використати для прискорення збіжності алгоритму зворотного поширення.

Дослідження з розробки більш швидких алгоритмів зворотного поширення можна розбити на такі дві групи: *евристичні методи* (наприклад, основані на зміні швидкості навчання або на використанні коефіцієнта імпульсу) та *методи, основані на стандартних числових методах оптимізації*. У цьому контексті розглянемо застосування двох методів числової оптимізації до навчання багатосарових мереж: алгоритми спряжених градієнтів та Левенберга–Марквардта (модифікований метод Ньютона). Зазначимо, що всі використовувані алгоритми застосовують процедуру зворотного поширення, тому їх можна назвати алгоритмами «зворотного зв'язку». Відмінності між алгоритмами полягають у механізмі отримання похідних, які використовують для налаштування вагової матриці. Іноді цей процес буває невдалим й алгоритм зворотного зв'язку є фактично алгоритмом найшвидшого спуску. Надалі скорочено називатимемо стандартний алгоритм зворотного поширення найшвидшим спуском зворотного зв'язку SDBP (Steepest Descent Backpropagation).

Оскільки середньоквадратична похибка для односарової лінійної мережі є квадратичною функцією, що має тільки одну стаціонарну точку мінімуму, то алгоритм LMS гарантує збіжність до розв'язку за мінімізації середньоквадратичної похибки. Крім того, матриця Гессе квадратичної функції є постійною, тому викривлення функції не відбувається й контурні лінії мають еліптичну форму.

Алгоритм SDBP є узагальненням алгоритму LMS (для односарових лінійних мереж він еквівалентний LMS). Як і алгоритм LMS, цей алгоритм також збігається до мінімальної точки, реалізуючи алгоритм найшвидшого

спуску для середньоквадратичної похибки. Однак для багат шарових мереж алгоритм SDBP має відмінності, пов'язані з формою поверхонь, що описують ефективність функціонування одношарових лінійних мереж і багат шарових нелінійних мереж (у вигляді середньоквадратичної похибки): якщо поверхня одношарової лінійної мережі має одну точку мінімуму, то поверхня багат шарової мережі може мати багато локальних точок мінімуму, при цьому її викривлення може змінюватися на великому діапазоні та на різних ділянках простору параметрів мережі (на «плоских» ділянках можна використовувати велике значення швидкості навчання, тоді як ділянки із сильним викривленням вимагають застосовувати невеликі значення).

Зазвичай початкові значення ваги та зсуву (параметри алгоритму SDBP) вибирають у вигляді малих випадкових значень. Недоцільно брати початкові параметри рівними нулю, оскільки початок координат простору параметрів може бути сідловою точкою поверхні ефективності функціонування мережі. Щоб переконатися, що алгоритм збігається до глобальної точки мінімуму, корисно використовувати декілька різних початкових припущень.

Розглянемо методи вдосконалення (модифікації) алгоритму SDBP [14], а саме евристичні методи (наприклад, основані на числовому імпульсі) та методи, основані на алгоритмах оптимізації.

## 8.2. Евристичні модифікації методу SDBP

*Метод із використанням імпульсу* – це модифікація алгоритму SDBP, основана на спостереженні, що збіжність алгоритму можна покращити, якщо вирівняти траєкторію коливання. Це можна зробити за допомогою фільтра низьких частот. Перш ніж застосувати імпульс до мережі, розглянемо простий приклад, який показує ефект згладжування.

**Приклад 8.1.** Розглянемо такий фільтр першого рівня:

$$y(k) = \gamma u(k-1) + (1-\gamma)w(k), \quad 0 \leq \gamma < 1,$$

де  $y(k)$  – вихід фільтра;  $\gamma$  – коефіцієнт імпульсу;  $w(k)$  – вхід фільтра.

Дію цього фільтра для входу у вигляді синусоїди  $w(k) = 1 + \sin\left(\frac{2\pi k}{16}\right)$  та

значень коефіцієнта імпульсу  $\gamma = 0,9$  і  $\gamma = 0,98$  зображено на рис. 8.1. Можна побачити, що:

- 1) коливання виходу фільтра менші, ніж коливання його входу;
- 2) зі збільшенням значення коефіцієнта імпульсу  $\gamma$  коливання виходу фільтра зменшуються;
- 3) середнє значення виходу фільтра збігається з середнім значенням його входу.

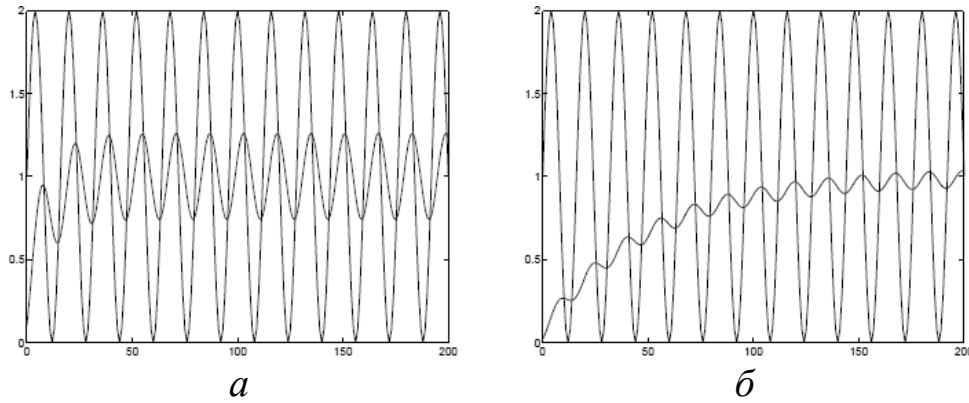


Рис. 8.1. Ефект згладжування завдяки імпульсу:

$a - \gamma = 0,9$ ;  $b - \gamma = 0,98$

Застосуємо одержані результати до НМ. Значення параметрів оновлюються відповідно до алгоритму SDBP:

$$\Delta \mathbf{W}^m(k) = -\alpha \mathbf{s}^m (\mathbf{y}^{m-1})^T; \Delta \mathbf{b}^m(k) = -\alpha \mathbf{s}^m. \quad (8.1)$$

Додавши імпульс до формул (8.1), одержимо рівняння зміни параметрів для модифікованого алгоритму зворотного поширення з імпульсом МОБР (Momentum Modification to Backpropagation) вигляду

$$\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) - (1-\gamma) \alpha \mathbf{s}^m (\mathbf{y}^{m-1})^T; \Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma) \alpha \mathbf{s}^m.$$

Використання імпульсу, як правило, прискорює процес збіжності.

**Метод із застосуванням змінного значення коефіцієнта швидкості навчання.** Як відомо, форма поверхні функціонування мережі для алгоритму SDBP може бути різною на різних ділянках простору параметрів. Можна прискорити збіжність алгоритму, коригуючи значення коефіцієнта швидкості навчання протягом всього періоду навчання мережі. Для цього потрібно лише вирішити, коли і як саме треба змінити коефіцієнт. Існують різні підходи щодо зміни коефіцієнта швидкості навчання. Покажемо, яким чином можна прискорити збіжність алгоритму SDBP, збільшуючи значення коефіцієнта навчання на плоских ділянках поверхні функціонування мережі, а потім зменшуючи його зі збільшенням кривизни. Опишемо процедуру пакетного функціонування НМ, коли цей коефіцієнт змінюється відповідно до ефективності функціонування мережі. Алгоритм зворотного поширення зі змінним значенням коефіцієнта швидкості навчання VLBP (Variable Learning Rate Backpropagation Algorithm) складається з таких кроків:

**Крок 1.** Якщо квадратична похибка на всій вибірці навчання після оновлення ваги збільшилася більш ніж на декілька відсотків від значення  $\zeta$  (зазвичай 1 – 5 %), то останнє оновлення ваги відміняється, а коефіцієнт швидкості навчання зменшується через множення його на коефіцієнт  $\rho$ :  $0 < \rho < 1$ , при цьому встановлюється значення імпульсу  $\gamma = 0$ .

*Крок 2.* Якщо квадратична похибка зменшилася після оновлення ваги, то це оновлення ваги є прийнятним і коефіцієнт швидкості навчання збільшується через множення його на деякий коефіцієнт  $\eta > 1$ . Якщо коефіцієнт імпульсу  $\gamma = 0$ , то його значення змінюється на початкове.

*Крок 3.* Якщо квадратична похибка збільшилася менше, ніж на  $\zeta$ , то оновлення ваги є прийнятним, а коефіцієнт швидкості навчання не змінюється. Якщо коефіцієнт імпульсу  $\gamma = 0$ , то його значення змінюється на попереднє.

Існує багато версій алгоритму VLBP, серед них алгоритми Т. Толленаера й Р. Джейкобса. Евристичні модифікації алгоритму SDBP часто забезпечують набагато швидшу збіжність для деяких задач, проте мають два основні недоліки: вони, по-перше, вимагають вибору декількох параметрів (наприклад,  $\zeta$ ,  $\rho$  та  $\gamma$ ), тоді як для алгоритму SDBP потрібний єдиний параметр – коефіцієнт швидкості навчання  $\alpha$ ; по-друге, іноді можуть не вирішити завдання, які вирішує алгоритм SDBP.

### 8.3. Методи, основані на алгоритмах оптимізації

Розглянемо методи, основані на алгоритмах оптимізації: спряжених градієнтів і Левенберга–Марквардта. У розд. 5 було розглянуто алгоритм спряжених градієнтів для квадратичних функцій. Щоб застосувати цей алгоритм у загальному випадку, додамо до нього дві процедури. Другий числовий метод оптимізації – алгоритм Левенберга–Марквардта є модифікацією методу Ньютона і придатний для навчання НМ.

**Метод спряжених градієнтів.** Алгоритм спряжених градієнтів не вимагає обчислення других похідних і є збіжним для квадратичних функцій. Покажемо, як такий алгоритм можна використати для навчання НМ. Назвемо його алгоритмом спряжених градієнтів зворотного поширення CGBP (Conjugate Gradient Backpropagation). Як зазначалося в розд. 5, алгоритм спряжених градієнтів складається з таких кроків:

*Крок 1.* Обираємо як перший напрямок пошуку  $\mathbf{p}_0$  від'ємного градієнта:  $\mathbf{p}_0 = -\mathbf{g}_0$ , де  $\mathbf{g}_k = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$

*Крок 2.* Визначаємо швидкість навчання  $\alpha_k$  таким чином, щоб мінімізувати функцію за напрямом пошуку (для КФ  $\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$ ):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

*Крок 3.* Обираємо наступний напрям пошуку відповідно до формули  $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$  з використанням однієї з формул:  $\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$ ,

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \text{ або } \beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}.$$

*Крок 4.* Повторюємо кроки 2 і 3, якщо алгоритм не збігається.

Оскільки показник ефективності функціонування НМ не є квадратичною функцією, то алгоритм спряжених градієнтів не можна застосовувати напряму для навчання НМ. Це впливає на алгоритм двома способами:

1) не можна використати рівняння  $\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$  для мінімізації

функції уздовж прямої, як цього вимагає крок 2;

2) точного мінімуму не буде досягнуто за скінченну кількість кроків, і тому після деякої кількості ітерацій алгоритм потрібно зупинити.

Враховуючи це, розглянемо спочатку метод лінійної оптимізації.

Загальна процедура визначення мінімуму функції у певному напрямі включає такі етапи:

1) визначення деякого початкового інтервалу, який містить точку мінімуму;

2) зменшення розміру попереднього інтервалу, поки не буде отримане значення мінімуму з бажаною точністю.

**Визначення інтервалу.** Використаємо метод порівняння значень функції (рис. 8.2). Розпочнемо визначення інтервалу з оцінювання значення функції у початковій точці  $a_1$ , яка відповідає початковим значенням ваги та зсуву мережі (інакше кажучи, оцінимо значення  $F(\mathbf{x}_0)$ ). На наступному кроці треба оцінити значення функції у точці  $b_1$ , яка розміщена на відстані  $\varepsilon$  від точки  $a_1$  уздовж першого напрямку пошуку  $\mathbf{p}_0$  (інакше кажучи, оцінюється значення  $F(\mathbf{x}_0 + \varepsilon \mathbf{p}_0)$ ). Продовжимо оцінювати показник ефективності функціонування НМ у нових точках  $b_i$ , послідовно подвоюючи відстань між ними. Цей процес зупиняється, коли функція зростає між двома послідовними точками – на рис. 8.2 між точками  $b_3$  і  $b_4$ , тому мінімум розміщений між точками  $a_5$  і  $b_5$ . Звузити інтервал не можна, оскільки мінімум може бути або в інтервалі  $[a_4; b_4]$ , або в інтервалі  $[a_3; b_3]$  (рис. 8.3).

**Звуження інтервалу.** Тепер, коли визначено інтервал, що містить точку мінімуму, виконаємо процедуру його звуження, оцінивши функцію в точках інтервалу  $[a_5; b_5]$ . З рис. 8.3 видно, що для зменшення розміру інтервалу необхідно оцінити функцію принаймні у двох точках. Рис. 8.3, а показує, що жодна внутрішня оцінка функції не надає інформації щодо розміщення точки мінімуму. Проте, якщо оцінити функцію у двох точках  $c$  і  $d$  (рис. 8.3, б), то можна зменшити інтервал невизначеності.

Якщо  $F(c) > F(d)$ , то мінімум розміщений в інтервалі  $[c; b]$ , а якщо  $F(c) < F(d)$ , то в інтервалі  $[a; d]$ .

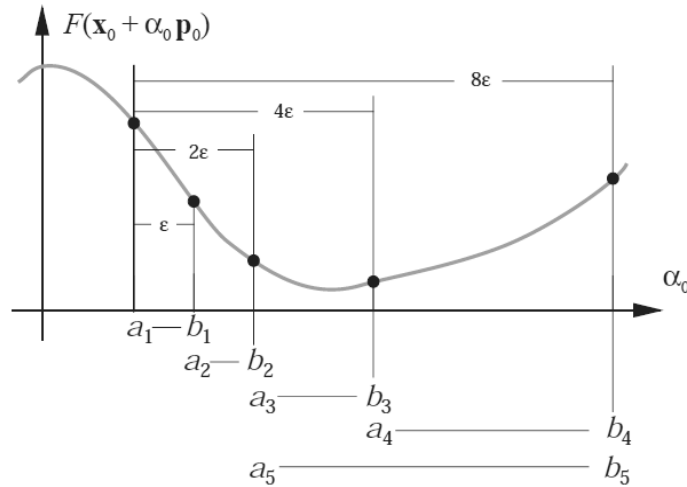


Рис. 8.2. Розміщення інтервалів

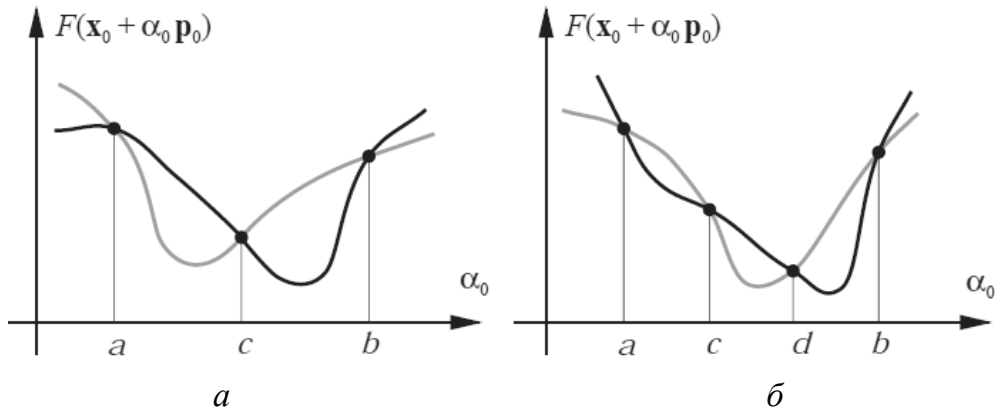


Рис. 8.3. Зменшення розміру інтервалу невизначеності:  
а – інтервал не зменшується; б – мінімум між точками  $c$  і  $b$

**Алгоритм золотого перетину.** Існують різні способи визначення розміщення точок  $c$  і  $d$ . Розглянемо метод золотого перетину, результатом якого є зменшення необхідної кількості оцінок функції. На кожній ітерації потрібно визначити лише одну нову оцінку. Наприклад, у випадку, зображеному на рис. 8.3, б, інтервал  $[a; b]$  буде замінено на інтервал  $[c; b]$ . У дод. Б наведено реалізацію (*m*-файл) алгоритму золотого перетину.

Зазвичай для квадратичних функцій алгоритм збігається до мінімуму за  $n$  ітерацій, де  $n$  – кількість параметрів, які оптимізуються. Оскільки показник ефективності функціонування багатошарових мереж не є квадратичною функцією, то алгоритм спряжених градієнтів не буде збігатися за  $n$  ітерацій. Існують різні процедури визначення напрямку пошуку після закінчення  $n$  ітерацій, найпростішою з них є повторне використання методу найшвидшого спуску (від'ємного градієнта).

Алгоритм спряжених градієнтів застосовують до багат шарових мереж (рис. 8.4), реалізуючи пакетний режим, оскільки градієнт обчислюється після того, як мережа отримала всю множину навчання. При цьому використовується алгоритм зворотного поширення для обчислення значень градієнтів  $\frac{\partial \bar{F}}{\partial w_{ij}^m} = s_i^m y_j^{m-1}$ ;  $\frac{\partial \bar{F}}{\partial b_i^m} = s_i^m$  та алгоритм спряжених градієнтів – для оновлення ваги.

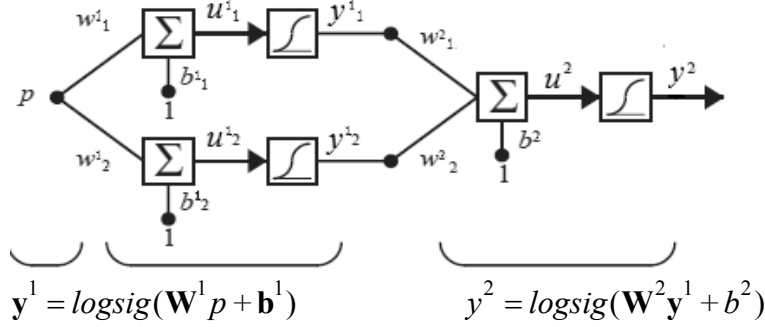


Рис. 8.4. Архітектура НМ, яка апроксимує задану функцію

**Алгоритм Левенберга–Марквардта зі зворотним зв’язком.** Levenberg–Marquardt Backpropagation (LMBP), або алгоритм Левенберга–Марквардта зі зворотним зв’язком – це модифікований метод Ньютона, призначений для визначення мінімуму функції, яка є сумою квадратів нелінійних функцій. Відомо, що метод Ньютона (розд. 5) використовують для оптимізації функції  $F(\mathbf{x})$ , він має вигляд

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k, \quad (8.2)$$

де  $\mathbf{A}_k = \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$ ,  $\mathbf{g}_k = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$ . Якщо припустити, що функція  $F(\mathbf{x})$  є сумою квадратів функцій:

$$F(\mathbf{x}) = \sum_{i=1}^N v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x}) \mathbf{v}(\mathbf{x}), \quad (8.3)$$

то  $j$ -й елемент її градієнта має такий вигляд:

$$[\nabla F(\mathbf{x})]_j = \frac{\partial F(\mathbf{x})}{\partial x_j} = 2 \sum_{i=1}^N v_i(\mathbf{x}) \frac{\partial v_i(\mathbf{x})}{\partial x_j}. \quad (8.4)$$

У матричній формі градієнт можна записати як  $\nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x})$ , де

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial \mathbf{v}_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial \mathbf{v}_1(\mathbf{x})}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial \mathbf{v}_N(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial \mathbf{v}_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \quad (8.5)$$

є матрицею Якобі.

Визначимо матрицю Гессе,  $(k, j)$ -й елемент якої  $[\nabla^2 F(\mathbf{x})]_{kj} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_j} = 2 \sum_{i=1}^N \left( \frac{\partial v_i(\mathbf{x})}{\partial x_k} \frac{\partial v_i(\mathbf{x})}{\partial x_j} + v_j(\mathbf{x}) \frac{\partial^2 v_i(\mathbf{x})}{\partial x_k \partial x_j} \right)$ . У матричній формі матрицю

Гессе можна записати як  $\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x})$ , де  $\mathbf{S}(\mathbf{x}) = \sum_{i=1}^N v_i(\mathbf{x}) \nabla^2 v_i(\mathbf{x})$ .

Якщо припустити, що функція  $\mathbf{S}(\mathbf{x})$  має мінімальне значення, то матриця Гессе наближено дорівнює

$$\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}). \quad (8.6)$$

Враховуючи формулу (8.6) і підставивши (8.4) у (8.2), одержимо формулу Ньютона-Гаусса:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1} 2\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k).$$

Зазначимо, що *перевага цього методу* відносно стандартного методу Ньютона полягає в тому, що він не вимагає обчислення других похідних. Недоліком методу Ньютона є те, що матриця Гессе  $\mathbf{H} = (\mathbf{J}^T\mathbf{J})$  може не мати оберненої. Однак цю ситуацію можна змінити, використавши наближену матрицю Гессе:  $\mathbf{G} = \mathbf{H} + \mu\mathbf{I}$ . Розглянемо, чи існує матриця, обернена до  $\mathbf{G}$ .

Нехай власні значення і власні вектори матриці  $\mathbf{H}$  дорівнюють відповідно  $\{\lambda_1, \dots, \lambda_n\}$  і  $\{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ . Тоді маємо:

$$\mathbf{G}\mathbf{z}_i = [\mathbf{H} + \mu\mathbf{I}]\mathbf{z}_i = \mathbf{H}\mathbf{z}_i + \mu\mathbf{z}_i = \lambda_i\mathbf{z}_i + \mu\mathbf{z}_i = (\lambda_i + \mu)\mathbf{z}_i.$$

Отже, власні вектори матриці  $\mathbf{G}$  збігаються з власними векторами матриці  $\mathbf{H}$ , а власні значення  $\mathbf{G}$  мають вигляд  $(\lambda_i + \mu)$ . Матрицю  $\mathbf{G}$  можна зробити додатно визначеною, збільшивши  $\mu$  так, щоб  $(\lambda_i + \mu) > 0$  для всіх  $1 \leq i \leq n$ . Тому можна стверджувати, що існує обернена до  $\mathbf{G}$  матриця.

Отже, алгоритм Левенберга–Марквардта має такий вигляд:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k),$$

або

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k).$$

Збільшення значення  $\mu_k$  наближає алгоритм Левенберга–Марквардта до алгоритму найшвидшого спуску з маленьким значенням коефіцієнта швидкості навчання. Для великих значень  $\mu_k$  маємо  $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{\mu_k} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) =$

$$= \mathbf{x}_k - \frac{1}{2\mu_k} \nabla F(\mathbf{x}_k). \text{ Якщо } \mu_k \rightarrow 0, \text{ то алгоритм Левенберга–Марквардта}$$

збігається з методом Ньютона–Гаусса.



Алгоритм розпочинається з вибору невеликого значення  $\mu_k$  (наприклад,  $\mu_k = 0,01$ ). Якщо після першої ітерації значення  $F(\mathbf{x})$  не зменшиться, то операцію повторюють зі значенням  $\nu\mu_k$ , де коефіцієнт  $\nu > 1$  (наприклад,  $\nu = 10$ ). У результаті значення  $F(\mathbf{x})$  має зменшитися, оскільки було зроблено крок у напрямку мінімуму функції за методом найшвидшого спуску. Якщо після першої ітерації значення  $F(\mathbf{x})$  зменшиться, то на наступному кроці використовують значення  $\mu_k/\nu$  й алгоритм наближатиметься до методу Ньютона–Гаусса, який повинен забезпечити швидшу збіжність. Отже, алгоритм Левенберга–Марквардта забезпечує узгодження між швидкістю методу Ньютона та гарантованою збіжністю методу найшвидшого спуску.

Розглянемо, як можна застосувати алгоритм Левенберга–Марквардта до навчання багат шарової мережі. Показник ефективності функціонування багат шарової мережі – це квадратична похибка вигляду  $F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{y})^T (\mathbf{t} - \mathbf{y})]$ . Якщо кожна ціль досягається з однаковою ймовірністю, то квадратична похибка пропорційна сумі квадратичних похибок для  $Q$  цілей у множині навчання:

$$F(\mathbf{x}) = \sum_{q=1}^Q (\mathbf{t}_q - \mathbf{y}_q)^T (\mathbf{t}_q - \mathbf{y}_q) = \sum_{q=1}^Q \mathbf{e}_q^T \mathbf{e}_q = \sum_{q=1}^Q \sum_{j=1}^{S^M} (e_{jq})^2 = \sum_{i=1}^N (\mathbf{v}_i(\mathbf{x}))^2, \quad (8.7)$$

де  $e_{jq}$  –  $j$ -й елемент похибки для  $q$ -ї пари вхід–ціль. Формула (8.7) еквівалентна формулі (8.3), для якої було запропоновано алгоритм Левенберга–Марквардта, тому потрібно адаптувати цей алгоритм до навчання мережі.

**Обчислення матриці Якобі.** Основний крок в алгоритмі Левенберга–Марквардта – обчислення матриці Якобі. Щоб виконати це обчислення, змінимо алгоритм зворотного поширення. Відповідно до стандартної процедури зворотного зв'язку SDPB необхідно обчислити похідні квадратів похибок щодо ваги та зсуву мережі. На відміну від цього, для визначення матриці Якобі слід обчислити похідні самих похибок.

Знову розглянемо формулу (8.7). Якщо застосувати вектор похибок  $\mathbf{v}$  та вектор параметрів  $\mathbf{x}$ , а саме

$$\begin{aligned} \mathbf{v} &= [v_1 \dots v_N]^T = [e_{11} \dots e_{S^M 1} \ e_{12} \dots e_{S^M Q}]^T; \\ \mathbf{x} &= [x_1 \dots x_n]^T = [w_{11}^1 \dots w_{S^1 R}^1 \ b_1^1 \dots b_{S^1}^1 \ w_{11}^2 \dots b_{S^M}^M]^T; \end{aligned} \quad (8.8)$$

$$N = QS^M, \quad n = S^1(R+1) + S^2(S^1+1) + \dots + S^M(S^{M-1}+1)$$

до формули (8.5), то матриця Якобі для навчання багат шарової мережі матиме вигляд

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_{11}^1} & \dots & \frac{\partial e_{11}}{\partial w_{S^1R}^1} \frac{\partial e_{11}}{\partial b_1^1} & \dots \\ \frac{\partial e_{21}}{\partial w_{11}^1} & \dots & \frac{\partial e_{21}}{\partial w_{S^1R}^1} \frac{\partial e_{21}}{\partial b_1^1} & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{S^M1}}{\partial w_{11}^1} & \dots & \frac{\partial e_{S^M1}}{\partial w_{S^1R}^1} \frac{\partial e_{S^M1}}{\partial b_1^1} & \dots \\ \frac{\partial e_{12}}{\partial w_{11}^1} & \dots & \frac{\partial e_{12}}{\partial w_{S^1R}^1} \frac{\partial e_{12}}{\partial b_1^1} & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}. \quad (8.9)$$

Елементи матриці Якобі можна обчислити завдяки простій модифікації алгоритму зворотного поширення. Стандартний зворотний зв'язок обчислює елементи  $\frac{\partial \bar{F}(\mathbf{x})}{\partial x_i} = \frac{\partial \mathbf{e}_q^T \mathbf{e}_q}{\partial x_i}$ . Елементи матриці Якобі обчислюють

за формулою  $[\mathbf{J}]_{hi} = \frac{\partial v_h}{\partial x_i} = \frac{\partial e_{kq}}{\partial x_i}$ . З урахуванням напрямку в алгоритмі

зворотного зв'язку маємо  $\frac{\partial \bar{F}}{\partial w_{ij}^m} = \frac{\partial \bar{F}}{\partial u_i^m} \cdot \frac{\partial u_i^m}{\partial w_{ij}^m}$ , де  $\frac{\partial \bar{F}}{\partial u_i^m} = s_i^m$  – коефіцієнт

чутливості.

Алгоритм зворотного зв'язку SDPB обчислює коефіцієнт чутливості через рекурентне співвідношення від останнього шару до першого. Цей спосіб можна використати для обчислення елементів матриці Якобі за формулою (8.9), якщо ввести нове поняття – *чутливість Марквардта*:

$\tilde{s}_{ih}^m = \frac{\partial v_h}{\partial u_{iq}^m} = \frac{\partial e_{kq}}{\partial u_{iq}^m}$ , де з формули (8.8) маємо  $h = (q-1)S^M + k$ . Тепер елементи

матриці Якобі можна обчислити за однією з формул:

1) якщо  $x_i \in$  вагою, маємо

$$[\mathbf{J}]_{hi} = \frac{\partial v_h}{\partial x_i} = \frac{\partial e_{kq}}{\partial w_{ij}^m} = \frac{\partial e_{kq}}{\partial u_{iq}^m} \cdot \frac{\partial u_{iq}^m}{\partial w_{ij}^m} = \tilde{s}_{ih}^m \cdot \frac{\partial u_{iq}^m}{\partial w_{ij}^m} = \tilde{s}_{ih}^m \cdot y_{jq}^{m-1}; \quad (8.10)$$

2) якщо  $x_i \in$  зсувом, маємо

$$[\mathbf{J}]_{hi} = \frac{\partial v_h}{\partial x_i} = \frac{\partial e_{kq}}{\partial b_i^m} = \frac{\partial e_{kq}}{\partial u_{iq}^m} \cdot \frac{\partial u_{iq}^m}{\partial b_i^m} = \tilde{s}_{ih}^m \cdot \frac{\partial u_{iq}^m}{\partial b_i^m} = \tilde{s}_{ih}^m. \quad (8.11)$$

Чутливість Марквардта можна обчислити за формулою (7.4):  $\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{u}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$ , яка визначає стандартне значення чутливості та має лише одну відмінність на вихідному шарі, а саме:

1) для стандартного зворотного зв'язку SDPB чутливість знаходять за формулою  $\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{u}^M)(\mathbf{t} - \mathbf{y})$ ;

2) для LMBP чутливість Марквардта визначають за формулою  $\tilde{s}_{ih}^M = \frac{\partial v_h}{\partial u_{iq}^M} = \frac{\partial e_{kq}}{\partial u_{iq}^M} = \frac{\partial (t_{kq} - y_{kq}^M)}{\partial u_{iq}^M} = \frac{\partial y_{kq}^M}{\partial u_{iq}^M} = \begin{cases} -\dot{f}^M(u_{iq}^M), & \text{якщо } i = k; \\ 0, & \text{якщо } i \neq k. \end{cases}$

Тому, коли на вхід мережі подається значення  $\mathbf{p}_q$  і мережа виводить значення  $\mathbf{y}_q^M$ , то алгоритм LMBP визначає початкове значення чутливості Марквардта як

$$\tilde{\mathbf{S}}_q^M = -\dot{\mathbf{F}}^M(\mathbf{u}_q^M), \quad (8.12)$$

де  $\dot{\mathbf{F}}^m(\mathbf{u}^m) = \begin{bmatrix} \dot{f}^m(u_1^m) & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \dot{f}^m(u_{s^m}^m) \end{bmatrix}$ . Щоб визначити один рядок матриці

Якобі, потрібно провести через мережу в зворотному напрямку кожний стовпець матриці  $\tilde{\mathbf{S}}_q^M$ . Ці стовпці також можна одержати на основі алгоритму зворотного зв'язку за формулою

$$\tilde{\mathbf{S}}_q^m = -\dot{\mathbf{F}}^m(\mathbf{u}_q^m)(\mathbf{W}^{m+1})^T \tilde{\mathbf{S}}_q^{m+1}. \quad (8.13)$$

Отже, повна матриця чутливості Марквардта для кожного шару має вигляд

$$\tilde{\mathbf{S}}^m = [\tilde{\mathbf{S}}_1^m | \dots | \tilde{\mathbf{S}}_Q^m]. \quad (8.14)$$

У зв'язку з тим, що алгоритм обчислює похідні кожної похибки, а не похідну суми квадратів похибок, необхідно поширити назад вектори чутливості  $\tilde{\mathbf{S}}_i^m$  для кожного входу. Зазначимо, що для кожного входу мережі маємо  $S^M$  похибок, що відповідає кількості нейронів вихідного шару, а для кожної похибки – один рядок матриці Якобі. Після визначення чутливості за допомогою алгоритму зворотного поширення обчислюємо матрицю Якобі за формулами (8.10) і (8.11). Отже, алгоритм LMBP включає такі дії:

*Крок 1.* У мережу вводять всі значення входів  $\mathbf{p} = \mathbf{p}_q$ ,  $1 \leq q \leq Q$ , та обчислюють відповідні виходи мережі:

$$\mathbf{y}^0 = \mathbf{p}; \quad \mathbf{y}^{m+1} = f^{m+1}(\mathbf{W}^{m+1}\mathbf{y}^m + \mathbf{b}^{m+1}), \quad m = 0, 1, \dots, M-1 \text{ і похибки } \mathbf{e}_q = \mathbf{t}_q - \mathbf{y}_q^M.$$

За формулою (8.7) обчислюють значення  $F(\mathbf{x})$  – суму квадратів похибок на всіх входах.

*Крок 2.* Матрицю Якобі обчислюють за формулою (8.9), а значення чутливості – за рекурентним співвідношенням (8.13). Після ініціалізації матриці чутливості Марквардта (8.12) її розширюють за формулою (8.14). Елементи матриці Якобі обчислюють за формулами (8.10) і (8.11).

*Крок 3.* Обчислюють значення  $\Delta \mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k)$ .

*Крок 4.* Повторно обчислюють суму квадратів похибок  $F(\mathbf{x})$  для  $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_k$ . Якщо  $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$ , то встановлюється  $\mu = \mu/\nu$  й алгоритм повертається до кроку 1 зі значенням  $\mathbf{x} = \mathbf{x}_{k+1}$ , в іншому разі встановлюється  $\mu = \nu\mu$  й алгоритм повертається до кроку 3 зі значенням  $\mathbf{x} = \mathbf{x}_k$ . Вважають, що алгоритм збігається, якщо норма градієнта  $\nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x})$  менша за деяке наперед встановлене значення.

Зазначимо, що алгоритм LMBP збігається за меншу кількість ітерацій, ніж розглянуті раніше алгоритми, але, зазвичай, він вимагає більшої кількості обчислень на кожній ітерації, оскільки використовує операцію обернення матриць. *Основний недолік алгоритму LMBP* – необхідність зберігати наближення матриці Гессе  $\mathbf{J}^T\mathbf{J}$  у вигляді матриці розмірністю  $[n \times n]$  (де  $n$  – кількість параметрів мережі), інші методи вимагають зберігати тільки градієнт, який має вигляд  $n$ -вимірного вектора. У зв'язку з цим використовувати алгоритм LMBP для навчання НМ з великою кількістю параметрів непрактично.

### Приклади розв'язання задач\*

**Приклад 8.2.** Нехай множина пар вхід–ціль має вигляд  $\{p_1 = -3, t_1 = 0,5\}; \{p_2 = 2, t_2 = 1\}$ . Потрібно навчити мережу, зображену на рис. 8.5, розпочавши алгоритм з такого припущення:  $w(0) = 0,4$ ;  $b(0) = 0,15$ . Виконати перший крок алгоритму зі зворотним зв'язком SDBP.

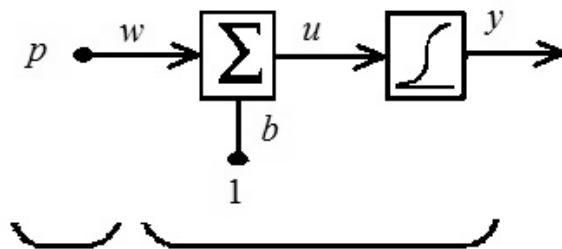


Рис. 8.5. Мережа до прикл. 8.2:  $y = \text{logsig}(wp + b)$

\* Задачі взято з посібника [14].

*Розв'язання.* Виконаємо перший крок алгоритму SDBP для першої пари вхід-ціль:  $y_1 = \text{logsig}(w(0)p_1 + b(0)) = \frac{1}{1 + \exp(-(0,4 \cdot (-3) + 0,15))} = 0,2592$ ;

$$e_1 = t_1 - y_1 = 0,5 - 0,2592 = 0,2408;$$

$$s_1 = -2\dot{f}(u_1)e_1 = -2y_1(1 - y_1)e_1 = -2 \cdot 0,2592 \cdot (1 - 0,2592) \cdot 0,2408 = -0,0925.$$

У напрямку, протилежному до градієнта, маємо  $\Delta w = -s_1 p_1 = -(-0,09025) \cdot (-3) = -0,2774$ ;  $\Delta b = -s_1 = -(-0,09025) = 0,09025$ . Отже, градієнт для першої пари вхід-ціль у точці початкового наближення має вигляд  $\begin{bmatrix} -0,2774 \\ 0,0925 \end{bmatrix}$ . Розглянемо другий крок алгоритму SDBP. Для цього

подамо на вхід мережі значення  $p_2$  та виконаємо перетворення у зворотному напрямку, одержимо:

$$y_2 = \text{logsig}(w(0)p_2 + b(0)) = \frac{1}{1 + \exp(-0,4 \cdot 2 + 0,15)} = 0,7211;$$

$$e_2 = t_2 - y_2 = 1 - 0,7211 = 0,2789;$$

$$s_2 = -2\dot{f}(u_2)e_2 = -2y_2(1 - y_2)e_2 = -2 \cdot 0,7211 \cdot (1 - 0,7211) \cdot 0,2789 = -0,1122.$$

Якщо тепер рухатися в напрямку, протилежному до градієнта, то одержимо  $\Delta w = -s_2 p_2 = -(-0,1122) \cdot 2 = 0,2243$ ;  $\Delta b = -s_2 = -(-0,1122) = 0,1122$ . Тому градієнт для другої пари вхід-ціль у точці початкового наближення має вигляд  $\begin{bmatrix} 0,2243 \\ 0,1122 \end{bmatrix}$ . Якщо тепер додати отримані результати

для обох пар вхід-ціль, то одержимо перший крок алгоритму SDBP:

$$\frac{1}{2} \left( \begin{bmatrix} -0,2774 \\ 0,0925 \end{bmatrix} + \begin{bmatrix} 0,2243 \\ 0,1122 \end{bmatrix} \right) = \frac{1}{2} \begin{bmatrix} -0,0531 \\ 0,2047 \end{bmatrix} = \begin{bmatrix} -0,0265 \\ 0,1023 \end{bmatrix}.$$

Зазначимо, що індивідуальні градієнти можуть мати різні напрямки відносно повного градієнта, однак у середньому після виконання декількох ітерацій траєкторія буде відповідати траєкторії алгоритму найшвидшого спуску.

**Приклад 8.3.** У розд. 5 було показано, що алгоритм найшвидшого спуску для КФ є стійким, якщо коефіцієнт швидкості навчання  $\alpha$  менший за  $2/\lambda_{\max}$ , де  $\lambda_{\max}$  – максимальне власне значення матриці Гессе.

Довести, що додавання до алгоритму найшвидшого спуску коефіцієнта імпульсу (або моменту)  $\gamma$  робить алгоритм стійким незалежно від значення  $\alpha$ .

*Розв'язання.* Стандартний алгоритм найшвидшого спуску має вигляд  $\Delta \mathbf{x}_k = -\alpha \nabla F(\mathbf{x}_k) = -\alpha \mathbf{g}_k$ . Якщо додати коефіцієнт імпульсу  $\gamma$ , то цей алгоритм матиме вигляд  $\Delta \mathbf{x}_k = -\gamma \Delta \mathbf{x}_{k-1} - (1-\gamma)\alpha \mathbf{g}_k$ .

Нагадаємо, що квадратична функція має вигляд  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$ , а її градієнт  $\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$ . Підставимо останній вираз у рівняння для алгоритму найшвидшого спуску з імпульсом, одержимо:  $\Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \gamma \Delta \mathbf{x}_{k-1} - (1-\gamma) \alpha (\mathbf{A} \mathbf{x}_k + \mathbf{d}) = \gamma (\mathbf{x}_k - \mathbf{x}_{k-1}) - (1-\gamma) \alpha (\mathbf{A} \mathbf{x}_k + \mathbf{d})$ , або  $\mathbf{x}_{k+1} = [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha \mathbf{A}] \mathbf{x}_k - \gamma \mathbf{x}_{k-1} - (1-\gamma)\alpha \mathbf{d}$ .

Нехай  $\tilde{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix}$ , тоді алгоритм найшвидшого спуску можна записати у вигляді  $\tilde{\mathbf{x}}_{k+1} = \begin{bmatrix} 0 & \mathbf{I} \\ -\gamma \mathbf{I} & (1+\gamma)\mathbf{I} - (1-\gamma)\alpha \mathbf{A} \end{bmatrix} \tilde{\mathbf{x}}_k + \begin{bmatrix} \mathbf{0} \\ -(1-\gamma)\alpha \mathbf{d} \end{bmatrix} = \mathbf{W} \tilde{\mathbf{x}}_k + \mathbf{v}$ .

Це рівняння зображує лінійну динамічну систему, яка є стійкою, якщо власні значення матриці  $\mathbf{W}$  менші за одиницю. Визначимо власні значення матриці  $\mathbf{W}$ : спочатку перепишемо її у такому вигляді:  $\mathbf{W} = \begin{bmatrix} 0 & \mathbf{I} \\ -\gamma \mathbf{I} & \mathbf{T} \end{bmatrix}$ , де

$\mathbf{T} = (1+\gamma)\mathbf{I} - (1-\gamma)\alpha \mathbf{A}$ . Власні значення та власні вектори матриці  $\mathbf{W}$  повинні задовольняти умову  $\mathbf{W} \mathbf{z}^w = \lambda^w \mathbf{z}^w$ , або  $\begin{bmatrix} 0 & \mathbf{I} \\ -\gamma \mathbf{I} & \mathbf{T} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{z}_1^w \\ \mathbf{z}_2^w \end{bmatrix} = \lambda^w \begin{bmatrix} \mathbf{z}_1^w \\ \mathbf{z}_2^w \end{bmatrix}$ .

Звідси маємо:  $\mathbf{z}_2^w = \lambda^w \mathbf{z}_1^w$ ;  $-\gamma \mathbf{z}_1^w + \lambda^t \mathbf{z}_2^w = \lambda^w \mathbf{z}_2^w$ . Обравши як вектор  $\mathbf{z}_2^w$  власний вектор матриці  $\mathbf{T}$  із відповідним власним значенням  $\lambda^t$ , отримаємо рівняння  $\mathbf{z}_2^w = \lambda^w \mathbf{z}_1^w$ ,  $-\gamma \mathbf{z}_1^w + \lambda^t \mathbf{z}_2^w = \lambda^w \mathbf{z}_2^w$ . Виразивши з першого рівняння

$\mathbf{z}_1^w$  та підставивши вираз  $\mathbf{z}_1^w = \frac{\mathbf{z}_2^w}{\lambda^w}$  у друге рівняння, одержимо:

$$-\frac{\gamma}{\lambda^w} \mathbf{z}_2^w + \lambda^t \mathbf{z}_2^w = \lambda^w \mathbf{z}_2^w, \text{ або } [(\lambda^w)^2 - \lambda^t (\lambda^w) + \gamma] \mathbf{z}_2^w = 0.$$

Отже, кожному власному значенню  $\lambda^t$  матриці  $\mathbf{T}$  відповідають два власних значення  $\lambda^w$  матриці  $\mathbf{W}$ , які є коренями квадратного рівняння:

$$(\lambda^w)^2 - \lambda^t (\lambda^w) + \gamma = 0, \text{ звідки } \lambda^w = \frac{\lambda^t \pm \sqrt{(\lambda^t)^2 - 4\gamma}}{2}.$$

Щоб алгоритм був стійким, кожне власне значення має бути меншим за одиницю. Покажемо, що це відбувається тільки тоді, коли  $\gamma$  змінюється на деякому інтервалі. Якщо власні значення  $\lambda^w$  є комплексними,

то  $|\lambda^w| = \sqrt{\left(\frac{(\lambda^t)^2}{4} + \frac{4\gamma - (\lambda^t)^2}{4}\right)} = \sqrt{\gamma} < 1$ , звідки  $0 < \gamma < 1$ . Залишається показати, що існує деякий інтервал значень  $\gamma$ , для якого всі власні значення  $\lambda^w$  є комплексними числами. Зазначимо, що  $\lambda^w$  є комплексним числом, якщо  $(\lambda^t)^2 - 4\gamma < 0$  або  $|\lambda^t| < 2\sqrt{\gamma}$ .

Покажемо тепер, що власні значення  $\lambda^t$  матриці **T** можна відобразити через власні значення матриці Гессе **A**. Нехай  $\{\lambda_1, \dots, \lambda_N\}$  і  $\{\mathbf{z}_1, \dots, \mathbf{z}_N\}$  – власні значення та власні вектори матриці Гессе відповідно. Тоді маємо:

$$\begin{aligned} \mathbf{T}\mathbf{z}_i &= [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha\mathbf{A}]\mathbf{z}_i = (1+\gamma)\mathbf{z}_i - (1-\gamma)\alpha\mathbf{A}\mathbf{z}_i = \\ &= (1+\gamma)\mathbf{z}_i - (1-\gamma)\alpha\lambda_i\mathbf{z}_i = [(1+\gamma) - (1-\gamma)\alpha\lambda_i]\mathbf{z}_i = \lambda_i^t\mathbf{z}_i. \end{aligned}$$

Тому власні вектори матриці **T** дорівнюють власним векторам матриці **A**, а її власні значення мають вигляд  $\lambda_i^t = \{(1+\gamma) - (1-\gamma)\alpha\lambda_i\}$ . Зазначимо, що  $\lambda^t$  набуває дійсних значень, якщо  $\gamma$ ,  $\alpha$  та  $\lambda_i$  є дійсними. Тому для комплексного числа  $\lambda^w$  маємо  $|\lambda^t| < 2\sqrt{\gamma}$ , або  $|(1+\gamma) - (1-\gamma)\alpha\lambda_i| < 2\sqrt{\gamma}$ . Якщо  $\gamma = 1$ , то обидві частини нерівності будуть дорівнювати двом. Функція у правій частині нерівності має нахил «1» за  $\gamma = 1$ , а функція у лівій частині – нахил  $(1 + \alpha\lambda_i)$ . Оскільки власні значення матриці Гессе будуть додатними, якщо функція має сильний мінімум та коефіцієнт швидкості навчання є додатним, то цей нахил має бути більшим за одиницю. Це показує, що нерівність завжди виконується для значень  $\gamma$ , які близькі до одиниці.

Отже, маємо:

1) якщо до квадратичної функції застосувати алгоритм найшвидшого спуску з використанням імпульсу, то існує коефіцієнт імпульсу, який зробить алгоритм стійким незалежно від коефіцієнта навчання;

2) якщо значення імпульсу  $\gamma$  є достатньо близьким до одиниці, то власні значення матриці **W** дорівнюють  $\sqrt{\gamma}$  й алгоритм збігатиметься швидко;

3) чим менше значення імпульсу  $\gamma$ , тим швидше збігається алгоритм: якщо значення  $\gamma$  наближається до одиниці, то час виконання алгоритму збільшується.

**Приклад 8.4.** Нехай задано функцію  $F(\mathbf{x}) = x_1^2 + 25x_2^2$ , яка описує ефективність функціонування мережі. Виконати три ітерації алгоритму навчання мережі зі змінним значенням швидкості навчання, розпочавши

алгоритм із точки  $\mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$  та використавши такі значення параметрів:

$\alpha = 0,05$ ;  $\gamma = 0,2$ ;  $\eta = 1,5$ ;  $\rho = 0,5$ ;  $\zeta = 5\%$ .

*Розв'язання.* Визначимо значення функції  $F(\mathbf{x})$  у точці  $\mathbf{x}_0$ :  $F(\mathbf{x}_0) = \frac{1}{2} \mathbf{x}_0^T \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \mathbf{x}_0 = \frac{1}{2} [0,5 \quad 0,5] \cdot \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \cdot \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} = 6,5$ . Градієнт функції  $F(\mathbf{x})$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix} \text{ і в точці } \mathbf{x}_0 \text{ становить } \mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}.$$

Запишемо *перший крок алгоритму*:

$$\Delta \mathbf{x}_0 = \gamma \Delta \mathbf{x}_1 - (1 - \gamma) \alpha \mathbf{g}_0 = 0,2 \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0,8 \cdot 0,05 \cdot \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} -0,04 \\ -1 \end{bmatrix};$$

$$\mathbf{x}_1^t = \mathbf{x}_0 + \Delta \mathbf{x}_0 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix} + \begin{bmatrix} -0,04 \\ -1 \end{bmatrix} = \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix}.$$

Щоб перевірити ефективність цього кроку, визначимо значення функції  $F(\mathbf{x})$  у новій точці:

$$F(\mathbf{x}_1^t) = \frac{1}{2} (\mathbf{x}_1^t)^T \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \mathbf{x}_1^t = \frac{1}{2} [0,46 \quad -0,5] \cdot \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \cdot \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix} = 6,4616.$$

Оскільки  $F(\mathbf{x}_1^t) < F(\mathbf{x}_0)$ , то перший крок є прийнятним (не відхиляється) і швидкість навчання збільшується:  $\mathbf{x}_1 = \mathbf{x}_1^t = \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix}$ ;  $F(\mathbf{x}_1) = 6,4616$ ;

$$\alpha = \eta \alpha = 1,5 \cdot 0,05 = 0,075.$$

Запишемо *другий крок алгоритму*:

$$\mathbf{g}_1 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_1} = \begin{bmatrix} 0,92 \\ -25 \end{bmatrix};$$

$$\Delta \mathbf{x}_1 = \gamma \Delta \mathbf{x}_0 - (1 - \gamma) \alpha \mathbf{g}_1 = 0,2 \cdot \begin{bmatrix} -0,04 \\ -1 \end{bmatrix} - 0,8 \cdot 0,075 \cdot \begin{bmatrix} 0,92 \\ -25 \end{bmatrix} = \begin{bmatrix} -0,0632 \\ 1,3 \end{bmatrix};$$

$$\mathbf{x}_2^t = \mathbf{x}_1 + \Delta \mathbf{x}_1 = \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix} + \begin{bmatrix} -0,0632 \\ 1,3 \end{bmatrix} = \begin{bmatrix} 0,3968 \\ 0,8 \end{bmatrix}.$$

Визначимо значення функції  $F(\mathbf{x})$  у точці  $\mathbf{x}_2^t = \begin{bmatrix} 0,3968 \\ 0,8 \end{bmatrix}$ :

$$F(\mathbf{x}_2^t) = \frac{1}{2} (\mathbf{x}_2^t)^T \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \mathbf{x}_2^t = \frac{1}{2} [0,3968 \quad 0,8] \cdot \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \cdot \begin{bmatrix} 0,3968 \\ 0,8 \end{bmatrix} = 16,157.$$

Оскільки значення  $F(\mathbf{x}_2^t)$  перевищує  $F(\mathbf{x}_1)$  більше, ніж на 5 %, то останній крок відхиляється і коефіцієнт швидкості навчання зменшується,



а значення коефіцієнта імпульсу встановлюється рівним нулю:

$$\mathbf{x}_2 = \mathbf{x}_1 = \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix}; F(\mathbf{x}_2) = F(\mathbf{x}_1) = 6,4616;$$

$$\alpha = \rho\alpha = 0,5 \cdot 0,075 = 0,0375; \gamma = 0.$$

Запишемо *третій* крок алгоритму:

$$\Delta\mathbf{x}_2 = -\alpha\mathbf{g}_2 = -0,0375 \begin{bmatrix} 0,92 \\ -25 \end{bmatrix} = \begin{bmatrix} -0,0345 \\ 0,9375 \end{bmatrix};$$

$$\mathbf{x}_3^t = \mathbf{x}_2 + \Delta\mathbf{x}_2 = \begin{bmatrix} 0,46 \\ -0,5 \end{bmatrix} + \begin{bmatrix} -0,0345 \\ 0,9375 \end{bmatrix} = \begin{bmatrix} 0,4255 \\ 0,4375 \end{bmatrix};$$

$$F(\mathbf{x}_3^t) = \frac{1}{2}(\mathbf{x}_3^t)^T \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \mathbf{x}_3^t = \frac{1}{2} [0,4255 \quad 0,4375] \cdot \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \cdot \begin{bmatrix} 0,4255 \\ 0,4375 \end{bmatrix} = 4,966.$$

Оскільки  $F(\mathbf{x}_3^t) < F(\mathbf{x}_2)$ , то цей крок є прийнятним: початкове значення імпульсу відновлюється, а коефіцієнт швидкості навчання збільшується ( $\mathbf{x}_3 = \mathbf{x}_3^t$ ,  $\gamma = 0,2$ ;  $\alpha = \eta\alpha = 1,5 \cdot 0,0375 = 0,05625$ ). На цьому завершується третій крок.

**Приклад 8.5.** Нехай задано функцію  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$ , яка описує ефективність функціонування мережі. Виконати одну ітерацію методу спряжених градієнтів із початковим припущенням у точці  $\mathbf{x}_0 = \begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix}$ . При цьому для лінійної мінімізації використати інтервал, визначений за допомогою оцінки функції. Скоротити цей інтервал за допомогою методу золотого перетину.

*Розв'язання.* Градієнт функції  $F(\mathbf{x})$  має вигляд  $\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix}$ .

Як *перший* крок методу спряжених градієнтів використаємо метод найшвидшого спуску:  $\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}$ . На першій ітерації необхідно мінімізувати значення функції  $F(\mathbf{x})$  уздовж прямої:  $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 = \begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix} + \alpha_0 \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}$ . Спочатку визначимо інтервал, на якому розміщено мінімальну оцінку функції  $F(\mathbf{x})$ . Нехай початковий розмір кроку  $\varepsilon = 0,075$ , тоді маємо:  $a_1 = 0\varepsilon$ ,  $F(a_1) = F\left(\begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix}\right) = 0,5025$ ;

$$b_1 = 1\varepsilon = 0,075, F(b_1) = F\left(\begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix} + 0,075 \cdot \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}\right) = 0,3721;$$

$$b_2 = 2\varepsilon = 0,15, F(b_2) = F\left(\begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix} + 0,15 \cdot \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}\right) = 0,2678;$$

$$b_3 = 4\varepsilon = 0,3, F(b_3) = F\left(\begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix} + 0,3 \cdot \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}\right) = 0,1373;$$

$$b_4 = 8\varepsilon = 0,6, F(b_4) = F\left(\begin{bmatrix} 0,8 \\ -0,25 \end{bmatrix} + 0,6 \cdot \begin{bmatrix} -1,35 \\ -0,3 \end{bmatrix}\right) = 0,1893.$$

Оскільки значення функції збільшується між двома останніми послідовними оцінками, то можна стверджувати, що мінімум розміщений на інтервалі  $[0,15; 0,6]$ . Ілюстрацію цього процесу зображено колами на рис. 8.6, а кінцевий інтервал позначений чорним колом.

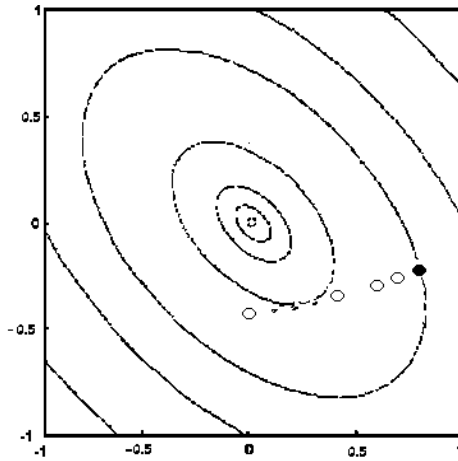


Рис. 8.6. Приклад лінійної мінімізації

Наступний крок у процесі лінійної мінімізації – це звуження інтервалу з використанням методу золотого перетину. Цей процес включає в себе такі дії:  $c_1 = a_1 + (1 - \tau)(b_1 - a_1) = 0,15 + 0,382(0,6 - 0,15) = 0,3219$ , де  $\tau = 0,618$ ;

$$d_1 = b_1 - (1 - \tau)(b_1 - a_1) = 0,6 - 0,382 \cdot (0,6 - 0,15) = 0,4281;$$

$$F_a = 0,2678; F_b = 0,1893; F_c = 0,1270; F_d = 0,1085.$$

Якщо  $F_c > F_d$ , маємо:  $a_2 = c_1 = 0,3219$ ;  $b_2 = b_1 = 0,6$ ;  $c_2 = d_1 = 0,4281$ ;

$$d_2 = b_2 - (1 - \tau)(b_2 - a_2) = 0,6 - 0,382 \cdot (0,6 - 0,3219) = 0,4938;$$

$$F_a = F_c = 0,1270, F_c = F_d = 0,1085, F_d = F(d_2) = 0,1232.$$

Якщо  $F_c < F_d$ , маємо:  $a_3 = a_2 = 0,3219$ ,  $b_3 = d_2 = 0,4938$ ,  $d_3 = c_2 = 0,4281$ ;

$$c_3 = a_3 + (1 - \tau)(b_3 - a_3) = 0,3219 + 0,382 \cdot (0,4938 - 0,3219) = 0,3876;$$

$$F_b = F_d = 0,1232; F_d = F_c = 0,1085; F_c = F(c_3) = 0,1094.$$

Цей процес триває доти, поки  $b_{k+1} - a_{k+1} < tol$ , де  $tol$  – встановлена користувачем точність.

**Приклад 8.6.** Розглянемо використання двошарової мережі (рис. 8.7) для апроксимації функції. Нехай ФА цієї мережі мають вигляд  $f^1(u) = (u)^2$ ;  $f^2(u) = u$ , а похідні, які їм відповідають,  $\dot{f}^1(u) = 2u$ ;  $\dot{f}^2(u) = 1$ . Множина пар вхід–ціль  $\{p_1 = 1, t_1 = 1\}$ ;  $\{p_2 = 2, t_2 = 2\}$ , а початкові значення параметрів такі:  $w^1 = 1$ ;  $b^1 = 0$ ;  $w^2 = 2$ ;  $b^2 = 1$ .

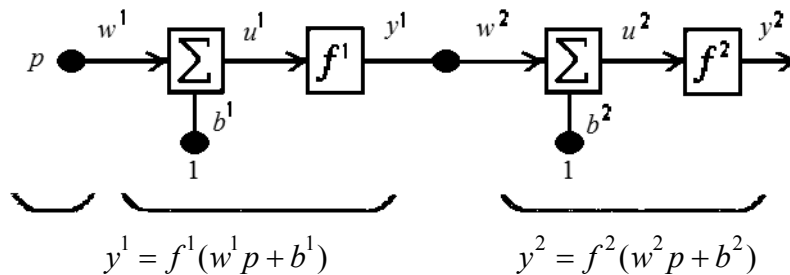


Рис. 8.7. Мережа для ілюстрації алгоритму LMBP

Визначити матрицю Якобі за методом Левенберга–Марквардта, виконавши перший крок методу.

*Розв’язання.* На першому кроці визначимо входи кожного шару та похибку мережі:

1) якщо вхід першого шару  $y_1^0 = p_1 = 1$ , то маємо:  $u_1^1 = w^1 y_1^0 + b^1 = 1 \cdot 1 + 0 = 1$ ;  $y_1^1 = f^1(u_1^1) = (1)^2 = 1$ ;  $u_1^2 = w^2 y_1^1 + b^2 = 2 \cdot 1 + 1 = 3$ ;  $y_1^2 = f^2(u_1^2) = 3$ , похибка  $e_1 = t_1 - y_1^2 = 1 - 3 = -2$ ;

2) якщо вхід першого шару  $y_2^0 = p_2 = 2$ , то маємо:  $u_2^1 = w^1 y_2^0 + b^1 = 1 \cdot 2 + 0 = 2$ ;  $y_2^1 = f^1(u_2^1) = (2)^2 = 4$ ;  $u_2^2 = w^2 y_2^1 + b^2 = 2 \cdot 4 + 1 = 9$ ,  $y_2^2 = f^2(u_2^2) = 9$ , похибка  $e_2 = t_2 - y_2^2 = 2 - 9 = -7$ .

Далі виконаємо ініціалізацію зворотного поширення і визначимо чутливість Марквардта:

– вплив похибки на вагу для 2-го шару, якщо  $p_1 = 1$ :

$$\tilde{S}_1^2 = -\dot{F}^2(u_1^2) = -1;$$

– вплив похибки на вагу для 1-го шару:

$$\tilde{S}_1^1 = \dot{F}^1(u_1^1) w^2 S_1^2 = 2u_1^1 \cdot 2 \cdot (-1) = 2 \cdot 1 \cdot 2 \cdot (-1) = -4;$$

– вплив похибки на вагу для 2-го шару, якщо  $p_2 = 2$ :

$$\tilde{S}_2^2 = -\dot{F}^2(u_2^2) = -1;$$

– вплив похибки на вагу для 1-го шару:

$$\tilde{S}_2^1 = \dot{F}^1(u_2^1)w^2 S_2^2 = 2u_2^1 \cdot 2 \cdot (-1) = 2 \cdot 2 \cdot 2 \cdot (-1) = -8;$$

– загальний вплив похибки на 1-й шар:  $\tilde{\mathbf{S}}^1 = [\tilde{S}_1^1 \ \tilde{S}_2^1] = [4 \ -8];$

– загальний вплив похибки на 2-й шар:  $\tilde{\mathbf{S}}^2 = [\tilde{S}_1^2 \ \tilde{S}_2^2] = [-1 \ -1].$

Тепер можна обчислити матрицю Якобі:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \frac{\partial v_1}{\partial x_2} & \frac{\partial v_1}{\partial x_3} & \frac{\partial v_1}{\partial x_4} \\ \frac{\partial v_2}{\partial x_1} & \frac{\partial v_2}{\partial x_2} & \frac{\partial v_2}{\partial x_3} & \frac{\partial v_2}{\partial x_4} \end{bmatrix} = \begin{bmatrix} \frac{\partial e_1}{\partial w^1} & \frac{\partial e_1}{\partial b^1} & \frac{\partial e_1}{\partial w^2} & \frac{\partial e_1}{\partial b^2} \\ \frac{\partial e_2}{\partial w^1} & \frac{\partial e_2}{\partial b^1} & \frac{\partial e_2}{\partial w^2} & \frac{\partial e_2}{\partial b^2} \end{bmatrix};$$

$$J_{11} = \frac{\partial v_1}{\partial x_1} = \frac{\partial e_1}{\partial w^1} = \frac{\partial e_1}{\partial u_1^1} \cdot \frac{\partial u_1^1}{\partial w^1} = \tilde{S}_1^1 \cdot \frac{\partial u_1^1}{\partial w^1} = \tilde{S}_1^1 \cdot y_1^0 = (-4) \cdot (1) = -4;$$

$$J_{12} = \frac{\partial v_1}{\partial x_2} = \frac{\partial e_1}{\partial b^1} = \frac{\partial e_1}{\partial u_1^1} \cdot \frac{\partial u_1^1}{\partial b^1} = \tilde{S}_1^1 \cdot \frac{\partial u_1^1}{\partial b^1} = \tilde{S}_1^1 = -4;$$

$$J_{13} = \frac{\partial v_1}{\partial x_3} = \frac{\partial e_1}{\partial u_1^2} \cdot \frac{\partial u_1^2}{\partial w^2} = \tilde{S}_1^2 \cdot \frac{\partial u_1^2}{\partial w^2} = \tilde{S}_1^2 \cdot y_1^1 = (-1) \cdot (1) = -1;$$

$$J_{14} = \frac{\partial v_1}{\partial x_4} = \frac{\partial e_1}{\partial b^2} = \frac{\partial e_1}{\partial u_1^2} \cdot \frac{\partial u_1^2}{\partial b^2} = \tilde{S}_1^2 \cdot \frac{\partial u_1^2}{\partial b^2} = \tilde{S}_1^2 = -1;$$

$$J_{21} = \frac{\partial v_2}{\partial x_1} = \frac{\partial e_2}{\partial u_2^1} \cdot \frac{\partial u_2^1}{\partial w^1} = \tilde{S}_2^1 \cdot \frac{\partial u_2^1}{\partial w^1} = \tilde{S}_2^1 \cdot y_2^0 = (-8) \cdot (2) = -16;$$

$$J_{22} = \frac{\partial v_2}{\partial x_2} = \frac{\partial e_2}{\partial b^1} = \frac{\partial e_2}{\partial u_2^1} \cdot \frac{\partial u_2^1}{\partial b^1} = \tilde{S}_2^1 \cdot \frac{\partial u_2^1}{\partial b^1} = \tilde{S}_2^1 = -8;$$

$$J_{23} = \frac{\partial v_2}{\partial x_3} = \frac{\partial e_2}{\partial u_2^2} \cdot \frac{\partial u_2^2}{\partial w^2} = \tilde{S}_2^2 \cdot \frac{\partial u_2^2}{\partial w^2} = \tilde{S}_2^2 \cdot y_2^1 = (-1) \cdot (4) = -4;$$

$$J_{24} = \frac{\partial v_2}{\partial x_4} = \frac{\partial e_2}{\partial b^2} = \frac{\partial e_2}{\partial u_2^2} \cdot \frac{\partial u_2^2}{\partial b^2} = \tilde{S}_2^2 \cdot \frac{\partial u_2^2}{\partial b^2} = \tilde{S}_2^2 = -1.$$

Тому матриця Якобі має такий вигляд:  $\mathbf{J}(\mathbf{x}) = \begin{bmatrix} -4 & -4 & -1 & -1 \\ -16 & -8 & -4 & -1 \end{bmatrix}.$

## Контрольні запитання

1. Визначте недоліки методу навчання зворотного поширення.
2. Опишіть евристичні модифікації методу зворотного поширення похибки: з використанням імпульсу, із застосуванням змінного значення коефіцієнта навчання.
3. Опишіть модифікації методу зворотного поширення як методи, основані на стандартних числових методах оптимізації: метод спряжених градієнтів, метод Левенберга–Марквардта.
4. Упишіть один із методів лінійної оптимізації, який може застосувати метод спряжених градієнтів.

## Задачі для самостійного розв'язання

**Задача 8.1.** Навчити НМ, зображену на рис. 8.8, на множині пар вхід–ціль вигляду  $\{p_1 = -2, t_1 = 0,8\}; \{p_2 = 2, t_2 = 1\}$ , де кожна пара подається на вхід з однаковою ймовірністю. Написати програму (у вигляді *m*-файлу) побудови графіка контурних ліній для середньоквадратичної похибки, що описує ефективність функціонування мережі.

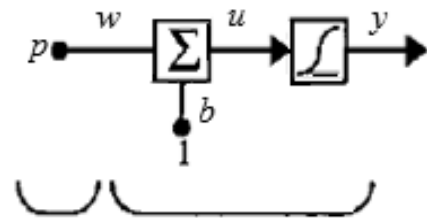


Рис. 8.8. Одношарова сигмоїдна мережа:  $y = \text{liogsig}(w^T p + b)$

**Задача 8.2.** Виконати алгоритм SDBP для задачі 8.1 при пакетному та непакетному режимах, використавши початкове наближення  $w(0) = 0$ ;  $b(0) = 0.5$ .

**Задача 8.3.** Нехай задано квадратичну функцію  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} \mathbf{x} + [4 \quad 4] \mathbf{x}$ . Застосувавши метод найшвидшого спуску із використанням імпульсу, визначити мінімум цієї функції.

**I.** Використовуючи метод, описаний в прикл. 8.3, визначити значення моменту  $\gamma$  для швидкості навчання  $\alpha = 0,2$ , (або  $\alpha = 20$ ) за якого алгоритм буде стабільним.

**II.** Написати програму (у вигляді *m*-файлу) побудови на контурних лініях функції  $F(\mathbf{x})$  траєкторії алгоритму для значень  $\alpha$  та  $\gamma$ , визначених у п. I, використавши початкове наближення:  $\mathbf{x}_0 = [-1 \quad -2,5]^T$ .

**Задача 8.4.** Для функції із задачі 8.3 виконати три ітерації алгоритму зі змінною швидкістю навчання, використавши початкове наближення  $\mathbf{x}_0 = [-1 \quad -2,5]^T$ . Написати програму (у вигляді *m*-файлу) побудови

на контурних лініях функції  $F(\mathbf{x})$  траєкторії алгоритму для параметрів  $\alpha = 0,4$ ;  $\gamma = 0,1$ ;  $\eta = 1,5$ ;  $\rho = 0,5$ ;  $\zeta = 5\%$ .

**Задача 8.5.** Для функції  $F(\mathbf{x}) = \mathbf{x}_1^2 + 25\mathbf{x}_2^2$  (або із задачі 8.3) виконати одну ітерацію алгоритму спряжених градієнтів, використавши початкове наближення  $\mathbf{x}_0 = [-1 \ -2,5]^T$ . Для лінійної мінімізації використати інтервал, визначений за допомогою оцінки функції. Звузити цей інтервал за допомогою методу золотого перетину. Накреслити траєкторію пошуку мінімуму функції  $F(\mathbf{x})$  на її контурному графіку.

**Задача 8.6.** Для апроксимації функції:  $g(p) = 1 + \sin\left(\frac{\pi}{4}p\right)$ ,  $-2 \leq p \leq 2$ , було використано мережу (рис. 8.9) з такими початковими параметрами:  $\mathbf{w}^1(0) = \begin{bmatrix} -0,27 \\ -0,41 \end{bmatrix}$ ;  $\mathbf{b}^1(0) = \begin{bmatrix} -0,48 \\ -0,13 \end{bmatrix}$ ;  $\mathbf{w}^2(0) = \begin{bmatrix} 0,09 \\ -0,17 \end{bmatrix}$ ;  $b^2(0) = 0,48$ . Для навчання мережі було використано значення функції  $g(p)$  у точках  $p_1 = 1$ ;  $p_2 = 0$ . Визначити матрицю Якобі для першого кроку алгоритму Левенберга–Марквардта.

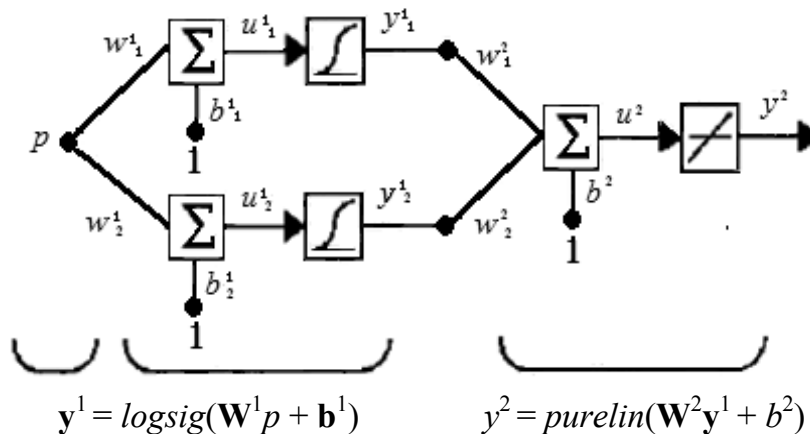


Рис. 8.9. Мережа, що апроксимує функцію

**Задача 8.7.** Показати, що для лінійної мережі за  $\mu = 0$  алгоритм LMВР збігається до оптимального розв'язку за одну ітерацію.

**Задача 8.8.** Нехай задано функцію  $g(p) = 1 + \sin\left(\frac{\pi}{8}p\right)$ ,  $-2 \leq p \leq 2$ .

Написати програму (у вигляді *m*-файлу), яка реалізує алгоритми МОВР, VLВР, CGВР, LMВР для мережі, зображеної на рис. 8.9, зі структурою 1-2-1, що апроксимує цю функцію. Як початкові значення ваги та зсуву вибрати випадкові числа, рівномірно розподілені на відрізку  $[-0,5, 0,5]$ .

## Розділ 9. НЕЙРОННІ МЕРЕЖІ НА ОСНОВІ РАДІАЛЬНИХ БАЗИСНИХ ФУНКЦІЙ

### 9.1. Задача інтерполяції

Розглянемо підхід, у межах якого розроблення НМ розглядається як задача апроксимації кривої за точками у просторі високої розмірності. При цьому навчання еквівалентне визначенню такої поверхні у багатовимірному просторі, яка найбільш точно відповідає даним навчання (критерій «найкращої відповідності» вибирається в деякому статистичному сенсі), а узагальнення еквівалентне використанню цієї поверхні для інтерполяції даних тестування. Такий підхід покладено в основу методу радіальних базисних функцій (РБФ), сутність якого полягає у традиційній інтерполяції в багатовимірному просторі.

Базова архітектура РБФ-мереж передбачає наявність двох шарів, де перший прихований шар виконує нелінійне перетворення вхідного простору у прихований (у більшості реалізацій прихований простір має більш високу розмірність, ніж вхідний простір); другий шар – лінійний. Нейрони прихованого шару реалізують набір «функцій», які формують довільний «базис» для розкладання вхідних образів (векторів). Відповідні перетворення називають радіальними базисними функціями (radial-basis function).

Розглянемо мережу прямого поширення з одним прихованим шаром і одним вихідним шаром, який містить один нейрон (один нейрон у вихідному шарі обраний для спрощення викладок без втрати спільності). Ця мережа призначена для нелінійного відображення вхідного простору у прихований, за яким слідує лінійне відображення прихованого простору у вихідний.

Нехай  $R$  – розмірність вхідного простору. Тоді мережа реалізує відображення  $R$ -вимірного вхідного простору в одновимірний вихідний простір  $F: \mathbb{R}^R \rightarrow \mathbb{R}$ . Відображення  $F$  можна розглядати як гіперповерхню (графік)  $\Gamma \subset \mathbb{R}^{R+1}$  (аналогічно тому, як відображення  $F: \mathbb{R} \rightarrow \mathbb{R}$  зображується у двовимірному просторі у вигляді параболи  $F(x) = x^2$ ). Поверхня  $\Gamma$  є багатовимірним графіком зміни вихідного сигналу залежно від вхідного. На практиці поверхня  $\Gamma$  залишається невідомою, а на дані навчання накладається шум. У цьому випадку етапи навчання й узагальнення можна описати таким чином:

1) *на етапі навчання* поверхня  $\Gamma$  оптимізується на підставі відомих точок даних, які подаються на вхід мережі у формі прикладів типу вхід–вихід;

2) *етап узагальнення* є рівносильним інтерполяції на інтервалах між точками даних. Ця інтерполяція здійснюється на обмеженій поверхні, яка згенерована процедурою підбору як оптимальна апроксимація істинної поверхні  $\Gamma$ .

Отже, маємо *теорію інтерполяції функції багатьох змінних в багатовимірному просторі*.

Задачу інтерполяції в її первісному значенні можна сформулювати так. Для заданої множини з  $N$  точок  $\{\mathbf{p}_i \in \mathcal{R}^R \mid i = 1, 2, \dots, N\}$  і відповідної множини з  $N$  дійсних чисел  $\{t_i \in \mathcal{R} \mid i = 1, 2, \dots, N\}$  визначити функцію  $F: \mathcal{R}^R \rightarrow \mathcal{R}$ , яка задовольняє таку умову інтерполяції:

$$F(\mathbf{p}_i) = t_i, i = 1, 2, \dots, N. \quad (9.1)$$

Для визначеної таким чином задачі поверхня інтерполяції (тобто функція  $F$ ) проходить через усі точки прикладів навчання. Метод радіальних базисних функцій зводиться до вибору функції  $F$  вигляду

$$y = F(\mathbf{p}) = \sum_{i=1}^N w_i \phi_i(\|\mathbf{p} - \mathbf{c}_i\|), \quad (9.2)$$

де  $\{\phi_i(\|\mathbf{p} - \mathbf{c}_i\|), i = 1, 2, \dots, N\}$  – множина із  $S^1 = N$  довільних (зазвичай нелінійних) функцій, які називають радіальними базисними функціями;  $\|\bullet\|$  – норма, зазвичай Евкліда. При цьому кількість нейронів шару із радіальними базисними функціями дорівнює кількості вхідних векторів (на практиці  $S^1 \ll N$ ). Для спрощення можна вибрати один тип РБФ, тому вираз (9.2) можна записати у вигляді

$$y = F(\mathbf{p}) = \sum_{i=1}^N w_i \phi(\|\mathbf{p} - \mathbf{c}_i\|). \quad (9.3)$$

Відомі точки даних  $\mathbf{p}_i \in \mathcal{R}^R, i = 1, 2, \dots, N$  вважають центрами РБФ. Підставляючи у вираз (9.3) умову інтерполяції (9.1) за  $R = N$ , отримаємо систему лінійних рівнянь для невідомих вагових коефіцієнтів  $\{w_i\}$  вигляду

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}, \quad (9.4)$$

де  $\phi_{ji} = \phi(\|\mathbf{p}_j - \mathbf{p}_i\|), j, i = 1, 2, \dots, N$ . Нехай  $\mathbf{t} = \begin{bmatrix} t_1 \\ \dots \\ t_N \end{bmatrix}$ ;  $\mathbf{w} = \begin{bmatrix} w_1 \\ \dots \\ w_N \end{bmatrix}$  – вектор

цільових значень та ваговий вектор відповідно,  $\mathbf{G}$  – матриця розмірністю  $[N \times N]$  з елементами  $\phi_{ji}$ :  $\mathbf{G} = \{\phi_{ji} \mid (j, i) = 1, 2, \dots, N\}$ , назвемо її матрицею



інтерполяції. Тепер вираз (9.4) можна переписати у такому вигляді:

$$\mathbf{G}\mathbf{w} = \mathbf{t}. \quad (9.5)$$

Припустимо, що матриця  $\mathbf{G}$  є несингулярною, тобто її визначник не дорівнює нулю (отже, для неї існує обернена матриця  $\mathbf{G}^{-1}$ ), тому розв'язок рівняння (9.5) щодо вагового вектора  $\mathbf{w}$  має вигляд  $\mathbf{w} = \mathbf{G}^{-1}\mathbf{t}$ . При цьому виникає запитання: як переконатися у несингулярності матриці  $\mathbf{G}$ ? Для великого класу РБФ за певних умов відповідь на це запитання дає така теорема.

*Теорема Мічеллі.* Нехай  $\{\mathbf{p}_i \in \mathbb{R}^R\}$ ,  $i = 1, 2, \dots, N$  – множина різних точок. Тоді матриця інтерполяції  $\mathbf{G}$  розмірністю  $[N \times N]$  з елементами  $\phi_{ji} = \phi(\|\mathbf{p}_j - \mathbf{p}_i\|)$  є несингулярною, якщо РБФ серед інших має вигляд:

1) функції Гаусса  $\phi(p) = e^{-(p^2/2\sigma^2)}$  для  $\sigma > 0, p \in \mathbb{R}$ ;

2) багатоквадратичної функції  $\phi(p) = (c^2 + p^2)^{1/2}$  для  $c > 0, p \in \mathbb{R}$ ;

3) оберненої багатоквадратичної функції  $\phi(p) = \frac{1}{(c^2 + p^2)^{1/2}}$  для  $c > 0,$

$p \in \mathbb{R}$ .

Щоб вказані РБФ були несингулярними, усі точки  $\{\mathbf{p}_i \in \mathbb{R}^R\}$ ,  $i = 1, 2, \dots, N$  мають відрізнятися. Для визначених значень  $\mathbf{p}_i \in \mathbb{R}^R$ ,  $i = 1, 2, \dots, N$  проблема інтерполяції може бути переписана як розв'язання рівняння (9.5). Чи можна і як саме для заданого  $\mathbf{t}$  модифікувати матрицю  $\mathbf{G}$ , щоб визначити точний розв'язок  $\mathbf{w}$ , який задовольняє рівняння (9.5) або мінімізує похибку вигляду  $\|\mathbf{t} - \mathbf{G}\mathbf{w}\|^2$  (при цьому  $\mathbf{w}$  є оптимальною точкою мінімуму похибки)? Оптимальний розв'язок  $\mathbf{w}^*$ , який мінімізує цю похибку, можна одержати за формулою  $\mathbf{w}^* = \mathbf{G}^+\mathbf{t}$ , де  $\mathbf{G}^+$  – узагальнена матриця, обернена до матриці  $\mathbf{G}$ :  $\mathbf{G}^+ \cong [\mathbf{G}^T \mathbf{G}]^{-1} \mathbf{G}^T$ . Для визначення матриці  $\mathbf{G}^+$  можна використати сингулярне розкладання, яке надає полегшений спосіб визначення матриці  $\mathbf{G}^+$ .

**Сингулярне розкладання** (Singular Value Decomposition – SVD). Для будь-якої не виродженої матриці  $\mathbf{G}$  розмірністю  $[N \times S^1]$  сингулярне розкладання задають таким чином:

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T, \quad (9.6)$$

де  $\mathbf{U}$  – матриця розмірністю  $[N \times S^1]$ , стовпці якої є власними векторами матриці  $\mathbf{G}\mathbf{G}^T$ ;  $\mathbf{\Lambda}$  – діагональна матриця розмірністю  $[S^1 \times S^1]$ , елементи якої називають сингулярними, вони мають вигляд квадратного кореня із власних значень матриці  $\mathbf{G}^T \mathbf{G}$ ;  $\mathbf{V}$  – матриця розмірністю  $[S^1 \times S^1]$ , стовпці якої є власними векторами матриці  $(\mathbf{G}^T \mathbf{G})$ . Матриці  $\mathbf{U}$  та  $\mathbf{V}$  – ортонормовані, тобто  $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ ,  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ . Підставляючи рівняння (9.6) у рівняння

$(G^T t) = (G^T G w)$ , отримаємо:  $V \Lambda U^T U \Lambda V^T w = V \Lambda U^T t$ . Використовуючи рівняння  $V^T V = U^T U = I$ , ваговий вектор можна визначити за формулою

$$w = V \Lambda^{-1} U^T t = \sum_{i=1}^{s^1} v_i \left( \frac{u_i^T t}{\lambda_i} \right), \lambda_i \neq 0, \text{ де } u_i \text{ та } v_i - i\text{-й вектор-стовпець матриць } U \text{ та } V \text{ відповідно; } \lambda_i - i\text{-те сингулярне значення (діагональний елемент матриці } \Lambda).$$

риць  $U$  та  $V$  відповідно;  $\lambda_i$  –  $i$ -те сингулярне значення (діагональний елемент матриці  $\Lambda$ ).

## 9.2. Мережа з використанням радіальних базисних функцій Гаусса

Найчастіше як радіальну функцію використовують функцію Гаусса (ГРБФ). Якщо її центр розміщено в точці  $c_i$ , вона може бути визначена як (рис. 9.1)

$$\phi(p) = \phi(\|p - c_i\|) = \exp\left(-\frac{\|p - c_i\|^2}{2\sigma_i^2}\right),$$

де  $\sigma_i$  – ширина функції. При цьому формула (9.3) набуває вигляду

$$F(p) = \sum_{i=1}^N w_i \exp\left(-\frac{\|p - c_i\|^2}{2\sigma_i^2}\right). \quad (9.7)$$

Для спрощення викладу часто беруть умову  $\sigma_i = \sigma \forall i$ . Незважаючи на те, що визначені таким чином функції мають дещо обмежений вигляд, вони залишаються універсальними апроксиматорами.

**Зважена норма.** Норма в наближеному розв'язку (9.7) зазвичай розглядається як норма Евкліда.

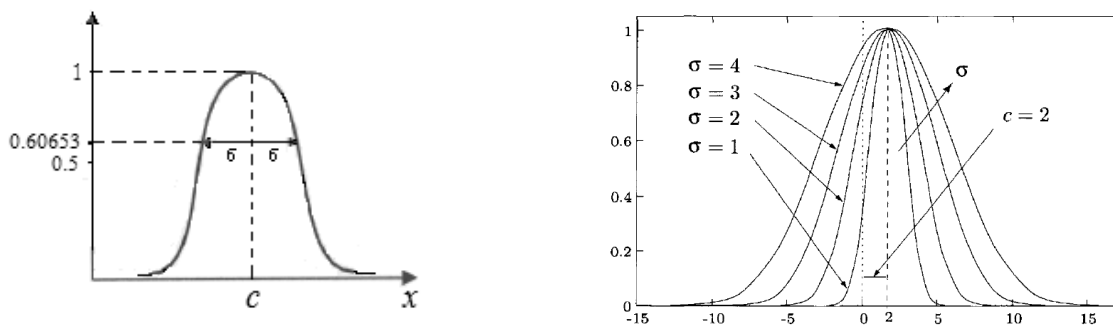


Рис. 9.1. Радіальна базисна функція Гаусса:

$$\phi(p) = \phi(\|p - c_i\|) = \exp\left(-\frac{\|p - c_i\|^2}{2\sigma_i^2}\right), \text{ якщо } c = 2, \sigma = 1, 2, 3, 4$$

Однак, якщо окремі елементи вхідного вектора  $p$  належать різним класам, то доцільно використати загальну зважену норму вигляду

$\|\mathbf{p}\|_{\mathbf{K}}^2 = (\mathbf{K}\mathbf{p})^T (\mathbf{K}\mathbf{p}) = (\mathbf{p}\mathbf{K}^T \mathbf{K}\mathbf{p})$ , де  $\mathbf{K}$  – матриця зваженої норми розмірністю  $[R \times R]$ ;  $R$  – розмірність вектора входу  $\mathbf{p}$ . Використовуючи визначення зваженої норми, вираз для апроксимації (9.7) можна переписати у вигляді

$$F(\mathbf{p}) = \sum_{i=1}^N w_i \exp\left(-(\mathbf{p} - \mathbf{c}_i)^T \mathbf{K}^T \mathbf{K} (\mathbf{p} - \mathbf{c}_i)\right) = \sum_{i=1}^N w_i \exp\left(-(\mathbf{p} - \mathbf{c}_i)^T \Sigma^{-1} (\mathbf{p} - \mathbf{c}_i)\right).$$

Вираз

$$\exp\left(-\frac{1}{2}(\mathbf{p} - \mathbf{c}_i)^T \Sigma^{-1} (\mathbf{p} - \mathbf{c}_i)\right) \quad (9.8)$$

відображає багатовимірний розподіл Гаусса з вектором середнього значення  $\mathbf{c}_i$  і матрицею коваріації  $\Sigma$ :  $\frac{1}{2}\Sigma^{-1} = \mathbf{K}^T \mathbf{K}$ , де матриця коваріації має такий вигляд:

1)  $\Sigma = \sigma^2 \mathbf{I}$ , де  $\mathbf{I}$  – одинична матриця;  $\sigma^2$  – загальна дисперсія: у цьому випадку (9.8) зображує гіперсферу із центром  $\mathbf{c}_i$  і радіусом  $\sigma$ ;

2)  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_R^2)$ , де  $\sigma_j^2$  – дисперсія  $j$ -го елемента вхідного вектора  $\mathbf{p}$ ;  $j = 1, 2, \dots, R$ : у цьому випадку (9.8) відображає гіпереліпсоїд, півосі якого збігаються з осями вхідного простору і мають довжину  $\sigma_j$ .

Симетрична РБФ формує однакове значення для всіх входів множини  $P$ , розміщених на гіпереліпсоїді з центром у точці  $\mathbf{c}_i$  та осями, визначеними матрицею масштабування  $\mathbf{K}$  (рис. 9.2, б).

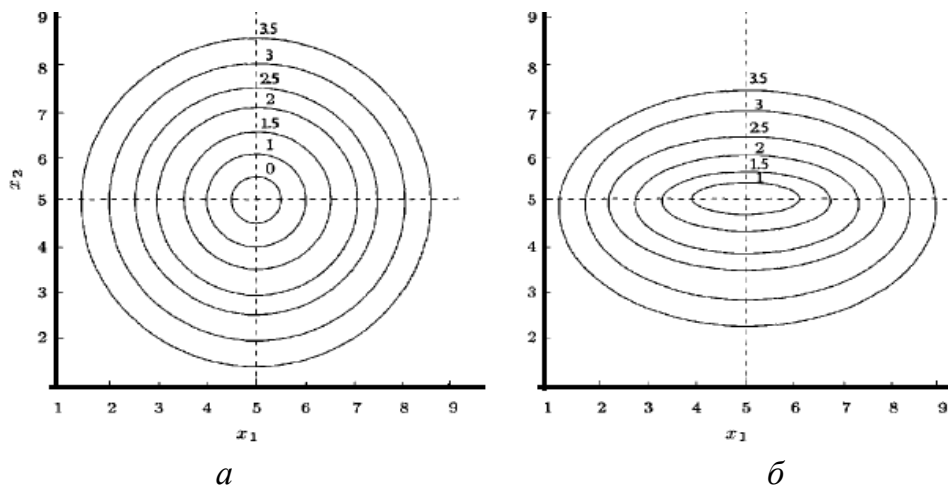


Рис. 9.2. Пиклад РБФ вигляду  $\|\mathbf{p} - \mathbf{c}\|^2$  у двовимірному просторі:

а –  $\|\mathbf{p} - \mathbf{c}\|^2$ , якщо  $\mathbf{c} = [5 \ 5]^T$ ; б –  $\|\mathbf{p} - \mathbf{c}\|_{\mathbf{K}}^2$ , якщо  $\mathbf{K} = \text{diag} [4 \ 3]$ ,  $\mathbf{c} = [5 \ 5]^T$

Радіальні НМ з використанням ГРБФ – це двошарові мережі, де перший шар об'єднує ГРБФ, а другий виконує суммування (рис. 9.3).

Нехай  $\mathbf{p} = [p_1 \dots p_R]^T$ ;  $\mathbf{t} = [t_1 \dots t_N]^T$  – це вхід і вихід мережі відповідно,  $\mathbf{u} = [u_1 \dots u_{S^1}]^T$  –  $S^1$  вихід із  $S^1$  гауссових вузлів. Радіальну базисну функцію Гауса  $\phi_i$  з нормою ваги знаходять за формулою  $\phi(\|\mathbf{p} - \mathbf{c}_i\|_{\mathbf{K}_i}) = \exp(-0,5d(\mathbf{p}, \mathbf{c}_i, \Sigma_i))$ , де  $d(\mathbf{p}, \mathbf{c}_i, \Sigma_i) \equiv \|\mathbf{p} - \mathbf{c}_i\|_{\mathbf{K}_i}^2 = (\mathbf{p} - \mathbf{c}_i)^T \Sigma_i^{-1} (\mathbf{p} - \mathbf{c}_i)$ ,  $\mathbf{c}_i \in \mathbb{R}^R$  – вектор центру.

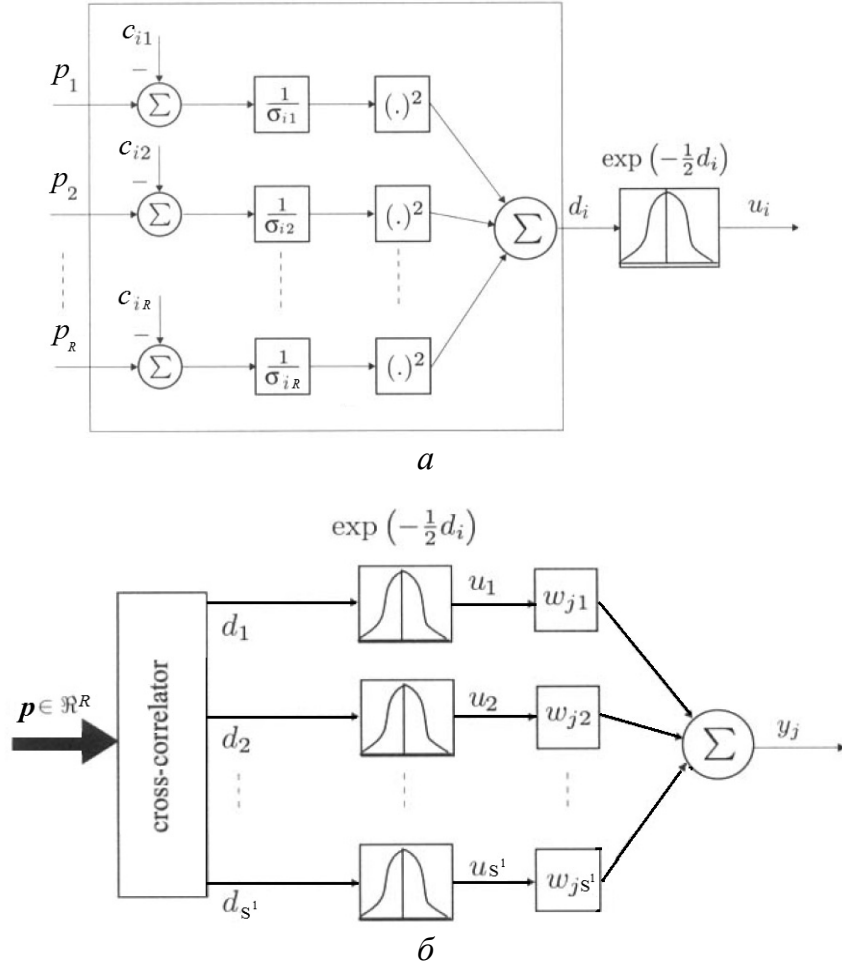


Рис. 9.3. Модель НМ з використанням радіальної функції Гауса:

*a* – Cross-correlator: з'єднання входів  $p_1, \dots, p_R$  із  $i$ -м нейроном прихованого шару,

$$d_i = \sum_{k=1}^R \left[ \left( \frac{p_k - c_{ik}}{\sigma_{ik}} \right) \right]^2, \quad u_i = \exp\left(-\frac{1}{2}d_i\right), \quad i = 1, \dots, S^1;$$

*б* – Зв'язок між нейронами прихованого шару та  $j$ -м нейроном вихідного шару

$$\text{Розпишемо } d(\mathbf{p}, \mathbf{c}_i, \Sigma_i) \text{ як } d(\mathbf{p}, \mathbf{c}_i, \Sigma_i) = \sum_{j=1}^R \sum_{k=1}^R h_{ijk} (p_j - c_{ij})(p_k - c_{ik}),$$

де  $c_{ij}$  –  $j$ -й елемент  $\mathbf{c}_i$ ;  $h_{ijk}$  –  $(j, k)$ -й елемент матриці  $\Sigma_i$ . Елемент  $h_{ijk}$  можна зобразити як співвідношення коефіцієнта кореляції  $k_{ijk}$  й добутку приме-

жових стандартних відхилень  $\sigma_{ij}$  і  $\sigma_{ik}$  таким чином:  $h_{ij} = k_{ijk} / \sigma_{ij} \sigma_{ik}$ , де  $\sigma_{ij}$  – додатне дійсне число,  $k_{ijk} = 1$ , якщо  $j = k$ , і  $|k_{ijk}| < 1$ , якщо  $j \neq k$ . Замість наведеної формули визначення  $h_{ijk}$  можна використати таку:

$$h_{ijk} = \begin{cases} 1/\sigma^2, & \text{якщо } j = k; \\ 0 & \text{в інших випадках,} \end{cases} \quad \text{де параметр } \sigma^2 \text{ керує процесом зміни шири-$$

ни функції Гаусса. Елементи матриці  $\Sigma_i$  – це додатні числа. Таким чином, взаємозв'язок вхід–вихід у радіальній мережі на основі функції Гаусса може мати множину виходів  $y_j$ , описаних у вигляді

$$u_i = \exp\left(-\frac{1}{2} \sum_{k=1}^R \left[\frac{p_k - c_{ik}}{\sigma_{ik}}\right]^2\right), \quad i = 1, 2, \dots, S^1;$$

$$y_j = \sum_{i=1}^{S^1} w_{ji} u_i = \sum_{i=1}^{S^1} w_{ji} \exp\left(-\frac{1}{2} \sum_{k=1}^R \left[\frac{p_k - c_{ik}}{\sigma_{ik}}\right]^2\right), \quad j = 1, 2, \dots, m, \quad (9.9)$$

де  $u_i$  –  $i$ -й вихід прихованих нейронів, описаних функцією Гаусса, яка формує гіпереліпсоїд у  $R$ -вимірному просторі  $\mathcal{R}^R$ ; параметри  $c_{ik}$  і  $\sigma_{ik}^2$  – центр і дисперсія  $i$ -ї ГРБФ (визначають геометричну форму й місце перебування гіпереліпсоїда в  $\mathcal{R}^R$ );  $S^1$  – кількість ГРБФ, які розраховують відстань Евкліда між центром і вектором входу мережі.

Зазвичай вихідний шар складається із множини лінійних нейронів. Позначимо коваріаційну матрицю як  $\Sigma_i \cong \text{diag}[\sigma_{i1}^2, \sigma_{i2}^2, \dots, \sigma_{in}^2]$ , тоді рівняння (9.9) можна переписати у вигляді

$$y_j = \sum_{i=1}^{S^1} w_{ji} \exp\left(-\frac{1}{2} (\mathbf{p} - \mathbf{c}_i)^T \Sigma_i^{-1} (\mathbf{p} - \mathbf{c}_i)\right), \quad j = 1, 2, \dots, m.$$

У разі введення вектора  $\mathbf{p}$  кожен нейрон прихованого шару видасть значення відстані вектора входу до вагового вектора. Таким чином, радіальні базисні нейрони з ваговим вектором, який значно відрізняється від вектора входу  $\mathbf{p}$ , будуть мати виходи близькі до нуля, і їх вплив на виходи лінійних нейронів буде незначним. Навпаки, радіальний базисний нейрон з ваговим вектором, близьким до вектора входу  $\mathbf{p}$ , буде мати вихід, значення якого близьке до одиниці – це значення і буде подане на лінійний нейрон з вагою, що відповідає вихідному шару. Отже, якщо один радіальний базисний нейрон має на виході значення «1», а всі інші – значення «0» (або дуже близьке до нуля), то вихід лінійного шару буде дорівнювати вазі активного вихідного нейрона. Однак це винятковий випадок, зазвичай вихід формують кілька нейронів з різними значеннями ваги.

### 9.3. Мережа з використанням радіальних базисних функцій Гаусса як універсальний апроксиматор

У 1981 г. А. Браун (А. С. Brown) показав, що РБФ-мережі дозволяють відобразити будь-яку функцію із замкненої обмеженої області  $\mathcal{R}^R$  у таку ж область із  $\mathcal{R}^1$ .

*Теорема про універсальну апроксимацію для нелінійного відображення вхід-вихід для мереж з використанням ГРБФ [12].* Нехай  $H: \mathcal{R}^R \rightarrow \mathcal{R}$  – обмежена, неперервна та інтегрована функція, така, що  $\int_{\mathcal{R}^R} H(\mathbf{p}) d\mathbf{p} \neq 0$  і  $H_G$  –

множина РБФ-мереж, які виконують відображення  $F: \mathcal{R}^R \rightarrow \mathcal{R}$  вигляду

$$F(\mathbf{p}) = \sum_{i=1}^{S^1} w_i G\left(\frac{\mathbf{p} - \mathbf{c}_i}{\sigma}\right), \text{ де } G\left(\frac{\mathbf{p} - \mathbf{c}_i}{\sigma}\right) = \begin{bmatrix} \phi_{11} & \dots & \phi_{1S^1} \\ \dots & \dots & \dots \\ \phi_{N1} & \dots & \phi_{NS^1} \end{bmatrix}; \quad \phi_{ji} = \exp\left(-\frac{\|\mathbf{p}_j - \mathbf{c}_i\|_{\mathbf{K}_i}^2}{2\sigma^2}\right),$$

$\sigma > 0$ ;  $w_i \in \mathcal{R}$ ;  $\mathbf{c}_i \in \mathcal{R}^R$  для  $i = 1, \dots, S^1$ . Тоді виконується теорема про універсальну апроксимацію РБФ-мереж:

- для будь-якої неперервної функції  $f(\mathbf{p})$  знайдеться РБФ-мережа із множиною центрів  $\{\mathbf{c}_i\}_{i=1}^{S^1}$  і загальною шириною  $\sigma > 0$ , така, що функція  $F(\mathbf{p})$ , яку реалізує мережа, буде близькою до  $f(\mathbf{p})$  за нормою  $L_k$ ,  $k \in [1, \infty]$ .

Теорема формує теоретичний базис для побудови РБФ-мереж.

### 9.4. Алгоритми навчання радіальних мереж на основі функцій Гаусса

Було показано, що мережі з використанням ГРБФ здатні до апроксимації неперервних функцій, визначених на компактній множині (у разі задовільної похибки апроксимації). Процес апроксимації виконується завдяки навчанню мережі таким чином, щоб кількість нейронів прихованого шару і відповідні параметри мережі зменшували похибку апроксимації. Існують різні алгоритми використання ГРБФ-мереж, у більшості з них задачу розбивають на два етапи:

- 1) визначення параметрів ГРБФ (кількості базисних функцій, точок центру та дисперсії);

- 2) навчання лінійних нейронів вихідного шару, що означає визначення параметрів ваги.

Навчання у прихованому шарі, зазвичай, здійснюється з використанням таких елементів.

1. *Випадковий вибір центрів* – найпростіше використати множину прикладів навчання як РБФ-центри. Потім можна оцінити параметр  $\sigma$ :

1) як середню відстань між сусідніми центрами (на значення  $\sigma_j$  впливає відстань між  $j$ -м центром  $c_j$  та його  $P$  найближчими сусідами, при цьому  $\sigma_j$  знаходять за формулою  $\sigma_j = \sqrt{\frac{1}{P} \sum_{k=1}^P \|c_j - c_k\|^2}$ );

2) як  $\sigma = \frac{d_{\max}}{\sqrt{2S^1}}$ , де  $S^1$  – кількість центрів;  $d_{\max}$  – максимальна відстань

між вибраними центрами; при цьому параметр ваги вихідного шару – єдиний параметр, який налаштовується у процесі навчання мережі (найпростіше їх налаштувати на основі методу  $\mathbf{w} = \mathbf{G}^+ \mathbf{t}$ ).

2. *Вибір центрів на основі самоорганізації* (наприклад, *кластеризації*) – РБФ-центри можна визначити на основі процедури кластеризації, наприклад,  $k$ -середніх, а параметр розкиду  $\sigma$  можна отримати із матриці коваріації, взятої з прикладів навчання кожного кластеру.

**Метод  $k$ -кластеризації як складова навчання РБФ-мереж.** Для визначення значень центрів ГРБФ у прихованому шарі можна використати численні алгоритми кластеризації. Найпростіший спосіб полягає у виборі випадковим чином вектора зі списку даних навчання, однак це має бути виконано таким чином, щоб кількість прихованих вузлів Гаусса була відносно невеликою для покриття всіх векторів входу. Один із популярних способів вибору векторів  $\mathbf{c}_i$  – метод  $k$ -кластеризації. Основне завдання цього алгоритму – групування досліджуваних даних на кластери і подальший вибір їх центрів відповідно до природного визначення центрів тяжіння (наприклад, щодо відстані Евкліда). Кожен центр кластера пов'язаний з одним із прихованих вузлів Гаусса.

*Алгоритм 1 (стандартний алгоритм  $k$ -кластеризації).* Розглянемо кроки алгоритму.

*Крок 1.* Визначають кількість кластерів:  $S^1 < N$ .

*Крок 2.* Обирають із множини навчання:  $\mathbf{p}_1, \dots, \mathbf{p}_N$   $S^1$  даних як центральних векторів  $\mathbf{c}_i = \mathbf{p}_j$ ,  $j = 1, 2, \dots, S^1$ .

*Крок 3.* Додають нові вектори  $\mathbf{p}_i$  ( $i = S^1 + 1, S^1 + 2, \dots, N$ ) в один із кластерів за критерієм найменшої відстані, що означає:  $\mathbf{p}_i$  належить  $j$ -му кластеру, якщо  $\|\mathbf{p}_i - \mathbf{c}_{j^*}\| = \min_j \|\mathbf{p}_i - \mathbf{c}_j\|$ ,  $j = 1, 2, \dots, S^1$ .

*Крок 4.* Перевизначають вектори  $\mathbf{c}_j = \frac{1}{N_j} \sum_{i \in C_j} \mathbf{p}_i$ ;  $j = 1, 2, \dots, S^1$ ;  $N_j$  –

кількість даних навчання, які належать  $j$ -му кластеру із центром у точці  $\mathbf{c}_j$ .

Результатом виконання алгоритму є множина центрів  $\mathbf{c}_1, \dots, \mathbf{c}_{S^1}$ . Як тільки алгоритм кластеризації завершений, може бути визначена ди-

сперсія, наприклад за формулою  $\sigma_{ij} = \frac{1}{N_i} \sum_{k \in C_i} (p_{kj} - c_{ij})^2$ ;  $i = 1, \dots, S^1$ ,

$j = 1, \dots, R$ , де  $p_{kj}$  –  $j$ -й елемент вектора даних  $\mathbf{p}_k$ . Значення  $\sigma_{ij}$  є мірою розподілу даних, пов'язаних з кожним вузлом. Описаний вище метод  $k$ -кластеризації має недолік, який полягає в тому, що задані точки (вектори), що належать прихованим кластерам, можуть не бути зв'язаними з новими кластерами з того моменту, як центри кластерів були оновлені (змінені). На рис. 9.4 зображено приклад, коли вектор даних  $\mathbf{d}_1$  спочатку належить кластеру 1, але після оновлення центрів переходить до кластера 2 (з того часу, як він стає найближчим до нового центру  $\mathbf{c}_2$ ).

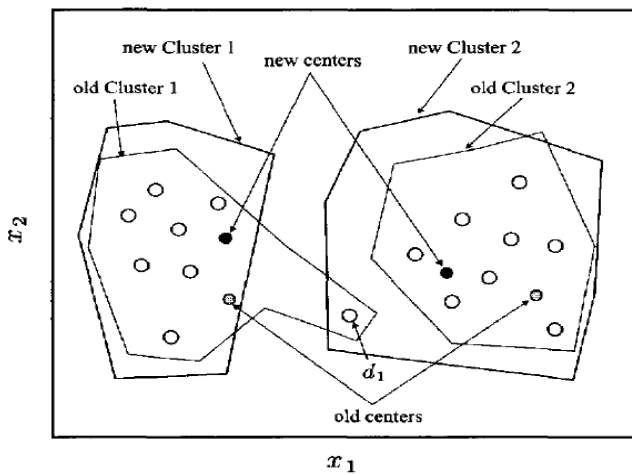


Рис. 9.4. Приклад роботи алгоритму  $k$ -кластеризації

Відомі модифікації алгоритму  $k$ -кластеризації Андерберга (1973 р.) та Спата (1980 р.). Більшість алгоритмів збільшують точність навчання за рахунок додаткових вимог.

Кількість кластерів в алгоритмах  $k$ -кластеризації обирають заздалегідь, однак алгоритм настільки швидко працює, що його можна повторити з використанням різних значень  $S^1$ .

**Метод навчання градієнтного спуску** [9; 15]. Якщо, подібно до алгоритму зворотного поширення для БП, розглянути всі параметри ГРБФ-мережі (такі як вага, центри і дисперсія) як невідомі, то для оновлення її параметрів можна використати метод градієнтного спуску.

Нехай маємо пари даних вхід–вихід  $\{\mathbf{p}(k), t(k)\}$ ,  $k = 1, \dots, N$ . Кожна

РБФ визначається як  $\phi_{ki} = \phi_i(\mathbf{p}_k) = \exp\left(-\frac{1}{2}u_{ik}\right)$ , де  $u_{ik} = \sum_{j=1}^R \left(\frac{p_{jk} - c_{ij}}{\sigma_{ij}}\right)^2$ .

Спочатку визначимо значення цільової функції:

$$E = \frac{1}{2} \sum_{k=1}^N (y_k - t_k)^2 = \frac{1}{2} \sum_{k=1}^N \left( \sum_{i=1}^{S^1} w_i \phi_i(\mathbf{p}_k) - t_k \right)^2.$$

У результаті її диференціювання отримаємо:

$$\frac{\partial E}{\partial c_{ij}} = \sum_{k=1}^N \left( (y_k - t_k) w_i \exp\left(-\frac{1}{2}u_{ik}\right) \left( \frac{p_{jk} - c_{ij}}{\sigma_{ij}^2} \right) \right);$$



$$\frac{\partial E}{\partial \sigma_{ij}} = \sum_{k=1}^N \left( (y_k - t_k) w_i \exp\left(-\frac{1}{2} u_{ik}\right) \left( \frac{(p_{jk} - c_{ij})^2}{\sigma_{ij}^3} \right) \right);$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{k=1}^N \left( (y_k - t_k) \exp\left(-\frac{1}{2} u_{ik}\right) \right);$$

$$i = 1, \dots, S^1; \quad j = 1, \dots, N.$$

Використання методу найшвидшого спуску дозволяє виконати оновлення параметрів за формулами  $w_i^{new} = w_i^{old} - \eta_1 \frac{\partial E}{\partial w_i}$ ;  $c_{ij}^{new} = c_{ij}^{old} - \eta_2 \frac{\partial E}{\partial c_{ij}}$ ;

$\sigma_{ij}^{new} = \sigma_{ij}^{old} - \eta_2 \frac{\partial E}{\partial \sigma_{ij}}$ ,  $i = 1, \dots, S^1$ ;  $j = 1, \dots, N$ ; де  $\eta_1, \eta_2$  – коефіцієнти на-

вчання, пов'язані з параметрами ваги, центрів і дисперсії. Ітеративний процес проходить навколо даних навчання доти, поки значення параметрів не будуть надавати необхідну похибку цільової функції.

Очевидний недолік алгоритму, який спостерігається для всіх вільних параметрів – це його складність розрахунку порівняно з методом кластеризації. Якщо значення параметрів  $c_{ij}$  і  $\sigma_{ij}$  визначені на основі методів кластеризації, то залишається тільки визначити значення ваги  $w_i$  (що допоможе зменшити час навчання й уникнути проблем потрапляння в локальний мінімум).

У разі введення вектора  $\mathbf{p}$  кожен нейрон радіального базисного шару видасть значення відстані вектора входу до вагового вектора. Таким чином, радіальні базисні нейрони з ваговим вектором, який значно відрізняється від вектора входу  $\mathbf{p}$ , будуть мати виходи, близькі до нуля, їх вплив на виходи лінійних нейронів буде незначним. Навпаки, радіальний базисний нейрон з ваговим вектором, близьким до вектора входу  $\mathbf{p}$ , буде мати вихід, значення якого близько до одиниці. Це значення і буде подано на лінійний нейрон з вагою, що відповідає вихідному шару. Отже, якщо один радіальний базисний нейрон має на виході значення «1», а всі інші – значення «0» (або дуже близько до нуля), то вихід лінійного шару буде дорівнювати вазі активного вихідного нейрона. Однак це винятковий випадок, зазвичай вихід формують кілька нейронів з різними значеннями ваги.

## 9.5. Методи підбирання кількості базисних функцій

Підбирання кількості базисних функцій, кожній з яких відповідає один прихований нейрон, вважається основною проблемою, що виникає під час розв'язання задач апроксимації. Як і в разі використання багат шарових персептронів, занадто мала кількість нейронів не дозволяє зменшити до необхідного значення похибку узагальнення на множині навчання, тоді як занадто велика кількість збільшує похибку виведеного рішення на множині даних тестування. Підбирання необхідної і достатньої кількості нейронів залежить від багатьох факторів, серед яких розмірність вектора входу та об'єм навчальної вибірки. Зазвичай кількість базисних функцій  $S^1$  становить певну частку від об'єму навчальних даних  $N$ , причому фактична величина цієї частки залежить від розмірності вектора входу та від розкиду очікуваних значень  $t_i$ , які відповідають вхідним векторам  $\mathbf{p}_i$ ,  $i = 1, \dots, N$ .

**Евристичні методи.** Унаслідок неможливості апіорно визначити точну кількість прихованих нейронів застосовують адаптивні методи, які дозволяють додавати або видаляти їх у процесі навчання. Запропоновано багато евристичних методів, які реалізують такі дії. Найчастіше навчання мережі розпочинається за будь-якої взятої на початку кількості нейронів, а згодом контролюються ступінь зменшення середньоквадратичної похибки і зміна значень невідомих параметрів мережі. Якщо середня зміна значень ваги після певної кількості циклів навчання занадто мала,  $\sum_i \langle \Delta w_i \rangle < \xi$ , то додають дві базисні функції (два нейрони) із центрами, що відповідають найбільшій та найменшій похибкам адаптації, після чого навчання розширеної мережі продовжується. Одночасно контролюються абсолютні значення ваги  $w_i$  усіх окремо взятих нейронів. Якщо вони менші від встановленого на початку порогу  $\delta$ , то нейрони, які їм відповідають, видаляються з мережі. Як додавання нейронів, так й їх видалення розпочинаються після виконання певної кількості циклів навчання, що може відбуватися протягом усього процесу навчання, аж до досягнення необхідної точності відображення.

Дж. Платт запропонував інший підхід до керування кількістю прихованих нейронів, який поєднує елементи самоорганізації та навчання з учителем. Після введення кожного чергового вектора входу визначається відстань Евкліда між вектором та центром найближчої радіальної функції. Якщо ця відстань перевищує встановлений поріг  $\delta(k)$ , то створюється центр нової радіальної функції (тобто додається нейрон), після чого мережа навчається з використанням градієнтних методів (нав-

чання з учителем). Процес додавання нейронів триває аж до досягнення необхідного рівня похибки відображення. Принципово важливим для цього методу вважається підбирання значення  $\delta(k)$ , відповідно до якого приймається рішення про розширення мережі. Зазвичай  $\delta(k)$  експоненційно змінюється з часом (залежно від кількості ітерацій) від значення  $\delta_{\max}$  на початку процесу до  $\delta_{\min}$  наприкінці його. Недоліком цього підходу є неможливість зменшення кількості нейронів у процесі оброблення інформації навіть тоді, коли в результаті навчання декілька нейронів починають дублювати один одного, виконуючи ту саму функцію. Крім того, цей метод чутливий до підбирання параметрів процесу навчання, особливо значень  $\delta_{\max}$  й  $\delta_{\min}$ .

**Метод ортогоналізації найменших квадратів** [9]. Найбільш ефективним методом керування процесом визначення кількості прихованих нейронів залишається застосування спеціальної технології навчання мережі, основаної на методі ортогоналізації найменших квадратів, що використовує алгоритм ортогоналізації Грама–Шмідта. Відправна точка цього методу – зображення завдання мінімізації значення вектора похибки  $\mathbf{e}$  у вигляді лінійної адаптації вагового вектора мережі  $\mathbf{w} = [w_1 \dots w_{S1}]^T$ . Для  $N$  навчальних пар вектор очікуваних значень має вигляд  $\mathbf{t} = [t_1 \dots t_N]^T$ . У разі використання  $S^1$  базисних функцій і  $N$  навчальних пар реакція прихованих

нейронів утворює матрицю  $\mathbf{G}$  вигляду  $\mathbf{G} = \begin{bmatrix} \phi_{11} & \phi_{21} & \dots & \phi_{S^1 1} \\ \phi_{12} & \phi_{22} & \dots & \phi_{S^1 2} \\ \dots & \dots & \dots & \dots \\ \phi_{1N} & \phi_{2N} & \dots & \phi_{S^1 N} \end{bmatrix}$ , де  $\phi_{ji}$  –

реакція  $i$ -ї радіальної функції на  $j$ -й вектор навчання  $\mathbf{p}_j$ :  $\phi_{ji} = \phi(\|\mathbf{p}_j - \mathbf{c}_i\|)$ . Якщо вектор реакції  $i$ -ї радіальної функції на всі вектори навчання позначити  $\phi_i = [\phi_{i1} \ \phi_{i2} \ \dots \ \phi_{iN}]^T$ , то матрицю  $\mathbf{G}$  можна зобразити у вигляді  $\mathbf{G} = [\phi_1 \ \dots \ \phi_{S^1}]$ . За таких позначень на кожному етапі навчання буде виконуватися лінійна рівність

$$\mathbf{t} = \mathbf{G}\mathbf{w} + \mathbf{e}, \quad (9.10)$$

де  $\mathbf{w}$  – ваговий вектор;  $\mathbf{e}$  – вектор фактичної похибки навчання,  $\mathbf{e} = [e_1 \ \dots \ e_N]^T$ .

Метод ортогоналізації найменших квадратів оснований на перетворенні векторів  $\phi_i$  у множину базисних ортогональних векторів, що

дозволяє оцінити індивідуальний внесок кожного з них у загальну енергію, зображену добутком  $\mathbf{G}\mathbf{w}$ . Це дозволяє видалити вектори, вплив яких на процес апроксимації є мінімальним. У процесі навчання матриця  $\mathbf{G} \in \mathbb{R}^{N \times S^1}$  розкладається на добуток

$$\mathbf{G} = \mathbf{Q}\mathbf{A}, \quad (9.11)$$

де матриця  $\mathbf{Q}$  задовольняє умову  $\mathbf{Q}^T \mathbf{Q} = \mathbf{H}$  (при цьому  $\mathbf{H}$  – діагональна матриця з елементами  $H_{ij} = \mathbf{q}_i^T \mathbf{q}_i = \sum_{j=1}^{S^1} q_{ij}^2$ ),  $\mathbf{Q} \in \mathbb{R}^{N \times S^1}$  і складається

з ортогональних стовпців  $\mathbf{q}_i$ , а матриця  $\mathbf{A} \in \mathbb{R}^{S^1 \times S^1}$  має вигляд

$$\mathbf{A} = \begin{bmatrix} 1 & a_{12} & a_{13} & \dots & a_{1S^1} \\ 0 & 1 & a_{22} & \dots & a_{2S^1} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}.$$

Розв'язання залежності (9.10) методом найменших квадратів може бути спроектоване у простір, утворений ортогональними векторами  $\mathbf{q}_i$ . Якщо ввести нову векторну змінну  $\mathbf{b}$ , визначену як

$$\mathbf{b} = \mathbf{A}\mathbf{w}, \quad (9.12)$$

то з рівняння (9.10) одержимо

$$\mathbf{t} = \mathbf{Q}\mathbf{b} + \mathbf{e}. \quad (9.13)$$

Наближений розв'язок рівняння (9.13)  $\hat{\mathbf{b}}$  методом найменших квадратів

$$\hat{\mathbf{b}} = [\mathbf{Q}^T \mathbf{Q}]^{-1} \mathbf{Q}^T \mathbf{t} = \mathbf{H}^{-1} \mathbf{Q}^T \mathbf{t}. \quad (9.14)$$

Враховуючи діагональний характер матриці  $\mathbf{H}$ , можна одержати формулу, яка описує  $i$ -й компонент вектора  $\hat{\mathbf{b}}$ :  $\hat{b}_i = \frac{\mathbf{q}_i^T \mathbf{t}}{\mathbf{q}_i^T \mathbf{q}_i}$ . Із залежності

(9.12) випливає, що ваговий вектор  $\mathbf{w}$  можна визначити за формулою

$$\hat{\mathbf{w}} = \mathbf{A}^{-1} \hat{\mathbf{b}}. \quad (9.15)$$

З урахуванням трикутної структури матриці  $\mathbf{A}$  обчислювальна складність розв'язання рівняння (9.15) щодо вектора  $\mathbf{w}$  є невеликою.

Ортогоналізація матриці  $\mathbf{Q}$  може бути проведена різними методами, найбільш ефективним серед яких вважається алгоритм Грамма–Шмідта. Відповідно до цього методу матриця  $\mathbf{A}$  формується послідовно, стовпець за стовпцем з одночасним формуванням стовпців ортогональної матриці  $\mathbf{Q}$ . На  $k$ -му кроці формується стовпець  $\mathbf{q}_k$ , ортогональний до

всіх створених раніше  $(k - 1)$  стовпців  $\mathbf{q}_i$  ( $i = 1, \dots, k - 1$ ). Процедура повторюється для значень  $k = 2, 3, \dots, S^1$ . Математична модель цієї операції має такий вигляд:

$$\mathbf{q}_1 = \Phi; \quad \mathbf{q}_k = \Phi_k - \sum_{i=1}^{k-1} a_{ik} \mathbf{q}_i,$$

$$\text{де } a_{ik} = \frac{\mathbf{q}_i^T \Phi_k}{\mathbf{q}_i^T \mathbf{q}_i}, \quad i = 1, 2, \dots, k; \quad k = 2, 3, \dots, S^1.$$

Процедура ортогоналізації дозволяє сформувати ортогональні вектори  $\mathbf{q}_i$ ,  $i=1, 2, \dots, S^1$  і матрицю  $\mathbf{A}$ , на основі яких можна одержати наближений розв'язок  $\mathbf{b}$  (рівняння (9.14)) методом найменших квадратів, а потім на основі (9.15) визначити вектор  $\mathbf{w}$ . Багаторазово повторена процедура ортогоналізації для  $K \ll S^1$  радіальних функцій означає скорочення кількості прихованих нейронів з початкової їх кількості  $S^1$  до  $K$ . Під час фіксації початкової величини  $S^1 = N$  після багаторазового повторення ортогоналізації Грамма–Шмідта можна відібрати  $K$  найбільш значущих базисних функцій і виключити інші. Такий спосіб зменшує кількість прихованих нейронів від початкової кількості  $S^1$  до  $K$ .

Розглянемо *алгоритм відбору найбільш значимих базисних функцій*.

*Крок 1.* Для  $i = 1, \dots, S^1$  розраховують значення

$$\mathbf{q}_i(1) = \Phi; \quad b_i(1) = \frac{[\mathbf{q}_i(1)]^T \mathbf{t}}{[\mathbf{q}_i(1)]^T \mathbf{q}_i(1)}; \quad \varepsilon_i(1) = \frac{[b_i(1)]^2 [\mathbf{q}_i(1)]^T [\mathbf{q}_i(1)]}{\mathbf{t}^T \mathbf{t}}.$$

Визначають  $\varepsilon_{i1}(1) = \max \{ \varepsilon_i(1), i = 1, \dots, S^1 \}$ , кладуть вектор

$$\mathbf{q}_1 = \mathbf{q}_{i1} = \Phi_{i1}.$$

*Крок 2.* Для  $i = 1, \dots, S^1$ ,  $i \neq i_1 \neq \dots \neq i_{k-1}$  розраховують значення

$$\mathbf{q}_i(k) = \Phi_i - \sum_{j=1}^{k-1} a_{jk}^{(i)} \mathbf{q}_j,$$

$$\text{де } a_{jk}^{(i)} = \frac{\mathbf{q}_j^T \Phi_i}{\mathbf{q}_j^T \mathbf{q}_j}; \quad \varepsilon_i(k) = \frac{[b_i(k)]^2 [\mathbf{q}_i(k)]^T [\mathbf{q}_i(k)]}{\mathbf{t}^T \mathbf{t}}; \quad b_i(k) = \frac{[\mathbf{q}_i(k)]^T \mathbf{t}}{[\mathbf{q}_i(k)]^T [\mathbf{q}_i(k)]};$$

$$\varepsilon_{ik}(k) = \max \{ \varepsilon_i(k), i = 1, \dots, S^1, i \neq i_1 \neq \dots \neq i_{k-1} \}.$$

*Крок 3.* Якщо виконується умова  $1 - \sum_{j=1}^{S^1} \varepsilon_j < \rho$ , то завершують ви-

конання алгоритму, інакше переходять на крок 2. Тут  $\rho$  ( $0 < \rho < 1$ ) – це заздалегідь встановлений поріг толерантності (від  $\rho$  залежить точність відображення навчальних даних і рівень складності НМ).

У результаті виконання процедури формуються  $K$  найбільш значимі радіальні функції, розміщені у наперед визначених центрах (наприклад, через самоорганізацію). Одночасно під час реалізації алгоритму обчислюються конкретні складові вектора  $\mathbf{b}$ , на основі яких за формулою (9.15) визначають значення ваги  $\mathbf{w}$  вихідного шару мережі.

### Приклади розв'язання задач\*

**Приклад 9.1.** Розглянемо реалізацію логічної функції виключна диз'юнкція від двох змінних. Нехай задано чотири бінарні вектори входу і пов'язаний з ними вихідний вектор  $\mathbf{t}$ , які мають такий вигляд:

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}; \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \mathbf{p}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}; \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \mathbf{t} = [-1 \ 1 \ 1 \ -1]^T. \text{ Спроектувати}$$

РБФ-мережу, використовуючи лінійну РБФ вигляду  $\phi(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}_i\|$ .

*Розв'язання.* Припустимо, що мережа має чотири лінійні РБФ у прихованому шарі, а її вихід має вигляд  $y = \sum_{i=1}^4 w_i \|\mathbf{p} - \mathbf{c}_i\|$ .

Вектори центрів лінійних РБФ можна вибрати таким чином:  $\mathbf{c}_1 = \mathbf{p}_1$ ;  $\mathbf{c}_2 = \mathbf{p}_2$ ;  $\mathbf{c}_3 = \mathbf{p}_3$ ;  $\mathbf{c}_4 = \mathbf{p}_4$ . У такому випадку інтерполяційну матрицю  $\mathbf{G}$  можна обчислити як

$$\mathbf{G} = \begin{bmatrix} \|\mathbf{p}_1 - \mathbf{c}_1\| & \|\mathbf{p}_1 - \mathbf{c}_2\| & \|\mathbf{p}_1 - \mathbf{c}_3\| & \|\mathbf{p}_1 - \mathbf{c}_4\| \\ \|\mathbf{p}_2 - \mathbf{c}_1\| & \|\mathbf{p}_2 - \mathbf{c}_2\| & \|\mathbf{p}_2 - \mathbf{c}_3\| & \|\mathbf{p}_2 - \mathbf{c}_4\| \\ \|\mathbf{p}_3 - \mathbf{c}_1\| & \|\mathbf{p}_3 - \mathbf{c}_2\| & \|\mathbf{p}_3 - \mathbf{c}_3\| & \|\mathbf{p}_3 - \mathbf{c}_4\| \\ \|\mathbf{p}_4 - \mathbf{c}_1\| & \|\mathbf{p}_4 - \mathbf{c}_2\| & \|\mathbf{p}_4 - \mathbf{c}_3\| & \|\mathbf{p}_4 - \mathbf{c}_4\| \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 & 2\sqrt{2} \\ 2 & 0 & 2\sqrt{2} & 2 \\ 2 & 2\sqrt{2} & 0 & 2 \\ 2\sqrt{2} & 2 & 2 & 0 \end{bmatrix}.$$

Можна перевірити, що матриця  $\mathbf{G}$  розмірністю  $[4 \times 4]$  є несингулярною. Отже, ваговий вектор  $\mathbf{w} = [w_1 \ w_2 \ w_3 \ w_4]^T$  можна визначити за допомогою матричного рівняння  $\mathbf{G}\mathbf{w} = \mathbf{t}$ .

Це рівняння в нашому випадку має вигляд

$$\begin{bmatrix} 0 & 2 & 2 & 2\sqrt{2} \\ 2 & 0 & 2\sqrt{2} & 2 \\ 2 & 2\sqrt{2} & 0 & 2 \\ 2\sqrt{2} & 2 & 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}, \text{ звідки одержимо: } \begin{cases} w_1 = w_4 = \frac{2 + \sqrt{2}}{4}; \\ w_2 = w_3 = -\frac{2 + \sqrt{2}}{4}. \end{cases}$$

**Приклад 9.2.** Розглянемо реалізацію логічної функції виключна диз'юнкція від двох змінних. У двовимірному вхідному просторі розглядаються чотири точки (вектори):  $[0 \ 0]^T$ ,  $[0 \ 1]^T$ ,  $[1 \ 0]^T$  і  $[1 \ 1]^T$ .

---

\* Задачі взято з [12; 15].

Побудувати класифікатор, який відносить точки  $[0 \ 1]^T$  і  $[1 \ 0]^T$  до класу «1», а точки  $[0 \ 0]^T$  і  $[1 \ 1]^T$  – до класу «0». Це означає, що найближчі одна до одної точки у вхідному просторі (в сенсі відстані Хеммінга) відображаються в області, які максимально віддалені одна від одної у вихідному просторі.

*Розв'язання.* Визначимо дві функції Гаусса таким чином:  $\phi_1(\mathbf{p}) = \exp(-\|\mathbf{p} - \mathbf{c}_1\|^2)$ ;  $\phi_2(\mathbf{p}) = \exp(-\|\mathbf{p} - \mathbf{c}_2\|^2)$ , де  $\mathbf{c}_1 = [1 \ 1]^T$ ;  $\mathbf{c}_2 = [0 \ 0]^T$ . Проаналізуємо зображений в табл. 9.1 результат для різних входів.

Таблиця 9.1

Значення функцій  $\phi_i, i = 1, 2$

Вхідний образ $\mathbf{p}$	Прихована функція $\phi_1(\mathbf{p})$	Прихована функція $\phi_2(\mathbf{p})$
$[1 \ 1]^T$	1	0,1353
$[0 \ 1]^T$	0,3678	0,3678
$[0 \ 0]^T$	0,1353	1
$[1 \ 0]^T$	0,3678	0,3678

Вхідні вектори відображаються на площину  $(\phi_1 - \phi_2)$  (рис. 9.5): тепер точки  $[0 \ 1]^T$  і  $[1 \ 0]^T$  стали лінійно подільними щодо точок  $[0 \ 0]^T$  і  $[1 \ 1]^T$ . Отже, завдання можна успішно вирішити за допомогою функцій  $\phi_1(\mathbf{p})$  і  $\phi_2(\mathbf{p})$ , не підвищуючи розмірності прихованого простору, поданого функціями Гаусса.

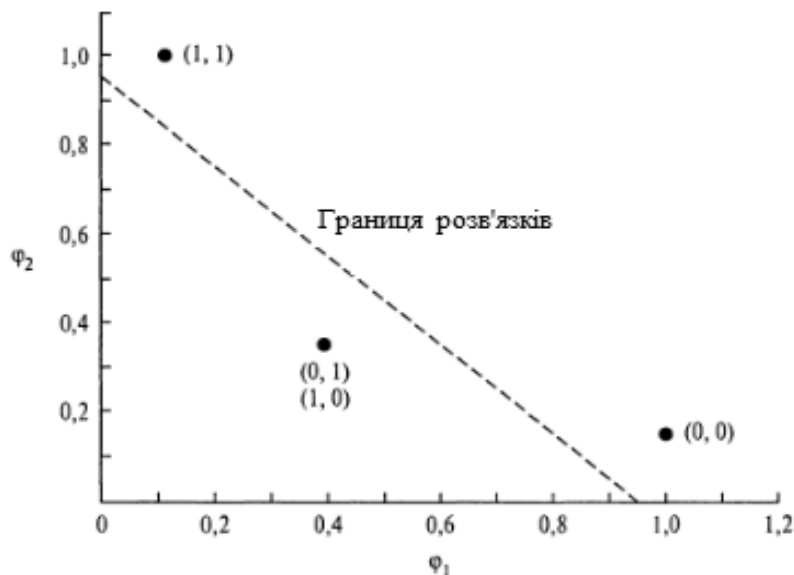


Рис. 9.5. Задані точки до прикл. 9.2

**Приклад 9.3.** Побудувати класифікатор, який відносить точки  $[0 \ 1]^T$  і  $[1 \ 0]^T$  до класу «1», а точки  $[0 \ 0]^T$  і  $[1 \ 1]^T$  – до класу «0» за допомогою ГРБФ-мережі, яка складається з двох функцій Гаусса  $G(\|\mathbf{p} - \mathbf{c}_i\|) = \exp(-\|\mathbf{p} - \mathbf{c}_i\|^2)$ ,  $i = 1, 2$  з центрами  $\mathbf{c}_1$  і  $\mathbf{c}_2$ :  $\mathbf{c}_1 = [1 \ 1]^T$ ;  $\mathbf{c}_2 = [0 \ 0]^T$ . Для описання вихідного елемента введемо такі припущення.

1. Функціонування вихідного нейрона ґрунтується на спільному використанні ваг, що обумовлено симетричним характером завдання. Це одна з форм включення апріорної інформації у конструкцію мережі. Таким чином, за наявності двох прихованих нейронів необхідно визначити одну вагу  $\mathbf{w}$ .

2. Вихідний нейрон має незалежну від даних змінну  $b$ . Важливість цього елемента пояснюється тим, що функція виключна диз'юнкція має середнє значення, відмінне від нуля.

*Розв'язання.* Співвідношення вхід–вихід для цієї мережі має такий вигляд:  $y(\mathbf{p}) = \sum_{i=1}^2 \mathbf{w} \mathbf{G}(\|\mathbf{p} - \mathbf{c}_i\|) + b$ . Відповідно до табл. 9.2 вихід мережі повинен задовольняти таку умову:  $y(\mathbf{p}_j) = t_j$ ,  $j = 1, 2, 3, 4$ , де  $\mathbf{p}_j$  – вхідний вектор;  $t_j$  – бажаний вихід. Нехай  $\phi_{ji} = \mathbf{G}(\|\mathbf{p}_j - \mathbf{c}_i\|)$ ,  $j = 1, 2, 3, 4$ ;  $i = 1, 2$ . Підставляючи значення з табл. 9.2 у цей вираз, можна отримати таку систему

$$\text{рівнянь: } \mathbf{G}\mathbf{w} = \mathbf{t}, \text{ де } \mathbf{G} = \begin{bmatrix} 1 & 0,1353 & 1 \\ 0,3678 & 0,3678 & 1 \\ 0,1353 & 1 & 1 \\ 0,3678 & 0,3678 & 1 \end{bmatrix}; \mathbf{t} = [0 \ 1 \ 0 \ 1]^T; \mathbf{w} = [w \ w \ b]^T.$$

Описана тут задача є перевизначеною в тому сенсі, що кількість точок даних перевищує кількість вільних параметрів. Це пояснює неквадратну форму матриці  $\mathbf{G}$ . Отже, обернена матриця  $\mathbf{G}^{-1}$  визначається неоднозначно. Тому використаємо рішення вигляду  $\mathbf{w} = \mathbf{G}^+ \mathbf{t} = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \mathbf{t}$ , одержимо:

$$\mathbf{G}^+ = \begin{bmatrix} 1,8292 & -1,2509 & 0,6727 & -1,2509 \\ 0,6727 & -1,2509 & 1,8292 & -1,2509 \\ -0,9202 & 1,4202 & -0,9202 & 1,4202 \end{bmatrix}; \mathbf{w} = [-2,5018 \ -2,5018 \ 2,8404]^T.$$

Таблиця 9.2

Перетворення вхід–вихід до прикл. 9.3

Точка даних $j$	Вхідний вектор $\mathbf{p}_j$	Бажаний вихід $t_j$
1	$[1 \ 1]^T$	0
2	$[0 \ 1]^T$	1
3	$[0 \ 0]^T$	0
4	$[1 \ 0]^T$	1

**Приклад 9.4.** Виконати перший крок алгоритму  $k$ -кластеризації групування заданих нижче векторів на два кластери:  $\mathbf{p}_1 = [1 \ -1 \ 1 \ -1 \ -1]^T$ ;  $\mathbf{p}_2 = [1 \ -1 \ 1 \ 1 \ -1]^T$ ;  $\mathbf{p}_3 = [-1 \ 1 \ -1 \ 1 \ 1]^T$ ;  $\mathbf{p}_4 = [-1 \ 1 \ -1 \ -1 \ -1]^T$ ;  $\mathbf{p}_5 = [-1 \ -1 \ -1 \ 1 \ -1]^T$ ;  $\mathbf{p}_6 = [-1 \ -1 \ 1 \ 1 \ -1]^T$ .

*Розв'язання.* Нехай маємо такі вектори центрів:  $\mathbf{c}_1 = \mathbf{p}_1 = [1 \ -1 \ 1 \ -1 \ -1]^T$ ;  $\mathbf{c}_2 = \mathbf{p}_2 = [1 \ -1 \ 1 \ 1 \ -1]^T$ .



Тоді маємо:  $\|\mathbf{c}_1 - \mathbf{p}_3\| = \sqrt{20}$ ,  $\|\mathbf{c}_2 - \mathbf{p}_3\| = \sqrt{16} - \mathbf{c}_2$ ;  $\|\mathbf{c}_1 - \mathbf{p}_4\| = 2\sqrt{3}$ ,  $\|\mathbf{c}_2 - \mathbf{p}_4\| = \sqrt{16} - \mathbf{c}_1$ ;  $\|\mathbf{c}_1 - \mathbf{p}_5\| = 2\sqrt{3}$ ,  $\|\mathbf{c}_2 - \mathbf{p}_5\| = 2\sqrt{2} - \mathbf{c}_2$ ;  $\|\mathbf{c}_1 - \mathbf{p}_6\| = 2\sqrt{2}$ ,  $\|\mathbf{c}_2 - \mathbf{p}_6\| = 2 - \mathbf{c}_2$ . Перший кластер містить точки  $\mathbf{p}_1, \mathbf{p}_4$  а другий – точки  $\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_6$ . Визначаємо їх центри як середні значення точок, що належать відповідному кластеру:  $\mathbf{c}_1 = [0 \ 0 \ 0 \ -1 \ -1]^T$ ;  $\mathbf{c}_2 = [0,5 \ -0,5 \ 0 \ 1 \ -0,5]^T$ .

**Приклад 9.5.** Використовуючи метод сингулярного розкладання, спроектувати ГРБФ-мережу, яка апроксимує лінійну функцію  $y = 2x + 5$  у трьох точках на інтервалі  $[-1; 1]$ , якщо кількість елементів у прихованому шарі  $S^1 = 3$ .

*Розв'язання.* Маємо такі три точки  $\mathbf{p} = [-1 \ 0 \ 1]$ . Значення  $\sigma$  визначимо за формулою  $\sigma = \frac{d_{\max}}{\sqrt{2S^1}} = \frac{1 - (-1)}{\sqrt{6}}$ . Матриця  $\mathbf{G}$  має такий вигляд:

$$\mathbf{G} = \begin{bmatrix} 1 & 0,4724 & 0,0498 \\ 0,4724 & 1 & 0,4724 \\ 0,0498 & 0,4724 & 1 \end{bmatrix}. \text{ Визначимо її сингулярне розкладання.}$$

Програмна реалізація сингулярного розкладання матриці  $\mathbf{G}$  у середовищі MatLab має такий вигляд:

```
clc; clear all; close all; S1 = 3; npts = 3;
range = [-1 1];
d1 = (range(2)-range(1))/(npts-1);
p = range(1):d1:range(2); t=2*p+5; t=t';
delta = (range(2)-range(1))/sqrt(2*S1); % sigma
sigma=1/delta; %sqrt(S1)/delta
for i1=1: length(p)
    c(i1)=p(i1); for i2=1:length(p)
        n(i1,i2) = (p(i2)-c(i1))*sigma;
        G(i1,i2) = exp(-0.5*n(i1,i2).^2);
    end;end;
f1=G'*G; f2=G*G'; [M1,M2]=eig(f1);
[N1,N2]=eig(f2)
lam=sqrt(M2); w=M1*inv(lam)*N1'*t
```

### Контрольні запитання

1. Опишіть структуру мережі з використанням ГРБФ.
2. Для чого використовують сингулярне розкладання матриці інтерполяції, у чому його сутність?
3. Які Ви знаєте алгоритми навчання мережі з використання ГРБФ? Опишіть їх.

## Задачі для самостійного розв'язання

**Задача 9.1.** Застосовуючи метод ортогоналізації найменших квадратів, який використовує алгоритм Грамма–Шмідта, спроектувати ГРБФ-мережу, яка апроксимує лінійну функцію  $y = 2x + 5$  у трьох точках на інтервалі  $[-1 \ 1]$ , якщо кількість елементів у прихованому шарі  $S^1 = 2$ .

**Задача 9.2.** Нехай задано мережу з використанням ГРБФ, яка виконує відображення  $\mathcal{R}^R \rightarrow \mathcal{R}$  й описана рівнянням вигляду  $y =$

$$= \sum_{i=1}^N w_i \exp \left( -\frac{1}{2} \sum_{j=1}^R \left[ \frac{p_j - c_{ij}}{\sigma_{ij}} \right]^2 \right); \text{ задано пари даних вхід–вихід } \{\mathbf{p}(k), t(k)\},$$

$k = 1, 2, \dots, N$ . Використовуючи алгоритм градієнтного спуску, визначити алгоритм навчання для встановлення таких параметрів: центри  $c_{ij}$ , розкид  $\sigma_{ij}$ , вагові коефіцієнти  $w_i$ .

**Задача 9.3.** Нехай задано чотири вектори:  $[-1 \ -1]^T$ ,  $[-1 \ 1]^T$ ,  $[1 \ -1]^T$  і  $[1 \ 1]^T$ . Побудувати класифікатор у вигляді РБФ-мережі, який відносить точки  $[-1 \ 1]^T$  і  $[1 \ -1]^T$  до класу 1, а точки  $[-1 \ -1]^T$  і  $[1 \ 1]^T$  – до класу 0. Центри РБФ визначаються цими векторами. Визначити матрицю інтерполяції  $\mathbf{G}$  для отриманої РБФ-мережі й обчислити для неї обернену матрицю. Обчислити лінійну вагу вихідного шару.

## Розділ 10. АСОЦІАТИВНЕ НЕКОНТРОЛЬОВАНЕ НАВЧАННЯ

### 10.1. Прості асоціативні мережі

Для НМ, які розглядалися в попередніх розділах, було характерним навчання з учителем. Розглянемо прості правила, які реалізують навчання мережі без учителя, наприклад, під час розв’язання задачі розпізнавання образів. Незважаючи на простоту, такі правила дозволяють зрозуміти принципи функціонування складних мереж.

Багато дослідників вивчали комп’ютерні асоціації, зокрема Теуво Кохонен, Джеймс Андерсон і Стівен Гроссберг. Андерсон і Кохонен незалежно один від одного (наприкінці 1960-х рр.) описали лінійну асоціативну мережу. У цей самий період Гроссберг визначив асоціативні нелінійні мережі неперервного часу.

Одну із форм асоціативного навчання – *класичний умовний рефлекс* – можна проілюструвати експериментом Павлова, коли собаку було навчено пов’язувати світло лампочки з їжею, повторюючи дослідження, в яких світло завжди вмикалося безпосередньо перед прийомом їжі.

Іншу форму асоціативного навчання – *вибірковий умовний рефлекс* – також можна проілюструвати експериментом: голодний щур, щоб отримати капсулу з живильними речовинами, мав натиснути носом певну кнопку тільки якщо горить сигнальне світло (їжа не надавалась, коли світло було вимкнене). Щур швидко вчиться натискати кнопку лише у разі появи світла. На таких експериментах ґрунтуються погляди Дональда Хебба, який спробував пояснити деякі природні явища з погляду біології. Його висновки були втілені у таке правило Хебба [12]: «*Синаптичний зв’язок між двома нейронами посилюється, якщо у процесі навчання обидва нейрони одночасно перебувають у стані збудження або гальмування*».

У розд. 4 було розглянуто роботу *контрольованого правила навчання Хебба*. Опишемо різні форми неконтрольованого навчання. Розглянемо неконтрольовані асоціативні правила навчання, мережі Кохонена і Гроссберга, які використовують асоціативне навчання.

Розглянемо найпростішу мережу [14], яка здатна реалізувати асоціації у вигляді одновхідного нейрона з чітким обмеженням (рис. 10.1), вихід якого визначається через вхід  $p$  за формулою  $y = \text{hardlim}(wp + b) = \text{hardlim}(wp - 0,5)$  за  $b = -0,5$ . Для простоти обмежимо значення вхідного  $p$  та вихідного  $y$  сигналів нейрона таким чином:

$$p = \begin{cases} 1, & \text{якщо вхідний сигнал є;} \\ 0, & \text{якщо вхідного сигналу немає;} \end{cases} \quad y = \begin{cases} 1, & \text{якщо відповідь є;} \\ 0, & \text{якщо відповіді немає.} \end{cases}$$

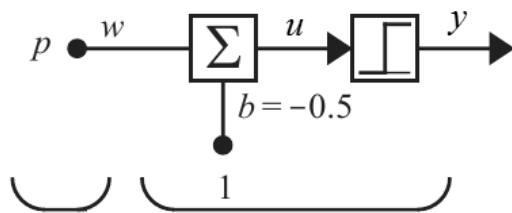


Рис. 10.1. Одновхідний асоціатор з чітким обмеженням:  $y = \text{hardlim}(wp + b)$

Наявність асоціації між вхідним сигналом  $p = 1$  і виходом мережі  $y = 1$  визначається значенням  $w$ . Мережа відповість на вхідний сигнал тільки якщо  $w > -b$  (візьмемо  $w > 0,5$ ). Проілюструємо, як асоціативні правила впливають на навчання

простих мереж, які мають два види вхідних сигналів: *безумовний* та *умовний*. Нехай один елемент вектора входу  $p^0$  зображує *безумовний* сигнал (аналог їжі, яку дали собаці в експерименті Павлова), а другий  $p$  – *умовний* (аналог світла в експерименті Павлова). Спочатку собака виділяє слину лише за наявності їжі (це природна властивість, якій не потрібно навчати). Проте, коли світло неодноразово поєднується з їжею, собака починає виділяти слину в разі появи світла навіть коли їжі немає.

Припустимо, що фіксоване значення вагового коефіцієнта  $w^0$  пов'язане зі значенням  $p^0$ , а вагове значення  $w$  – зі значенням  $p$  та уточнюється за допомогою застосування відповідного правила навчання. Мережу, здатну розпізнати банан, зображено на рис. 10.2.

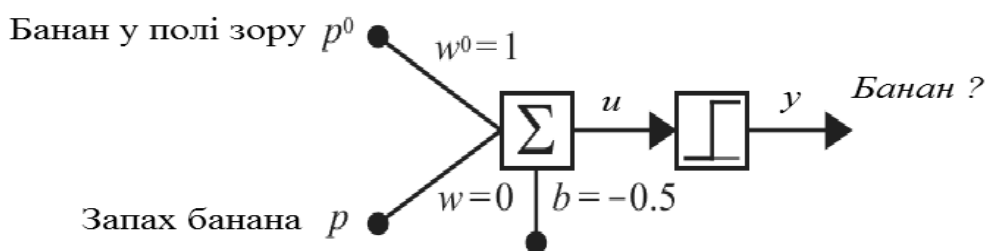


Рис. 10.2. Асоціатор (розпізнавач) банана:  $y = \text{hardlim}(w^0 p^0 + wp + b)$

Нехай на вхід мережі подаються такі сигнали: форма банана (безумовний сигнал), запах банана (умовний сигнал). Припускається, що запах може бути більш вагомим, ніж вигляд. У цьому прикладі вибір вхідного сигналу є випадковим і використовується для ілюстрації роботи правила навчання Хебба. Припустимо, що наявні такі вхідні сигнали:

$$p^0 = \begin{cases} 1, & \text{якщо виявлено форму;} \\ 0, & \text{якщо не виявлено форми;} \end{cases} \quad p = \begin{cases} 1, & \text{якщо виявлено запах;} \\ 0, & \text{якщо не виявлено запаху.} \end{cases}$$

Необхідно, щоб мережа у разі введення сигналу, який визначає форму банана (але не запах), встановлювала асоціацію, тобто надавала відповідь «Банан». Це завдання вирішується за допомогою вибору значень  $w^0 > -b$  та  $w < -b$ . Оскільки за  $b = -0,5$  значення  $w^0 = 1$  і  $w = 0$

задовольняють ці вимоги, то асоціатор банана спрощується до вигляду  $y = \text{hardlim}(p^0 - 0,5)$ . Ця мережа відповість тільки якщо банан перебуває в полі зору ( $p^0 = 1$ ), при цьому запах банана  $p$  не має значення.

Використаємо описану мережу для ілюстрації роботи декількох асоціативних правил навчання.

## 10.2. Неконтрольоване правило навчання Хебба

Для спрощення проблеми спроектуємо мережу з фіксованою множиною асоціацій. Корисною є мережа, яка може навчатися різним асоціаціям.

Відомо, що тварини і люди прагнуть пов'язати речі, які відбуваються одночасно. На думку Хебба, якщо вхідний сигнал у вигляді запаху банана супроводжується одночасно введенням його зображення, мережа повинна підсилити зв'язок між цими образами таким чином, щоб пізніше це могло активізувати її уявлення про банан у відповідь тільки на запах банана.

Неконтрольоване правило навчання Хебба збільшує значення ваги  $w_{ij}$  нейронів пропорційно добутку входу  $p_j$  та виходу  $y_i$ :

$$w_{ij}(q) = w_{ij}(q-1) + \alpha y_i(q) p_j(q), \quad (10.1)$$

або у векторній формі:  $\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{y}(q) \mathbf{p}^T(q)$ . Коефіцієнт навчання  $\alpha$  показує, як довго вхідний сигнал і відповідь мають виникати разом для того, щоб асоціація відбулася. У мережі на рис. 10.2 асоціація виникне, якщо буде виконуватися умова  $w > -b$  ( $b = -0,5$ ). У цьому випадку значення  $p = 1$  спричинить вихід  $y = 1$ , незважаючи на значення  $p^0$ . Зазначимо, що рівняння (10.1) використовує тільки сигнали, які зумовлюють модифікацію ваги. Правила, які задовольняють цю умову, називають *локальними правилами навчання* – їх розглянуто в цьому розділі.

Навчання мережі виконується на множині вхідних сигналів (множині навчання) вигляду  $\{\mathbf{p}(1), \dots, \mathbf{p}(Q)\}$ . Щоб підкреслити послідовну природу вхідних сигналів відносно часу, використовуємо символи  $\mathbf{p}(q)$  замість  $\mathbf{p}_q$ . Результатом  $q$ -ї ітерації є вихід, який відповідає входу  $\mathbf{p}(q)$ , та вагова матриця  $\mathbf{W}$ , модифікована за правилом Хебба.

**Приклад 10.1 (розпізнавання банана).** Застосуємо неконтрольоване правило Хебба до асоціатора банана. Асоціатор розпочинає функціонування зі значеннями ваги  $w^0 = 1$ ,  $w(0) = 0$ , за яких відповідь з'явиться лише в результаті реакції на зображення (але не на запах) банана. Нехай асоціатору неодноразово показують банан. Тоді навчальна послідовність має вигляд  $\{p^0(1) = 0, p(1) = 1\}$ ,  $\{p^0(2) = 1, p(2) = 1\}, \dots$ . Вага  $w^0$ , яка відповідає безумовному вхідному сигналу  $p^0$ , зберігатиме своє значення,

тоді як вага  $w$  буде змінювати своє значення на кожній ітерації після застосування неконтрольованого правила навчання Хебба з коефіцієнтом навчання  $\alpha = 1$ :  $w(q) = w(q - 1) + y(q)p(q)$ . Під час першої ітерації ( $q = 1$ ) вихід має такий вигляд:  $y(1) = \text{hardlim}(w^0 p^0(1) + w(0)p(1) - 0,5) = \text{hardlim}(1 \cdot 0 + 0 \cdot 1 - 0,5) = 0$ , тобто відповіді немає. Це означає, що один лише запах не формує відповіді. Оскільки без відповіді правило Хебба не змінює значення ваги  $w$ , то  $w(1) = w(0) + y(1)p(1) = 0 + 0 \cdot 1 = 0$ .

Під час другої ітерації виявлено форму та запах банана. На виході мережі одержуємо:

$y(2) = \text{hardlim}(w^0 p^0(2) + w(1)p(2) - 0,5) = \text{hardlim}(1 \cdot 1 + 0 \cdot 1 - 0,5) = 1$ , тобто банан.

Оскільки вхідний сигнал (запах і форма) та відповідь (банан) відбулися одночасно, то правило Хебба збільшує вагу:  $w(2) = w(1) + y(2)p(2) = 0 + 1 \cdot 1 = 1$ .

Під час третьої ітерації датчик зору нічого не виявляє, але мережа в будь-якому випадку одержує на виході банан. Це відбувається завдяки встановленій асоціації (зв'язку) між запахом банана та відповіддю мережі. Маємо:

$y(3) = \text{hardlim}(w^0 p^0(3) + w(2)p(3) - 0,5) = \text{hardlim}(1 \cdot 0 + 1 \cdot 1 - 0,5) = 1$ , тобто банан;  $w(3) = w(2) + y(3)p(3) = 1 + 1 \cdot 1 = 2$ .

Тепер мережа надаватиме відповідь «банан», якщо його виявлено одним із двох датчиків – зору або запаху.

Отже, неконтрольоване правило навчання Хебба (10.1) може формувати корисні асоціації, але воно має такі *практичні недоліки*:

1) якщо продовжити введення сигналів і модифікацію значення ваги  $w$  у розглянутому прикладі, то вага  $w$  матиме занадто велике значення (що відрізняється від біологічних систем, адже синапси не можуть рости необмежено);

2) не існує механізму зменшення (або затримки) ваги.

Якщо входи або виходи мережі Хебба мають вигляд шуму, то вага буде повільно збільшуватися, поки мережа не буде відповідати на будь-який сигнал. Отже, багаторазове введення сигналу  $p_j$  приводить до збільшення ваги  $w$  та експоненційного зростання активності, що, у свою чергу, приводить до насичення синаптичного зв'язку.

### 10.3. Правило навчання Хебба зі зменшенням (затримкою) ваги

Єдиний спосіб вдосконалити правило Хебба – додати елемент зменшення ваги:

$$\begin{aligned} \mathbf{W}(q) &= \mathbf{W}(q-1) + \alpha \mathbf{y}(q) \mathbf{p}^T(q) - \gamma \mathbf{W}(q-1) = \\ &= (1-\gamma) \mathbf{W}(q-1) + \alpha \mathbf{y}(q) \mathbf{p}^T(q), \end{aligned} \quad (10.2)$$

де коефіцієнт зменшення ваги  $\gamma$  ( $0 \leq \gamma < 1$ ) має такі особливості:

1) у разі наближення  $\gamma$  до нуля правило навчання (10.2) стає стандартним правилом Хебба;

2) у разі наближення  $\gamma$  до одиниці правило навчання (10.2) швидко забуває образи, які були введені давно, а пам'ятає тільки недавно введені образи (це утримує вагову матрицю від необмеженого зростання).

Максимальне значення ваги  $w_{ij}^{\max}$  залежить від  $\gamma$  й визначається за допомогою підстановки у рівняння (10.2) замість значень  $y_i$  і  $p_j$  значення «1» (для всіх  $q$ ) та розв'язання рівняння для стійкого стану ваги (коли попереднє й нове значення ваги збігаються):  $w_{ij}^{\max} = (1-\gamma)w_{ij}^{\max} + \alpha y_i p_j$ , звідки  $w_{ij}^{\max} = (1-\gamma)w_{ij}^{\max} + \alpha$ , тому

$$w_{ij}^{\max} = \frac{\alpha}{\gamma}. \quad (10.3)$$

Розглянемо правило Хебба із затримкою ваги на прикладі асоціатора банана.

**Приклад 10.2** (продовження прикл. 10.1). Установимо коефіцієнт зменшення ваги на рівні  $\gamma = 0,1$ , а коефіцієнт навчання  $\alpha = 1$ . Після першої ітерації, коли на вхід подається лише сигнал запаху (датчик зору не працює), вихід мережі має такий вигляд:

$$\begin{aligned} y(1) &= \text{hardlim}(w^0 p^0(1) + w(0)p(1) - 0,5) = \text{hardlim}(1 \cdot 0 + 0 \cdot 1 - 0,5) = 0; \\ w(1) &= w(0) + y(1)p(1) - 0,1w(0) = 0 + 0 \cdot 1 - 0,1(0) = 0. \end{aligned}$$

На другій ітерації датчик зору працює, а одночасне поєднання на вході запаху та форми сприяє утворенню нової асоціації:

$$\begin{aligned} y(2) &= \text{hardlim}(w^0 p^0(2) + w(1)p(2) - 0,5) = \text{hardlim}(1 \cdot 1 + 0 \cdot 1 - 0,5) = 1, \\ \text{тобто банан, } w(2) &= w(1) + y(2)p(2) - 0,1w(1) = 0 + 1 \cdot 1 - 0,1(0) = 1. \end{aligned}$$

Результат третьої ітерації такий самий:  $y(3) = \text{hardlim}(w^0 p^0(3) + w(2)p(3) - 0,5) = \text{hardlim}(1 \cdot 0 + 1 \cdot 1 - 0,5) = 1$ , тобто банан.

Мережа навчилася реагувати на запах, тому вага продовжує зростати, проте не на значення 1,0, а на 0,9:

$$w(3) = w(2) + y(3)p(3) - 0,1w(3) = 1 + 1 \cdot 1 - 0,1 \cdot 1 = 1,9.$$

Коефіцієнт зменшення ваги  $\gamma$  обмежує значення ваги таким чином, що, незважаючи на те, що асоціація зміцнюється, вага  $w$  ніколи не зросте більше, ніж до фіксованого значення  $w_{ij}^{\max}$ :  $w_{ij}^{\max} = \frac{\alpha}{\gamma} = \frac{1}{0,1} = 10$ .

Нове правило гарантує, що асоціації, яким навчилася мережа, не будуть хибними (будь-які невеликі випадкові зростання ваги швидко нейтралізуються). Відповідь правила Хебба без та зі зменшенням (затримкою) ваги для асоціатора банана зображено на рис. 10.3.

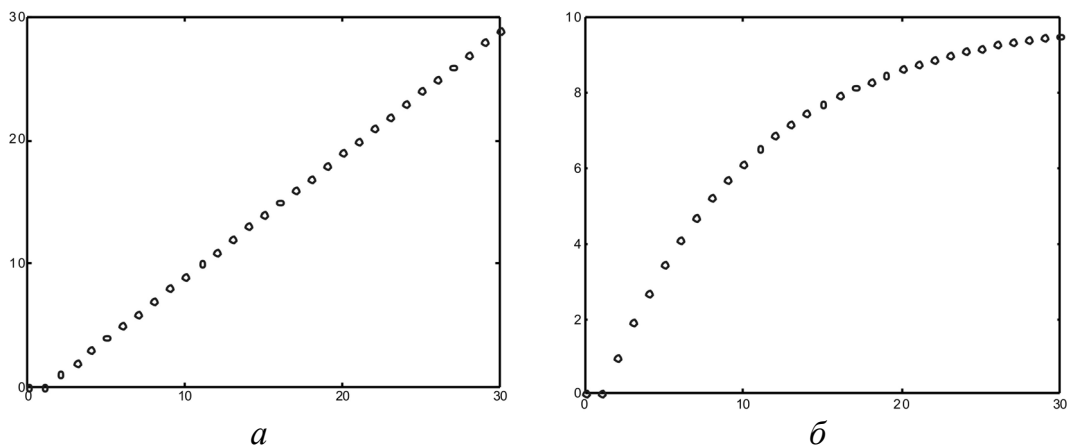


Рис. 10.3. Залежність  $w_{ij}$  від номера ітерації для правила Хебба без (а) та зі (б) зменшенням ваги

Якщо застосоване звичайне правило Хебба, вага продовжує зростати щоразу, коли нейрон активовано. З додаванням елемента зменшення ваги  $\gamma$  вага поступово (з експоненційною швидкістю) наближається до максимального значення  $w_{ij}^{\max} = 10$ . Правило Хебба зі зменшенням ваги вирішує проблему великих значень ваги, проте воно потребує перерахунку всіх вхідних сигналів, які мають асоціації (без цього асоціації загасають).

Для ілюстрації цього випадку розглянемо рівняння (10.2) за  $y_i = 0$ , одержимо  $w_{ij}(q) = (1 - \gamma)w_{ij}(q - 1)$ . Враховуючи, що  $\gamma = 0,1$ , маємо  $w_{ij}(q) = 0,9w_{ij}(q - 1)$ , тому значення  $w_{ij}$  буде зменшено на 10 % кожного разу, коли  $y_i = 0$ . Будь-яка асоціація, якій навчилася мережа, буде втрачена (забута). Розглянемо, як вирішити цю проблему.



## 10.4. Одношарова мережа Інстар. Правило навчання нейрона Інстар

Розглянемо нейрон Інстар (рис. 10.4), який має на вході вектор значень  $\mathbf{p}$  і є найпростішою мережею, яка здатна розпізнавати образи [14]. Рядки вагової матриці зображують образи, які потрібно розпізнати. Проаналізуємо здатність нейрона Інстар розпізнавати образи. Вихід нейрона має вигляд  $y = \text{hardlim}(\mathbf{W}\mathbf{p} + b) = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p} + b)$ . Нейрон Інстар буде активним кожного разу, коли  ${}_1\mathbf{w}^T \mathbf{p} \geq -b$ .

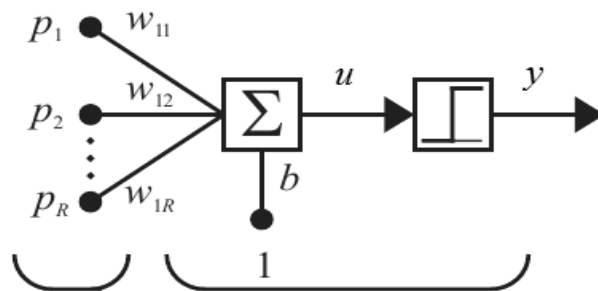


Рис. 10.4. Нейрон Інстар:  $y = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b)$

Відомо, що для двох векторів постійної довжини скалярний добуток буде найбільшим, якщо вектори мають однаковий напрямок. Можна показати це, використовуючи рівняння  ${}_1\mathbf{w}^T \mathbf{p} = \|{}_1\mathbf{w}\| \cdot \|\mathbf{p}\| \cos \theta \geq -b$ , де  $\theta$  – кут між векторами  ${}_1\mathbf{w}$  і  $\mathbf{p}$ . Зрозуміло, що скалярний добуток буде максимальним за  $\theta = 0$ . Якщо вектори  $\mathbf{p}$  і  ${}_1\mathbf{w}$  мають однакову довжину (тобто  $\|\mathbf{p}\| = \|{}_1\mathbf{w}\|$ ), то скалярний добуток буде найбільшим за  $\mathbf{p} = {}_1\mathbf{w}$ .

Враховуючи зазначені факти, нейрон Інстар (рис. 10.4) буде активним, якщо вектор  $\mathbf{p}$  розміщений близько до  ${}_1\mathbf{w}$ . Обравши відповідне значення зсуву  $b$  можна визначити, наскільки вхідний вектор має наблизитися до вектора ваги, щоб активізувати нейрон Інстар. Якщо встановити  $b = -\|{}_1\mathbf{w}\| \cdot \|\mathbf{p}\|$ , то нейрон Інстар буде активним тільки тоді, коли вектори  $\mathbf{p}$  і  ${}_1\mathbf{w}$  співнаправлені (тобто якщо  $\theta = 0$ ). У цьому разі матимемо нейрон, який розпізнає тільки образ  ${}_1\mathbf{w}$ . Якщо встановити  $b > -\|{}_1\mathbf{w}\| \cdot \|\mathbf{p}\|$ , то нейрон Інстар розпізнаватиме будь-який образ, схожий на  ${}_1\mathbf{w}$  (нейрон буде активовано для невеликого значення  $\theta$ ). Чим більшим є значення зсуву  $b$ , тим більше образів може активізувати нейрон Інстар. Зазначимо, що проведений аналіз припускає однакову довжину всіх вхідних векторів, тобто нормованість.

Знову розглянемо правило Хебба (10.2). Щоб отримати користь від зменшення ваги, обмеживши схильність до забування, необхідно, щоб коефіцієнт зменшення ваги був пропорційним значенню  $y_i(q)$ :

$$w_{ij}(q) = w_{ij}(q-1) + \alpha y_i(q) p_j(q) - \gamma y_i(q) w_{ij}(q-1). \quad (10.4)$$

Рівняння (10.4) можна спростити, вибравши  $\gamma = \alpha$ . Тоді одержимо правило Інстар вигляду  $w_{ij}(q) = w_{ij}(q-1) + \alpha y_i(q)(p_j(q) - w_{ij}(q-1))$ , яке можна записати у векторній формі:

$${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1) + \alpha y_i(q)(\mathbf{p}(q) - {}_i \mathbf{w}(q-1)). \quad (10.5)$$

Проілюструємо виконання правила Інстар (рис. 10.5). Якщо нейрон Інстар:

1) *активований*, тобто  $y_i(q) = 1$ , то рівняння (10.5) має вигляд  ${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1) + \alpha y_i(q)(\mathbf{p}(q) - {}_i \mathbf{w}(q-1)) = (1 - \alpha) {}_i \mathbf{w}(q-1) + \alpha \mathbf{p}(q)$ ;

2) *неактивований*, тобто  $y_i(q) = 0$ , то рівняння (10.5) має вигляд  ${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1)$ .

Після активації нейрона Інстар ваговий вектор  ${}_i \mathbf{w}(q)$  переміщується в напрямку вхідного вектора  $\mathbf{p}(q)$  уздовж лінії, яка сполучає попереднє значення вагового вектора  ${}_i \mathbf{w}(q-1)$  із вхідним вектором  $\mathbf{p}(q)$  (рис. 10.5). Відстань переміщення вагового вектора залежить від значення коефіцієнта навчання  $\alpha$ :

- 1) якщо  $\alpha = 0$ , то  ${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1)$  й переміщення немає;
- 2) якщо  $\alpha = 1$ , то  ${}_i \mathbf{w}(q) = \mathbf{p}(q)$  й переміщення максимальне;
- 3) якщо  $\alpha = 0,5$ , то  ${}_i \mathbf{w}(q)$  буде розміщено між  ${}_i \mathbf{w}(q-1)$  та  $\mathbf{p}(q)$ .

Відмітимо корисну *особливість правила Інстар*: якщо вхідні вектори нормалізовані, то вектор  ${}_i \mathbf{w}$  також буде нормалізованим (після навчання на конкретному прикладі вектора входу  $\mathbf{p}$ ). Отже, отримано правило, яке не тільки мінімізує схильність нейрона до забування, а й надає результат у вигляді нормалізованих вагових векторів, якщо вхідні вектори нормалізовані. Застосуємо правило Інстар до мережі, яка має такі два входи (рис. 10.6):

1) безумовний вхідний сигнал  $p^0$ , який показує, чи був плід ідентифікований датчиком зору як апельсин;

2) умовний вхідний сигнал  $\mathbf{p}$ , що містить значення трьох параметрів плоду: форми, структури, ваги.

Вихід цієї мережі має вигляд  $y = \text{hardlim}(w^0 p^0 + \mathbf{W}\mathbf{p} + b)$ .

Нехай елементи вхідного вектора  $\mathbf{p}$  набувають значення «1» або «-1». Це обмеження гарантує, що  $\mathbf{p}$  – вектор із довжиною  $\|\mathbf{p}\| = \sqrt{3}$ . Вхідні сигнали  $p^0$  і  $\mathbf{p}$  мають вигляд

$$p^0 = \begin{cases} 1, & \text{датчик зору функціонує;} \\ 0, & \text{датчик зору не функціонує;} \end{cases} \quad \mathbf{p} = \begin{bmatrix} \text{форма} \\ \text{структура} \\ \text{вага} \end{bmatrix}.$$

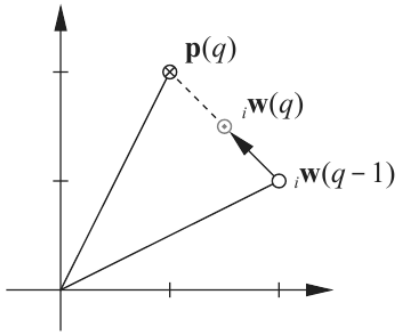


Рис. 10.5. Графічне зображення виконання правила Інстар

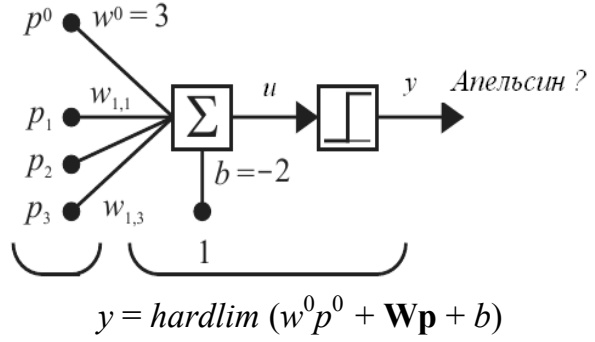


Рис. 10.6. Асоціатор апельсина

Оскільки спочатку мережа не повинна відповідати на будь-яку комбінацію значень вимірних параметрів апельсина, то початкове значення відповідної ваги має вигляд  $\mathbf{W}(0) = {}_1\mathbf{w}^T(0) = [0 \ 0 \ 0]$ . Вихід мережі має вигляд  $y = \text{hardlim}(w^0 p^0 + \mathbf{W}\mathbf{p} + b)$  і буде дорівнювати одиниці, якщо  $(w^0, p^0) + ({}_1\mathbf{w}, \mathbf{p}) \geq -b$ . Другий доданок дорівнює нулю на початку навчання мережі, тому маємо  $(w^0, p^0) \geq -b$ , звідки константа асоціації  $w^0$  між зображенням апельсина та виходом мережі відповідає умові  $w^0 > -b$  (якщо зсув  $b = -2$ ;  $w^0 = 3$ ). Вага, яка відповідає параметрам апельсина, буде модифікована за правилом Інстар з коефіцієнтом навчання  $\alpha = 1$ :  ${}_1\mathbf{w}(q) = {}_1\mathbf{w}(q-1) + y(q)(\mathbf{p}(q) - {}_1\mathbf{w}(q-1))$ .

Нехай послідовність навчання складається з таких пар: значення датчика (детектора) зображення та набору значень параметрів апельсина. Для того, щоб продемонструвати дію правила Інстар, припустимо, що система зору функціонує правильно, але на рівномірних проміжках часу. Внаслідок недоліків її конструкції вона може не спрацювати:

$$\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}; \left\{ p^0(2) = 1, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}; \dots . \text{ Оскільки вага } \mathbf{W}$$

спочатку складається лише з нулів, то після першої ітерації нейрон Інстар не відповідає на вхідні значення параметрів апельсина:

$$y(1) = \text{hardlim}(w^0 p^0(1) + \mathbf{W}\mathbf{p}(1) - 2) = \text{hardlim}\left(3 \cdot 0 + [0 \ 0 \ 0] \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 0,$$

відповіді немає. У цьому випадку, згідно з правилом Інстар, вага  ${}_1\mathbf{w}$  не змінюється:

$${}_1\mathbf{w}(1) = {}_1\mathbf{w}(0) + y(1)(\mathbf{p}(1) - {}_1\mathbf{w}(0)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0 \left( \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Проте нейрон надає відповідь, якщо апельсин був ідентифікований датчиком зору. Тому після другої ітерації одержимо

$$y(2) = \text{hardlim}(w^0 p^0(2) + \mathbf{W}\mathbf{p}(2) - 2) = \text{hardlim}\left(3 \cdot 1 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1,$$

маємо апельсин.

У результаті проведеного навчання нейрон пов'язав вектор вимірних параметрів апельсина з відповіддю. При цьому ваговий вектор  ${}_1\mathbf{w}$  стає рівним вектору параметрів апельсина:

$${}_1\mathbf{w}(2) = {}_1\mathbf{w}(1) + y(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 1 \cdot \left( \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$$

На третій ітерації нейрон надає відповідь, хоча система зорової ідентифікації знову не функціонує. Маємо:

$$y(3) = \text{hardlim}(w^0 p^0(3) + \mathbf{W}\mathbf{p}(3) - 2) = \text{hardlim}\left(3 \cdot 0 + \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1,$$

тобто апельсин.

Мережа закінчила вивчати параметри апельсина, тому вага перестала змінюватися (менше значення коефіцієнта навчання  $\alpha$  вимагає

$$\text{більшої кількості ітерацій): } {}_1\mathbf{w}(3) = {}_1\mathbf{w}(2) + y(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2)) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 1 \times$$

$$\times \left( \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}. \text{ Тепер мережа може розпізнати апельсин за зна-}$$

ченнями його вимірних параметрів (навіть коли система зорової ідентифікації не функціонує).

Отже, мережа Інстар з векторним входом і скалярним виходом може виконувати розпізнавання образів, пов'язуючи окремий вхідний векторний сигнал з відповіддю.

## 10.5. Правило Кохонена

Розглянемо правило асоціативного навчання Кохонена, пов'язане з правилом Інстар:  $w_i(q) = w_i(q-1) + \alpha(p(q) - w_i(q-1))$ ,  $\forall i \in X(q)$ . Подібно до правила Інстар, правило Кохонена дозволяє вазі нейрона навчатися на вхідних даних, а тому може бути застосоване для розв'язання задач розпізнавання образів. На відміну від правила Інстар, правило Кохонена не є пропорційним виходу нейрона  $y_i(q)$ : навчання відбувається, коли індекс нейрона  $i$  належить множині  $X(q)$ . Оскільки правило Інстар застосовують до шару нейронів, в якому ФА повертає тільки значення «0» або «1» (як, наприклад, *hardlim*), то правило Кохонена еквівалентне правилу Інстар, якщо визначити  $X(q)$  як множину всіх  $i$  таких, що  $y_i(q) = 1$ . Це корисно для навчання таких мереж, як мапа із самоорганізацією.

## 10.6. Одношарова мережа Оутстар. Правило навчання Оутстар

Мережа Оутстар (рис. 10.7) має скалярний вхід і векторний вихід. Вона може повторно викликати образи, пов'язуючи вхідний сигнал із відповіддю у вигляді вектора. Вихід цієї мережі  $y = \text{satlins}(\mathbf{W}p)$ .

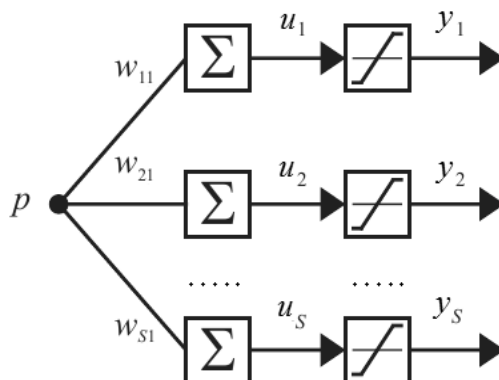


Рис. 10.7. Одношарова мережа Оутстар:  $y = \text{satlins}(\mathbf{W}p)$

Зазначимо, що симетричну ФА *satlins* обрали тому, що ця мережа використовується для повторного виклику векторів з елементами «-1» або «1». Якщо необхідно, щоб мережа пов'язала вхідний сигнал «1» з окремим вихідним вектором-прототипом образу  $y^*$ , то можна встановити  $\mathbf{W} = y^*$  (при цьому  $\mathbf{W}$  містить тільки один вектор-стовпець). Тоді, якщо  $p = 1$ , вихід  $y = \text{satlins}(\mathbf{W}p) = \text{satlins}(y^* \cdot 1) = y^*$  й образ класифіковано правильно. Стовпці вагової матриці зображують образи, які потрібно розпізнати.

Спроекуємо мережу, яка може повторно викликати відомий вектор  $y^*$ . Для цього потрібне неконтрольоване правило навчання. Щоб отримати правило навчання Інстар, коефіцієнт зменшення ваги

у правилі Хебба має бути пропорційним виходу мережі  $y_i$ . Щоб отримати правило навчання Оутстар, цей коефіцієнт має бути пропорційним входу мережі  $p_j$ :  $w_{ij}(q) = w_{ij}(q-1) + \alpha y_i(q) p_j(q) - \gamma p_j(q) w_{ij}(q-1)$ . Якщо встановити  $\gamma = \alpha$ , то отримаємо  $w_{ij}(q) = w_{ij}(q-1) + \alpha(y_i(q) - w_{ij}(q-1))p_j(q)$ . Навчання відбувається кожного разу, коли значення  $p_j \neq 0$ , при цьому  $j$ -й стовпець  $\mathbf{w}_j$  матриці  $\mathbf{W}$  переміщується в напрямку вектора виходу.

Правило Оутстар у векторній формі має такий вигляд:  $\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha(\mathbf{y}(q) - \mathbf{w}_j(q-1))p_j(q)$ , де  $\mathbf{w}_j$  –  $j$ -й стовпець матриці  $\mathbf{W}$ .

Щоб перевірити правило Оутстар, навчимо мережу, зображену на рис. 10.8. Вихід мережі обчислюється за формулою  $\mathbf{y} = \text{satlins}(\mathbf{W}^0 \mathbf{p}^0 + \mathbf{w}p)$ . Мережа має два входи:

1) безумовний вхідний сигнал  $\mathbf{p}^0$ , який зображує виміряні параметри ананаса;

2) умовний вхідний сигнал  $p$ , який вказує, чи був ананас ідентифікований датчиком зору.

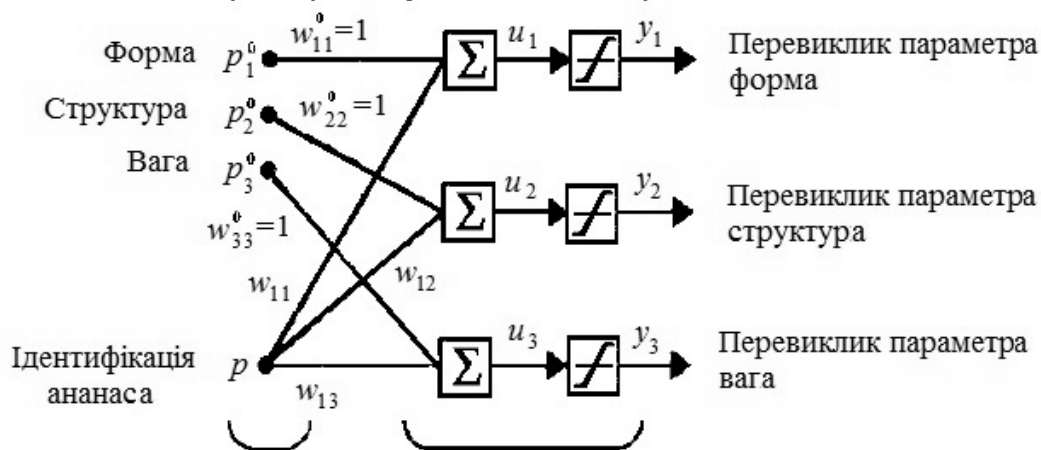


Рис. 10.8. Асоціатор ананаса:  $\mathbf{y} = \text{satlins}(\mathbf{W}^0 \mathbf{p}^0 + \mathbf{w}p)$

Вхідні сигнали  $p$  і  $\mathbf{p}^0$  мають вигляд

$$p = \begin{cases} 1, & \text{якщо ананас був ідентифікований,} \\ 0, & \text{в іншому випадку;} \end{cases} \quad \mathbf{p}^0 = \begin{bmatrix} \text{форма} \\ \text{структура} \\ \text{вага} \end{bmatrix}.$$

Вагова матриця  $\mathbf{W}$  буде модифікована за правилом Оутстар вигляду ( $\alpha = 1$ ):  $\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + (\mathbf{y}(q) - \mathbf{w}_j(q-1))p_j(q)$ .

**Приклад 10.3.** Нехай  $\alpha = 1$ , а навчальна послідовність складається з пар зорового зображення та значень вимірних параметрів ананаса, які мають вигляд  $\mathbf{p}^{\text{ананас}} = [-1 \ -1 \ 1]^T$ . Припускається, що:

1) унаслідок дефектів системи вимірювання значення параметрів

доступні тільки на скінченній кількості ітерацій:  $\left\{ \mathbf{p}^0(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, p(1) = 1 \right\};$

$$\left\{ \mathbf{p}^0(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, p(2) = 1 \right\}, \dots;$$

2) початкове значення ваги для безумовного вхідного сигналу  $\mathbf{W}^0 = \mathbf{I}$ , тобто будь-яка множина вимірюваних параметрів у вигляді вектора  $\mathbf{p}^0$  (зі значеннями « $\pm 1$ ») дублюватиметься на виході; а для умовного вхідного сигналу  $\mathbf{w}(0) = [0 \ 0 \ 0]^T$ , тобто сигнал  $p = 1$  не генерує відповіді.

На першій ітерації ананас був ідентифікований датчиком зору, але даних вимірювання немає:

$$\mathbf{y}(1) = \text{satlins}(\mathbf{W}^0 \mathbf{p}^0(1) + \mathbf{w}p(1)) = \text{satlins} \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot 1 \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{ отже, від-}$$

повіді немає.

Мережа бачить ананас, але не може вивести відповідні значення параметрів, оскільки вона цьому не навчилася (система вимірювань не працює). Вага залишається незмінною:

$$\mathbf{w}(1) = \mathbf{w}(0) + (\mathbf{y}(1) - \mathbf{w}(0))p(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \cdot 1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

На другій ітерації ананас був ідентифікований датчиком зору, а його параметри виміряні належним чином, тому на виході маємо

$$\mathbf{y}(2) = \text{satlins} \left( \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot 1 \right) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \text{ тобто параметри ананаса, при цьому}$$

вага змінюється:

$$\mathbf{w}(2) = \mathbf{w}(1) + (\mathbf{y}(2) - \mathbf{w}(1))p(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left( \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \cdot 1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}.$$

З моменту, коли зображення ананаса та значення його вимірюваних параметрів стали доступними, мережа формує асоціацію між ними. Після другої ітерації ваговий вектор збігається з параметрами ананаса, тому їх можна повторно викликати пізніше.

На третій ітерації вимірювань немає, але на виході маємо

$$\mathbf{y}(3) = \text{satlins} \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \cdot 1 \right) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \text{ тобто параметри ананаса. Тепер}$$

мережа може повторно викликати значення параметрів ананаса, коли його зображення є доступним, навіть якщо виникнуть збої у системі вимірювання. Надалі вагова матриця не змінить свого значення, якщо ананас не буде ідентифікований з іншими значеннями параметрів:

$$\mathbf{w}(3) = \mathbf{w}(2) + (\mathbf{y}(2) - \mathbf{w}(2))p(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + \left( \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \right) \cdot 1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}.$$

### **Приклади розв'язання задач\***

**Приклад 10.4.** Максимальне значення ваги за правилом Хебба з використанням затримки у вигляді коефіцієнта  $\gamma$  можна обчислити, якщо на кожному кроці  $p_j = 1$  і  $y_i = 1$ . Визначити максимальне значення ваги  $w_{ij}^{\max}$ , якщо  $p_j$  і  $y_i$  набувають одночасно значення «0» або «1».

*Розв'язання.* Правило Хебба з використанням затримки має такий вигляд:  $w_{ij}(q) = (1 - \gamma)w_{ij}(q-1) + \alpha y_i(q)p_j(q)$ . Для індексів  $(q+1)$  і  $(q+2)$  цей вираз можна переписати таким чином:  $w_{ij}(q+1) = (1 - \gamma)w_{ij}(q) + \alpha y_i(q)p_j(q)$ ;  $w_{ij}(q+2) = (1 - \gamma)w_{ij}(q+1) + \alpha y_i(q+1)p_j(q+1)$ . Підставивши вираз для  $w_{ij}(q+1)$  у  $w_{ij}(q+2)$ , одержимо формулу зміни значення  $w_{ij}$  через кожні два кроки:

$$w_{ij}(q+2) = (1 - \gamma)((1 - \gamma)w_{ij}(q) + \alpha y_i(q)p_j(q)) + \alpha y_i(q+1)p_j(q+1).$$

На цьому етапі можна замінити  $p_j$  та  $y_i$ . Враховуючи те, що треба знайти максимальну вагу, замінимо  $y_i(q)p_j(q) = 0$ , а  $y_i(q+1)p_j(q+1) = 1$ . Це означає, що на першому кроці вага зменшується, а на другому збільшується. Таким чином,  $w_{ij}(q+2)$  має вигляд  $w_{ij}(q+2) = (1 - \gamma)^2 w_{ij}(q) + \alpha$ . Якщо  $w_{ij}$  наприкінці постійно має одне й те саме значення, то його можна обчислити, замінивши  $w_{ij}(q+2)$  і  $w_{ij}(q)$  на еквівалентне значення  $w_{ij}^{\max}$ . Одержимо:

$$w_{ij}^{\max} = (1 - \gamma)^2 w_{ij}^{\max} + \alpha, \text{ звідки } w_{ij}^{\max} = \frac{\alpha}{(2\gamma - \gamma^2)}.$$

---

\* Задачі взято з посібника [14].



Використаємо систему MatLab для побудови графіка функціональної залежності максимальних значень ваги від коефіцієнтів навчання  $\alpha$  та зменшення ваги  $\gamma$ :

```
lr=0:0.025:1; dr=0.025: 0.025:1;
[LR, DR]=meshgrid (dr, lr);
MW=LR./ (DR.* (2-DR));
mesh (DR, LR, MW);
xlabel('Коефіцієнт навчання lr')
ylabel('Коефіцієнт зменшення ваги dr')
zlabel('Максимальна вага')
```

Графік показує, що  $w_{ij}^{\max}$  наближається до нескінченності зі зменшенням  $\alpha$  та збільшенням  $\gamma$  (рис. 10.9).

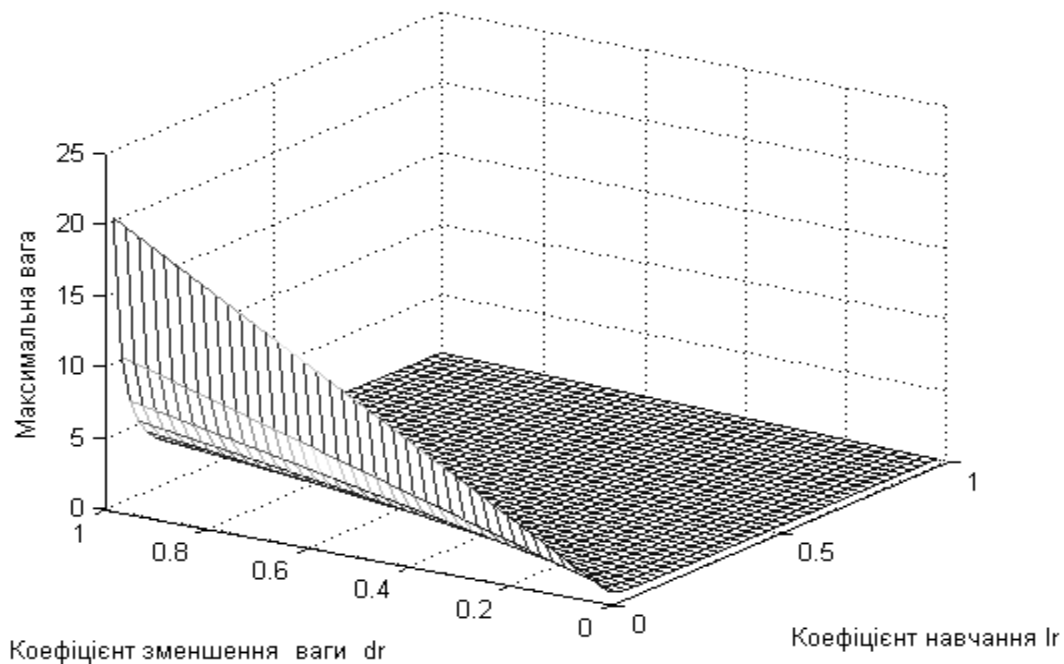


Рис. 10.9. Залежність ваги  $w_{ij}^{\max}$  від  $\alpha$  та  $\gamma$

**Приклад 10.5.** Використовуючи правило навчання Інстар з коефіцієнтом навчання  $\alpha = 0,4$ , навчити мережу, зображену на рис. 10.6, розпізнавати апельсин. Нехай навчальна послідовність  $\left\{ p^0(1)=0, \mathbf{p}(1)=\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\};$

$\left\{ p^0(2)=1, \mathbf{p}(2)=\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}; \dots$ , де вхідні сигнали  $p^0$  і  $\mathbf{p}$  мають такий ви-

гляд:  $p^0 = \begin{cases} 1, \text{ датчик зору функціонує,} \\ 0, \text{ датчик зору не функціонує;} \end{cases} \quad \mathbf{p} = \begin{bmatrix} \text{форма} \\ \text{структура} \\ \text{вага} \end{bmatrix}$ . Скільки по-

трібно виконати ітерацій, щоб навчити мережу розпізнавати апельсин?

*Розв'язання.* Задана послідовність навчання повторюється доти, поки мережа не почне відповідати на значення параметрів апельсина  $\mathbf{p} = [1 \ -1 \ -1]^T$  навіть у випадку, коли датчик зору не функціонує (тобто  $p^0 = 0$ ). Для обчислень використаємо систему MatLab. Встановимо такі початкові значення ваги:  $w^0=3$ ;  $w_{(0)} = [0 \ 0 \ 0]^T$ . Змоделюємо процес функціонування мережі на першому кроці:

```
w0=3; w=[0 0 0]; p0=0; p=[1; -1; -1];
y=hardlim(w0*p0 + w*p-2); y
y= 0
```

Отже, нейрон все ще не розпізнає апельсин і генерує на виході «0». Після виконання модифікації за правилом навчання Інстар вага не змінюється:

```
w=w+0.4*y*(p'-w); w
w= 0 0 0
```

На другій ітерації нейрон починає вивчати параметри апельсина. Одержимо:

```
p0=1; p=[1; -1;-1]; y= hardlim(w0*p0 + w*p-2); y
y= 1
w=w+0.4*y*(p'-w); w
w= 0.4000 -0.4000 -0.4000
```

Однак на третій ітерації відповіді немає, одержимо:

```
p0=0; p=[1; -1;-1]; y= hardlim(w0*p0 + w*p-2); y
y= 0
w=w+0.4*y*(p'-w); w
w= 0.4000 -0.4000 -0.4000
```

Результат четвертої ітерації:

```
y= 1
w= 0.6400 -0.6400 -0.6400
```

Результат п'ятої ітерації:

```
y= 0
w= 0.6400 -0.6400 -0.6400
```

Результат шостої ітерації:

```
y= 1
w= 0.7840 -0.7840 -0.7840
```

На сьомій ітерації мережа може самостійно розпізнати параметри апельсина:

```
p0=1; p=[1; -1;-1]; y= hardlim(w0*p0 + w*p-2); y
y= 1
w=w+0.4*y*(p'-w); w
w= 0.8704 -0.8704 -0.8704
```

Протягом шести ітерацій мережа навчається розпізнавати апельсин, поки не сформує стійкого зв'язку (асоціації) між параметрами та апельсином.

**Приклад 10.6.** Нарисувати структуру та визначити параметри мережі, яка здатна розпізнати й реагувати (надавати відповідь) на такі вектори:

$$\mathbf{p}_1 = [5 \quad -5 \quad 5]^T; \mathbf{p}_2 = [-5 \quad 5 \quad 5]^T.$$

*Розв'язання.* Мережа має надавати відповідь, коли вхідний вектор збігається з одним із заданих векторів, тому вона повинна мати три входи для розпізнавання вектора, який складається з трьох елементів, і два виходи. Таку мережу можна зобразити у вигляді поєднання в одному шарі двох нейронів Інстар (рис. 10.10).

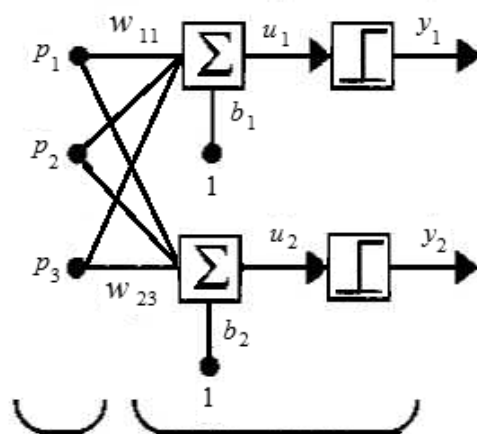


Рис. 10.10. Мережа, яка розпізнає два вектори:  $\mathbf{y} = \text{hardlink}(\mathbf{W}\mathbf{p} + \mathbf{b})$

Щоб вихід першого нейрона  $y_1$  був максимальним, коли вхідний вектор спрямований у напрямку вектора  $\mathbf{p}_1$ , визначимо вагу цього нейрона як  ${}_1\mathbf{w} = \mathbf{p}_1$ . Щоб другий нейрон мав максимальну чутливість до векторів, спрямованих у напрямку вектора  $\mathbf{p}_2$ , визначимо вагу цього нейрона як  ${}_2\mathbf{w} = \mathbf{p}_2$ . Одержимо вагову матрицю вигляду  $\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 5 & -5 & 5 \\ -5 & 5 & 5 \end{bmatrix}$ .

Довжина векторів  $\mathbf{p}_1$  і  $\mathbf{p}_2$  однакова:  $\|\mathbf{p}_1\| = \|\mathbf{p}_2\| = \sqrt{5^2 + (-5)^2 + 5^2} = \sqrt{75}$ . Щоб гарантувати реакцію мережі тільки у випадку повної відповідності між вектором входу та вектором-рядком у ваговій матриці, обидва зсуви визначимо таким чином:  $b_1 = b_2 = -\|{}_1\mathbf{w}\| \cdot \|\mathbf{p}_1\| = \|\mathbf{p}_1\|^2 = -(\sqrt{75})^2 = -75$ .

Використовуючи систему MatLab, перевіримо, як мережа реагує на вхід  $\mathbf{p}_1$ :

```
w=[5 -5 5; -5 5 5]; b=[-75; -75]; p1=[5; -5; 5];
y= hardlim(w*p1+b); y
y=    1
    0
```

Отже, мережа розпізнала вектор  $\mathbf{p}_1$ : вихід першого нейрона  $y_1 = 1$  вказує на те, що вхідним вектором був вектор  $\mathbf{p}_1$ , а вихід другого  $y_2 = 0$  на те, що вхідний вектор не був вектором  $\mathbf{p}_2$ . Протестуємо мережу на третьому векторі, який не дорівнює жодному із вхідних векторів  $\mathbf{p}_1$  і  $\mathbf{p}_2$ :

```
p3=[-5; 5; -5]; y= hardlim(w*p3+b); y
y=    0
    0
```

Отже, жодний з нейронів не розпізнав нового вектора входу  $\mathbf{p}_3$ , тому на виході одержано нулі.

**Приклад 10.7.** Нехай нейрон Інстар з вагою  $\mathbf{W} = {}_1\mathbf{w}^T = [1 \ -1 \ -1]$  та зсувом  $b = -2$  використовується для розпізнавання образу.

**I.** Визначити, наскільки близько повинен перебувати вхідний вектор  $\mathbf{p}$  із довжиною  $\sqrt{3}$  до вектора ваги  ${}_1\mathbf{w}^T$  для того, щоб вихід нейрона дорівнював одиниці.

**II.** Встановити вектор, який міститься на межі між векторами, що розпізнаються та не розпізнаються.

*Розв'язання. I.* Вихід нейрона Інстар  $y = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p} + b)$  буде дорівнювати одиниці, якщо  $({}_1\mathbf{w}, \mathbf{p}) \geq -b$ , тобто  $({}_1\mathbf{w}, \mathbf{p}) = \|{}_1\mathbf{w}\| \cdot \|\mathbf{p}\| \cos \theta \geq -b$ . Визначимо максимальний кут між векторами  ${}_1\mathbf{w}$  і  $\mathbf{p}$ :  $\sqrt{3} \cdot \sqrt{3} \cos \theta \geq 2$ , звідки  $\theta \leq \arccos\left(\frac{2}{3}\right) = 48,19^\circ$ .

**III.** Для визначення граничного вектора  $\mathbf{p}$  з довжиною  $\sqrt{3}$  потрібно, щоб він відповідав таким вимогам:  $\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + p_3^2} = \sqrt{3}$  і  ${}_1\mathbf{w}^T \mathbf{p} + b = w_1 p_1 + w_2 p_2 + w_3 p_3 + b = p_1 - p_2 - p_3 - 2 = 0$ .

Нехай  $p_1 = 0$ , тоді  $p_2^2 + p_3^2 = 3$ . Із рівняння  $p_1 - p_2 - p_3 - 2 = 0$  маємо  $p_2 + p_3 = -2$ , звідки  $(p_2 + p_3)^2 = (p_2^2 + p_3^2) + 2p_2 p_3 = 3 + 2p_2 p_3 = 4$ , тому  $p_2 p_3 = 0,5$  і  $p_2(p_2 + p_3) = (p_2^2 + p_2 p_3) = p_2^2 + 0,5 = p_2(-2) = -2p_2$ . Розв'язавши рівняння  $p_2^2 + 2p_2 + 0,5 = 0$ , знаходимо можливі значення  $p_2$ :  $p_2 = -1 \pm \sqrt{0,5}$ . Визначимо значення  $p_3$ :  $p_2 + p_3 = -1 \pm \sqrt{0,5} + p_3 = -2$ ,

звідки  $p_3 = -1 \mp \sqrt{0,5}$ . Отже, вектор  $\mathbf{p}$  має такий вигляд:  $\mathbf{p} = \begin{bmatrix} 0 \\ -1 - \sqrt{0,5} \\ -1 + \sqrt{0,5} \end{bmatrix}$ .

Перевіримо, чи може мережа розпізнати отриманий результат  $\mathbf{p}$ , подавши значення  $\mathbf{p}$  на вхід мережі:

$$y = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b) = \text{hardlim} \left( \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 + \sqrt{5} \\ -1 - \sqrt{5} \end{bmatrix} - 2 \right) = \text{hardlim}(0) = 1.$$

У разі введення вектора  $\mathbf{p}$  мережа видає результат  $y = 1$ , тому  $\mathbf{p}$  перебуває на межі активної області дії нейрона Інстар.

**Приклад 10.8.** Нехай задано мережу Інстар (рис. 10.11), послідовність навчання якої має вигляд  $\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}$ ;

$\left\{ p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}$ ; ... . Повторюючись, ці дві пари входів подаються на вхід мережі доти, поки ваговий вектор  $\mathbf{w}$  не перестане змінюватися.

**I.** Виконати чотири ітерації правила Інстар зі швидкістю навчання  $\alpha = 0,5$ , встановивши такі початкові значення ваги:  $\mathbf{W}(0) = \mathbf{w}(0) = [0 \ 0]^T$ ,  $w^0 = 3$ .

**II.** Накреслити графік виконання кожної ітерації правила Інстар.

*Розв'язання. I.* На першій ітерації нейрон Інстар не відповів, оскільки елементи вагової матриці  $\mathbf{W}$  дорівнюють нулю:

$$y(1) = \text{hardlim}(w^0 p^0(1) + \mathbf{w}^T(0) \mathbf{p}(1) - 2) = \text{hardlim} \left( 3 \cdot 0 + \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2 \right) = 0.$$

Виконання правила Інстар не змінює значення ваги:

$$\mathbf{w}(1) = \mathbf{w}(0) + 0,5 y(1) (\mathbf{p}(1) - \mathbf{w}(0)) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0,5 \cdot 0 \cdot \left( \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Оскільки на другій ітерації з'явився безумовний сигнал  $p^0(2) = 1$ , то нейрон Інстар надав відповідь:

$$y(2) = \text{hardlim}(w^0 p^0(2) + \mathbf{w}^T(1) \mathbf{p}(2) - 2) = \text{hardlim} \left( 3 \cdot 1 + \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2 \right) = 1.$$

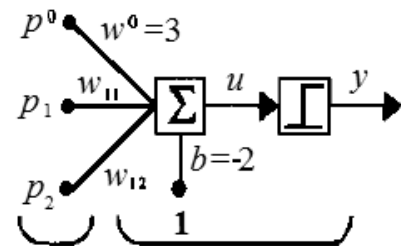


Рис. 10.11. Мережа Інстар:  
 $y = \text{hardlim}(w^0 p^0 + \mathbf{w} \mathbf{p} + b)$

Після виконання правила Інстар вага змінилася:

$${}_1\mathbf{w}(2) = {}_1\mathbf{w}(1) + 0,5y(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1)) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0,5 \cdot 1 \cdot \left( \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix}.$$

На третій ітерації маємо: безумовного сигналу немає,  $p^0(3) = 0$ , а значення ваги достатньо близько розміщене до входу, але нейрон Інстар не надає відповіді:

$$\begin{aligned} y(3) &= \text{hardlim}(w^0 p^0(3) + {}_1\mathbf{w}^T(2)\mathbf{p}(3) - 2) = \\ &= \text{hardlim}\left(3 \cdot 0 + \begin{bmatrix} -0,5 & 0,5 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 0. \end{aligned}$$

У цьому випадку вага не змінюється:

$$\begin{aligned} {}_1\mathbf{w}(3) &= {}_1\mathbf{w}(2) + 0,5y(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2)) = \\ &= \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} + 0,5 \cdot 0 \cdot \left( \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} \right) = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix}. \end{aligned}$$

Оскільки на четвертій ітерації безумовний сигнал  $p^0$  знову з'явився, то нейрон Інстар надає відповідь:

$$\begin{aligned} y(4) &= \text{hardlim}(w^0 p^0(4) + {}_1\mathbf{w}^T(3)\mathbf{p}(4) - 2) = \\ &= \text{hardlim}\left(3 \cdot 1 + \begin{bmatrix} -0,5 & 0,5 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 1. \end{aligned}$$

Після виконання правила Інстар вага змінилася:

$$\begin{aligned} {}_1\mathbf{w}(4) &= {}_1\mathbf{w}(3) + 0,5y(4)(\mathbf{p}(4) - {}_1\mathbf{w}(3)) = \\ &= \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} + 0,5 \cdot 1 \cdot \left( \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} \right) = \begin{bmatrix} -0,75 \\ 0,75 \end{bmatrix}. \end{aligned}$$

На цьому завершується четверта ітерація. Якщо продовжити цей процес, то значення вектора  ${}_1\mathbf{w}$  збігатиметься зі значенням вхідного вектора  $\mathbf{p}$ .

**II.** Зазначимо, що значення ваги змінюється тільки на другій і четвертій ітераціях (коли нейрон Інстар виводить «1»). У цьому випадку правило навчання нейрона Інстар можна записати у вигляді

$$\begin{aligned} {}_1\mathbf{w}(q) &= {}_1\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_1\mathbf{w}(q-1)) = \\ &= (1 - \alpha){}_1\mathbf{w}(q-1) + \alpha\mathbf{p}(q). \end{aligned}$$

Якщо нейрон Інстар стає активним, то ваговий вектор рухається назустріч вхідному вектору вздовж лінії, що з'єднує попередній ваговий вектор із вхідним вектором. Процес зміни значення вагового вектора зображено на рис. 10.12. Оскільки  $\alpha = 0,5$ , то кожного разу, коли нейрон Інстар стає

активним, ваговий вектор проходить половину шляху в напрямку до вхідного вектора:  ${}_1w(q) = 0,5{}_1w(q-1) + 0,5p(q)$ .

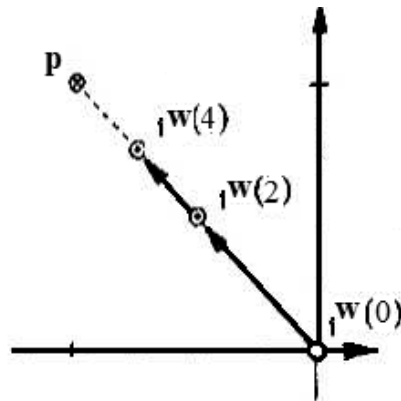


Рис. 10.12. Графік виконання правила навчання нейроном Інстар

### Контрольні запитання

1. Наведіть приклади простих асоціативних мереж.
2. Опишіть неконтрольоване правило Хебба та правило Хебба із зменшенням значення ваги.
3. Опишіть просту мережу Інстар та її правило навчання.
4. Опишіть просту мережу Оутстар та її правило навчання.

### Задачі для самостійного розв'язання

**Задача 10.1.** Мережу, яка навчається за правилом Хебба зі зменшенням ваги, зображено на рис. 10.13. Нехай коефіцієнти навчання та зменшення ваги  $\alpha = 0,3$  й  $\gamma = 0,1$  відповідно.

**I.** Визначити, скільки разів необхідно послідовно подавати на вхід мережі пари навчання для того, щоб нейрон почав надавати правильну відповідь, якщо встановити такі початкові значення:  $w(0) = 0$ ;  $w^0 = 1$  і  $b = -0,8$ . Пара навчання має вигляд  $\{p^0 = 1, p = 1\}$ , а тестова пара  $\{p^0 = 0, p = 1\}$ . Побудувати графік залежності ваги  $w$  від кількості ітерацій.

**II.** Нехай початкове значення ваги  $w(0) = 1$ . Визначити, скільки разів необхідно послідовно подавати на вхід мережі пари навчання для того, щоб нейрон не припинив надавати відповідь. Пара навчання має вигляд  $\{p^0 = 0, p = 0\}$ , а тестова пара  $\{p^0 = 0, p = 1\}$ . Побудувати графік залежності ваги  $w$  від кількості ітерацій.

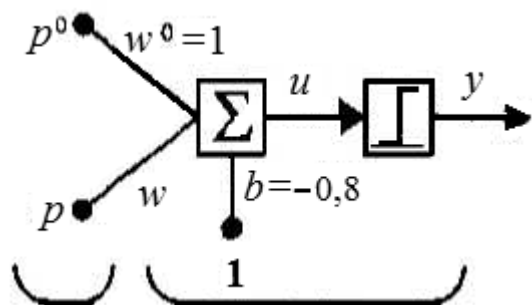


Рис. 10.13. Асоціативна мережа:

$$y = \text{hardlim}(w^0 p^0 + wp + b)$$

**III.** Використати формулу (10.3) визначення стійкого значення ваги  $w^{\max}$ . Перевірити, чи збігається отримана відповідь із графіком першої частини вправи (п. I).

**Задача 10.2.** Правило вигляду  $\Delta w_{ij} = -\alpha y_i(p_j + w_{ij}^{\text{old}})$  називають простим правилом навчання нейрона Інстар, але його поведінка не є однозначною. Визначити умови, за яких  $\Delta w_{ij} = 0$ , та значення, які має вага, якщо  $\Delta w_{ij} \neq 0$ .

**Задача 10.3.** Розглянемо мережу Інстар для розпізнавання образів (рис. 10.11). Послідовність навчання для заданої мережі  $\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}; \left\{ p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}; \dots$ . Задані пари входів будуть повторюватися доти, поки ваговий вектор  $\mathbf{w}$  не перестане змінюватися.

Виконати перші вісім ітерацій правила Інстар, якщо коефіцієнт навчання  $\alpha = 0,25$ , початкове значення ваги  $\mathbf{w}(0) = [1 \ 0]^T$ .

**Задача 10.4.** Нехай задана мережа Інстар (див. рис. 10.11) використовується для розпізнавання векторів.

**I.** Використовуючи правило навчання Інстар, навчити задану мережу розпізнавати таку послідовність навчання:

$$\begin{aligned} & \left\{ p^0(1) = 1, \mathbf{p}(1) = \begin{bmatrix} 0,174 \\ 0,985 \end{bmatrix} \right\}; \left\{ p^0(2) = 0, \mathbf{p}(2) = \begin{bmatrix} -0,174 \\ 0,985 \end{bmatrix} \right\}; \\ & \left\{ p^0(3) = 1, \mathbf{p}(3) = \begin{bmatrix} 0,174 \\ 0,985 \end{bmatrix} \right\}; \left\{ p^0(4) = 0, \mathbf{p}(4) = \begin{bmatrix} -0,174 \\ 0,985 \end{bmatrix} \right\}; \\ & \left\{ p^0(5) = 1, \mathbf{p}(5) = \begin{bmatrix} 0,174 \\ 0,985 \end{bmatrix} \right\}; \left\{ p^0(6) = 0, \mathbf{p}(6) = \begin{bmatrix} -0,174 \\ 0,985 \end{bmatrix} \right\}. \end{aligned}$$

Правило Інстар використовувати тільки до кожного другого входу, початкові значення вагової матриці дорівнюють нулю, а швидкість навчання  $\alpha = 0,6$ . Інші значення ваги та зсуву залишаються незмінними ( $w^0 = 3$ ;  $b = -2$ ). Для виконання обчислень можна використати систему MatLab.

**II.** Визначити кінцеве значення вагового вектора  $\mathbf{w}$ . Як цей кінцевий результат співвідноситься з векторами послідовності навчання?

**III.** Якого значення ваги слід очікувати після навчання, якщо мережа продовжить процес навчання на заданій послідовності?

**Задача 10.5.** Накреслити діаграму мережі Інстар, яка здатна розпізнавати три різні чотирьохелементні вектори зі значеннями «1» або «-1», якщо вона одержує збуджуючий сигнал зі значенням «1»:  $p^0(1) = 1, p^0(2) = 1, \dots$



**I. Визначити:**

- 1) скільки входів і виходів має мережа;
- 2) яку ФА можна використати.

**II. Визначити значення вагової матриці, за допомогою якої мережа**

може розпізнати кожний із таких векторів:  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$ ;

$\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$ , та значення зсуву, яке можна використати. Протестувати

мережу на одному із заданих векторів та векторі  $\mathbf{p} = [-1 \ -1 \ 1 \ 1]^T$ .

**Задача 10.6.** Мережу було встановлено на підйомник (рис. 10.14), який використовують три працівники. Підйомник має кнопки, марковані відповідно від «1» до «4» для чотириповерхового будинку з підвальним поверхом. Коли працівник входить до підйомника на підвальному поверсі, система сканує сітківку його ока та визначає, яка людина увійшла. Далі мережа обирає номер поверху, на який цей працівник найчастіше піднімається (має доступ). Перший вхід мережі  $\mathbf{p}^0$

вказує номер поверху, якщо було натиснуто кнопку:  $\mathbf{p}_1^0 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$  – перший поверх;  $\mathbf{p}_2^0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$  – другий поверх;  $\mathbf{p}_3^0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$  – третій поверх;

$\mathbf{p}_4^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$  – четвертий поверх;  $\mathbf{p}_0^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$  – жодна з кнопок не натиснута.

Вхідному вектору  $\mathbf{p}^0$  відповідають вагова матриця та вектор зсуву  $\mathbf{W}^0 = \mathbf{I}$ ;  $\mathbf{b} = \begin{bmatrix} -0,5 \\ -0,5 \end{bmatrix}$ . Другий вхід зображує трьох працівників компанії:

$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$  – 1-й працівник;  $\mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$  – 2-й працівник;  $\mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$  – 3-й праців-

ник. Мережа вчиться визначати поверхи, яким надають перевагу пра-

цівники, змінюючи вагову матрицю за правилом Інстар з коефіцієнтом навчання  $\alpha = 0,6$ . Початкове значення вагової матриці  $\mathbf{W}(0) =$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

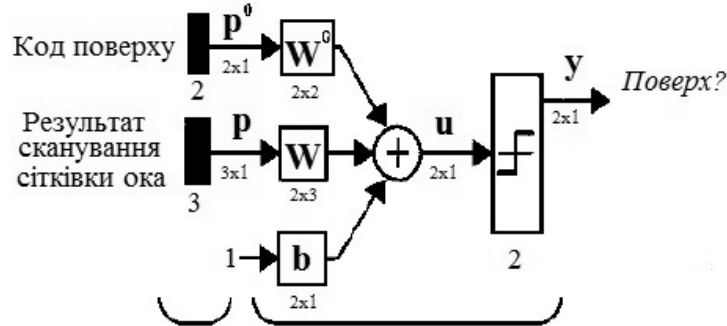


Рис. 10.14. Мережа для підйомника:  $y = \text{hardlims}(\mathbf{W}^0 \mathbf{p}^0 + \mathbf{W} \mathbf{p} + b)$

**I.** Використовуючи систему MatLab, змодельовати мережу для такої послідовності навчання:  $\{\mathbf{p}^0 = \mathbf{p}_4^0, \mathbf{p} = \mathbf{p}_1\}$ ;  $\{\mathbf{p}^0 = \mathbf{p}_3^0, \mathbf{p} = \mathbf{p}_2\}$ ;  $\{\mathbf{p}^0 = \mathbf{p}_1^0, \mathbf{p} = \mathbf{p}_3\}$ ;  $\{\mathbf{p}^0 = \mathbf{p}_3^0, \mathbf{p} = \mathbf{p}_2\}$ ;  $\{\mathbf{p}^0 = \mathbf{p}_2^0, \mathbf{p} = \mathbf{p}_3\}$ ;  $\{\mathbf{p}^0 = \mathbf{p}_4^0, \mathbf{p} = \mathbf{p}_1\}$  (1-й працівник натиснув кнопку «4», 2-й – кнопку «3», 3-й – кнопку «1», 2-й – кнопку «3», 3-й – кнопку «2», 1-й – кнопку «4»). Визначити кінцеве значення вагової матриці.

**II.** Продовжити моделювання мережі для таких випадків: 1-й працівник не натиснув кнопки, 2-й – не натиснув кнопки, 3-й – не натиснув кнопки.

**III.** Визначити:

- 1) який поверх вибрала мережа для кожного з працівників;
- 2) яким буде вигляд вагової матриці після багаторазового натискання працівниками таких кнопок: 1-й натиснув кнопку «3», 2-й – кнопку «2», 3-й – кнопку «4».

## Розділ 11. КОНКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ

### 11.1. Нейронна мережа Хеммінга

Розглянемо мережі, які ґрунтуються на *конкурентному навчанні* [12]: окремі нейрони вихідного шару такої мережі змагаються за право активації, у результаті чого активним стає один нейрон мережі. Вихідний нейрон, який переміг у цьому змаганні, називають *переможцем*. Один зі способів організації такої конкуренції між нейронами полягає у використанні латеральних (від'ємних зворотних) зв'язків між ними. Цю ідею вперше запропонував Френк Розенблатт, який у 1959 р. розробив простий бінарний класифікатор у вигляді одношарового персептрона, що навчався без учителя. Наприкінці 1960-х – на початку 1970-х рр. Стефан Гроссберг розробив декілька конкурентних мереж, які використовували латеральне гальмування.

У 1973 р. Крістоф фон дер Мальсбург ввів *самоорганізуюче правило навчання*, яке дозволяло мережі класифікувати вхідні дані так, щоб сусідні нейрони відповідали схожим даним (його мережі моделювали топологічні структури даних). Ст. Гроссберг розширив теорію фон дер Мальсбурга, запропонувавши правило навчання Інстар, яке раніше було введено в 1965 р. Нільсом Нільссоном. Гроссберг довів, що правило Інстар усуває необхідність нормалізації вагового вектора, оскільки вагові вектори, які навчаються розпізнавати нормалізовані вектори входу, також будуть нормалізованими.

Протягом 1970-х рр., використовуючи результати досліджень фон дер Мальсбурга та Ст. Гроссберга щодо проблеми ефективного математичного описання конкурентних мереж, Теуво Кохонен розробив спрощену версію правила Інстар і запропонував ефективний спосіб використання топологічних структур даних у конкурентній мережі.

Розглянемо конкурентні мережі Кохонена, які легше піддаються математичній обробці порівняно з мережами Гроссберга (оскільки мережі Кохонена пов'язані з мережею Хеммінга, то спочатку розглянемо останню). Покажемо, як конкуренція може поєднуватися з правилом асоціативного навчання при створенні потужних самоорганізованих мереж, які навчаються без вчителя.

*Мережа Хеммінга* (рис. 11.1) – один із найпростіших прикладів конкурентної мережі, в якій нейрони вихідного другого шару конкурують між собою з метою визначення нейрона-переможця, що показує, якому вектору-прототипу найкраще відповідає вхідний вектор. Конкуренція реалізована за допомогою *латерального гальмування* – набору від'ємних з'єднань між нейронами вихідного шару.

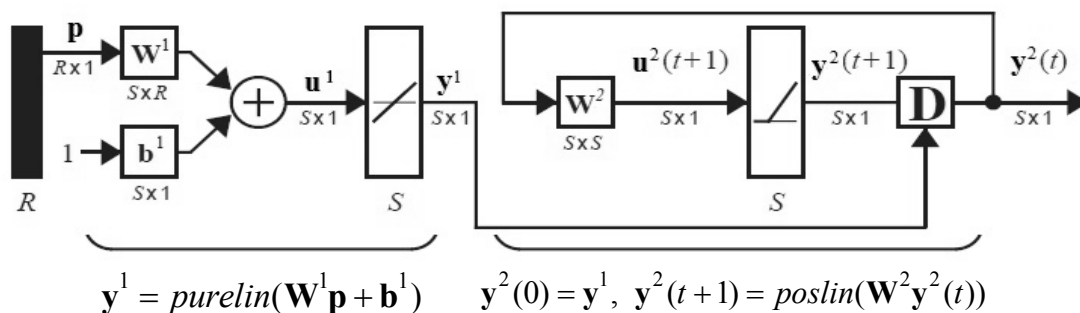


Рис. 11.1. Мережа Хеммінга

Мережа Хеммінга складається з двох шарів (кількість нейронів у першому та другому шарі однакова):

– *перший шар* (Інстар) – це шар із прямим зв’язком, який визначає кореляцію (скалярний добуток) між вхідним вектором і векторами-прототипами образів, зафіксованими у мережі;

– *другий шар* (конкурентний) – це рекурентний шар, який реалізує конкуренцію нейронів для визначення вектора-прототипа, найближчого відносно відстані Хеммінга до вектора входу (відстань Хеммінга між двома двійковими векторами дорівнює кількості їхніх елементів, які відрізняються один від одного).

Оскільки один нейрон Інстар може розпізнати тільки один образ, то для класифікації декількох образів потрібно мати багато нейронів типу Інстар.

Нехай необхідно спроектувати мережу, здатну розпізнавати множину векторів-прототипів  $\{\mathbf{p}_1, \dots, \mathbf{p}_Q\}$ , де  $\mathbf{p}_i \in \mathbb{R}^R$ . Тоді вагова матриця

$$\mathbf{W}^1 \text{ і вектор зсуву } \mathbf{b}^1 \text{ першого шару мають вигляд } \mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \dots \\ \mathbf{p}_Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \\ \dots \\ \mathbf{p}_Q^T \end{bmatrix};$$

$\mathbf{b}^1 = \begin{bmatrix} R \\ \dots \\ R \end{bmatrix}$ , де кожний рядок матриці  $\mathbf{W}^1$  зображає вектор-прототип образу, який необхідно розпізнати, а вектор  $\mathbf{b}^1$  – множину чисел, кожне з яких дорівнює кількості елементів відповідного вхідного вектора (тобто  $R$ ). Кількість нейронів  $S$  дорівнює кількості векторів-прототипів  $Q$ , які необхідно розпізнати (тобто  $S = Q$ ). Таким чином, вихід першого шару

$$\text{має такий вигляд: } \mathbf{y}^1 = \mathbf{W}^1 \mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \mathbf{p} + R \\ \dots \\ \mathbf{p}_Q^T \mathbf{p} + R \end{bmatrix} \text{ і показує, наскільки близько}$$

кожний з векторів-прототипів образів розміщений до вектора входу.

Вихід першого шару  $\mathbf{y}^1$  використовується для ініціалізації другого шару:  $\mathbf{y}^2(0) = \mathbf{y}^1$ . Далі нейрони змагаються один з одним і визначають переможця, після чого тільки один нейрон на виході має ненульове значення. Нейрон-переможець показує, якому класу належить введений до мережі вектор входу (кожний вектор-прототип відомого образу являє собою один клас). Потім вихід другого шару оновлює своє значення за формулою

$$\mathbf{y}^2(t+1) = \text{poslin}(\mathbf{W}^2 \mathbf{y}^2(t)), \quad (11.1)$$

де лінійна функція активації *poslin* є додатною для додатних значень і набуває значення «0» для від'ємних значень.

*Латеральне гальмування.* Вагова матриця другого шару  $\mathbf{W}^2$  складається з таких елементів:  $w_{ij}^2 = \begin{cases} 1, & \text{якщо } i = j; \\ -\varepsilon, & \text{якщо } i \neq j, \end{cases} \quad 0 < \varepsilon < \frac{1}{S-1}$ , і виконує латеральне гальмування, за якого вихід кожного нейрона має гальмівний вплив на всі інші нейрони. Проілюструємо цей ефект, переписавши вираз (11.1) для  $i$ -го нейрона у вигляді  $y_i^2(t+1) = \text{poslin}(y_i^2(t) - \varepsilon \sum_{j \neq i} y_j^2(t))$ , звідки можна зробити висновок, що нейрон з найбільшою початковою умовою виграє змагання.

Під час кожної ітерації кожний вихід нейрона буде зменшуватися прямо пропорційно до суми виходів інших нейронів (з мінімальним значенням виходу «0»). При цьому вихід нейрона з найбільшим початковим значенням буде зменшуватися повільніше за виходи інших нейронів, і поступово додатне значення буде мати тільки один цей нейрон. На цьому етапі мережа досягає стабільного стану. Номер нейрона другого шару з додатним виходом збігається з номером вектора-прототипа образу, який найкраще відповідає входу. Описаний процес називають змаганням «Переможець одержує все».

**Приклад 11.1.** Нехай задано вектори-прототипи образів яблука та апельсина, які мають вигляд тривимірного вектора з такими елементами:

$\mathbf{p} = \begin{bmatrix} \text{форма} \\ \text{структура} \\ \text{вага} \end{bmatrix}$ . Вектори  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$  і  $\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$  зображують прототипи

апельсина і яблука відповідно. Тоді перший шар мережі Хеммінга визначає кореляцію між кожним із заданих векторів-прототипів і вхідним вектором:  $\mathbf{y}^1 = \mathbf{W}^1 \mathbf{p} + \mathbf{b}^1$ . Для цього рядки вагової матриці  $\mathbf{W}^1$  повинні дорівнювати

заданим векторам-прототипам, тобто  $\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$ .

Перший шар із прямим зв'язком використовує лінійну ФА. При цьому кожний елемент вектора зсуву  $\mathbf{b}^1$  дорівнює кількості елементів вхідного вектора  $R = 3$ , тобто  $\mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ . На виході першого шару одержимо зна-

чення  $\mathbf{y}^1 = \mathbf{W}^1 \mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} \mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \mathbf{p} + 3 \\ \mathbf{p}_2^T \mathbf{p} + 3 \end{bmatrix}$ . Зазначимо, що для двох

векторів однакової довжини їхній скалярний добуток буде мати найбільше значення, якщо ці вектори є співнапрямленими, та найменше, якщо вони мають протилежні напрямки. Додавання числа  $R$  до скалярного добутку гарантує, що значення на виході першого шару ніколи не будуть від'ємними (це необхідно для правильної роботи другого рекурентного шару).

Описану мережу називають мережею Хеммінга, оскільки нейрон першого шару з найбільшим значенням виходу буде відповідати вектору-прототипу, який є найближчим відносно відстані Хеммінга до вектора входу. Можна показати, що вихід першого шару дорівнює  $2R$  мінус подвійна відстань Хеммінга від векторів-прототипів до вектора входу.

Розглянемо процес функціонування другого шару. Маємо два класи: яблуко та апельсин. Рівняння, які описують процес змагання, мають вигляд  $\mathbf{y}^2(0) = \mathbf{y}^1$ ;  $\mathbf{y}^2(t+1) = \text{poslin}(\mathbf{W}^2 \mathbf{y}^2(t))$ . Вагова матриця  $\mathbf{W}^2$  має вигляд

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}, \quad (11.2)$$

де  $\varepsilon < 1/(S-1)$ ,  $S$  – кількість нейронів у другому шарі.

Один крок функціонування рекурентного шару має такий вигляд:

$$\mathbf{y}^2(t+1) = \text{poslin} \left( \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \cdot \mathbf{y}^2(t) \right).$$

На першій ітерації значення кожного елемента вихідного вектора зменшується. На виході всіх нейронів рекурентного шару, за винятком того, який мав найбільше початкове значення, одержимо «0». Нейрон із ненульовим виходом відповідає вектору-прототипу та розміщений найближче до вхідного вектора відносно відстані Хеммінга. Перевіримо ефективність функціонування спроектованої мережі Хеммінга з прикладу 11.1. Нехай на вхід мережі подається вектор (довгий апельсин):

$$\mathbf{p} = [-1 \ -1 \ -1]^T. \text{ Тоді вихід першого шару } \mathbf{y}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$i$  є початковою умовою для рекурентного шару. Нехай вагова матриця рекурентного шару  $\mathbf{W}^2$  має вигляд (11.2), де  $\varepsilon = 0,5$ . Тоді перша ітерація функціонування рекурентного шару

$$\mathbf{y}^2(1) = \text{poslin}(\mathbf{W}^2 \mathbf{y}^2(0)) = \text{poslin}\left(\begin{bmatrix} 1 & -0,5 \\ -0,5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix}\right) = \text{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}.$$

Запишемо другу ітерацію:

$$\mathbf{y}^2(2) = \text{poslin}(\mathbf{W}^2 \mathbf{y}^2(1)) = \text{poslin}\left(\begin{bmatrix} 1 & -0,5 \\ -0,5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \text{poslin}\left(\begin{bmatrix} 3 \\ 1,5 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}.$$

Як тільки вихід першого нейрона став відмінним від нуля, було обрано вектор-прототип апельсина, – це правильна відповідь, оскільки відстань Хеммінга від вектора входу до вектора-прототипа апельсина дорівнює одиниці, а до вектора-прототипа яблука – двом.

Отже, мережу Хеммінга можна застосовувати для розпізнавання образів. При цьому необхідно, щоб вектори-прототипи образів були відомі на початку процесу розпізнавання та включені в мережу у вигляді рядків вагової матриці.

Розглянемо мережі, які схожі за структурою та функціонуванням на мережу Хеммінга, але, на відміну від неї, використовують правила асоціативного навчання для класифікації образів: конкурентну мережу, шар  $i$  мапу Кохонена та мережу з дискретизацією навчальних векторів.

## 11.2. Конкурентна одношарова мережа

Функція активації конкурентного шару вигляду  $y = \text{compet}(u)$  визначає номер нейрона з найбільшим значенням входу та присвоює виходу цього нейрона значення «1», а виходам всіх інших нейронів – значення «0»:

$$\forall i \ y_i = \begin{cases} 1, & \text{якщо } i = i^*; \\ 0, & \text{якщо } i \neq i^*, \end{cases} \quad u_{i^*} \geq u_i.$$

Розглянемо мережу з одним конкурентним шаром (рис. 11.2), в якій вектори-прототипи образів зберігаються у вигляді рядків вагової матриці  $\mathbf{W}$ . Вхід нейрона  $u_i$  визначає відстань між вектором входу мережі  $\mathbf{p}$  і вектором-прототипом  $i\mathbf{w}$ , при цьому вектори  $\mathbf{p}$  і  $i\mathbf{w}$  є нормалізованими (тобто їхня довжина дорівнює одиниці). Функція активації  $\mathbf{y} = \text{compet}(\mathbf{W}\mathbf{p})$  для нейрона, ваговий вектор якого вказує в напрямку найближчого вхідного вектора, набуває значення «1». Спроекуємо класифікатор, вибравши як рядки матриці  $\mathbf{W}$  вектори-прототипи образів.

Розглянемо правило навчання Інстар за умови, що вектори-прототипи заздалегідь невідомі:  $i\mathbf{w}(q) = i\mathbf{w}(q-1) + \alpha y_i(q)(\mathbf{p}(q) - i\mathbf{w}(q-1))$ . Це правило

можна використати для налаштування значень ваги конкурентної мережі. У конкурентній мережі вихід  $y$  має ненульове значення тільки для нейрона-переможця (наприклад, нейрона з номером  $i = i^*$ ). Тому можна одержати аналогічні результати, використовуючи правило Кохонена:

1) якщо  $i = i^*$ , маємо

$${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i \mathbf{w}(q-1)) = (1 - \alpha) {}_i \mathbf{w}(q-1) + \alpha \mathbf{p}(q);$$

2) якщо  $i \neq i^*$ , маємо  ${}_i \mathbf{w}(q) = {}_i \mathbf{w}(q-1)$ .

Отже, рядок  $i^* \mathbf{w}$  вагової матриці  $\mathbf{W}$ , який є найближчим до вектора входу (має найбільше значення скалярного добутку  $i^* \mathbf{w}^T \mathbf{p}$ ), переміщується в напрямку вектора входу  $\mathbf{p}$  уздовж лінії, яка сполучає старе значення  ${}_i \mathbf{w}(q-1)$  рядка вагової матриці  $\mathbf{W}$  з вектором входу  $\mathbf{p}$  (рис. 11.3).

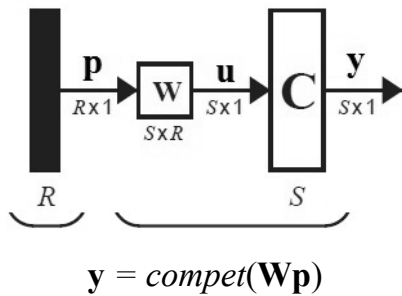


Рис. 11.2. Конкурентна одношарова мережа

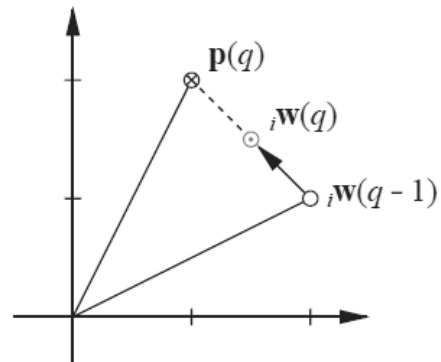


Рис. 11.3. Графічне зображення правила Кохонена

**Приклад 11.2.** Розглянемо, як конкурентний шар навчається класифікувати образи (рис. 11.4), задані такою множиною нормалізованих векторів:

$$\mathbf{p}_1 = \begin{bmatrix} -0,1961 \\ 0,9806 \end{bmatrix}; \mathbf{p}_2 = \begin{bmatrix} 0,1961 \\ 0,9806 \end{bmatrix}; \mathbf{p}_3 = \begin{bmatrix} 0,9806 \\ 0,1961 \end{bmatrix}; \mathbf{p}_4 = \begin{bmatrix} 0,9806 \\ -0,1961 \end{bmatrix};$$

$$\mathbf{p}_5 = \begin{bmatrix} -0,5812 \\ 0,8137 \end{bmatrix}; \mathbf{p}_6 = \begin{bmatrix} -0,8137 \\ -0,5812 \end{bmatrix}.$$

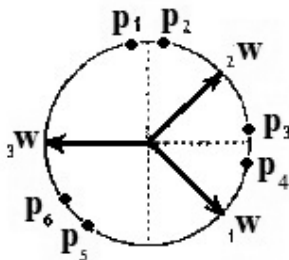


Рис. 11.4. Вектори входу та початкові значення вагових векторів

Нехай мережа має три нейрони і може класифікувати задані вектори на три класи. Випадковим чином було вибрано нормалізовані початкові значення ваги:

$${}_1 \mathbf{w} = \begin{bmatrix} 0,7071 \\ -0,7071 \end{bmatrix};$$

$${}_2 \mathbf{w} = \begin{bmatrix} 0,7071 \\ 0,7071 \end{bmatrix}; {}_3 \mathbf{w} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}; \mathbf{W} = \begin{bmatrix} {}_1 \mathbf{w}^T \\ {}_2 \mathbf{w}^T \\ {}_3 \mathbf{w}^T \end{bmatrix}. \text{ Подавши}$$



вектор  $\mathbf{p}_2$  на вхід мережі, одержимо:

$$\begin{aligned} \mathbf{y} = \text{compet}(\mathbf{W}\mathbf{p}_2) &= \text{compet}\left(\begin{bmatrix} 0,7071 & -0,7071 \\ 0,7071 & 0,7071 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0,1961 \\ 0,9806 \end{bmatrix}\right) = \\ &= \text{compet}\left(\begin{bmatrix} -0,5547 \\ 0,8321 \\ -0,1961 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \end{aligned}$$

Ваговий вектор другого нейрона є найближчим до вектора входу  $\mathbf{p}_2$ , тому він виграє змагання (тобто  $i^* = 2$ ) і його вихід  $y_2 = 1$ . Застосувавши правило навчання Кохонена до нейрона-переможця з коефіцієнтом навчання  $\alpha = 0,5$ , одержимо:

$$\begin{aligned} {}_2\mathbf{w}^{\text{new}} &= {}_2\mathbf{w}^{\text{old}} + \alpha(\mathbf{p}_2 - {}_2\mathbf{w}^{\text{old}}) = \begin{bmatrix} 0,7071 \\ 0,7071 \end{bmatrix} + \\ &+ 0,5 \cdot \left( \begin{bmatrix} 0,1961 \\ 0,9806 \end{bmatrix} - \begin{bmatrix} 0,7071 \\ 0,7071 \end{bmatrix} \right) = \begin{bmatrix} 0,4516 \\ 0,8438 \end{bmatrix}. \end{aligned}$$

Правило Кохонена переміщує вектор  ${}_2\mathbf{w}$  ближче до вектора  $\mathbf{p}_2$ . Якщо продовжити випадковим чином подавати на вхід мережі задані вектори, то на кожній ітерації ваговий вектор, який є найближчим до вектора входу, буде переміщуватися в напрямку вхідного вектора. Поступово кожний ваговий вектор буде вказувати на різні групи вхідних векторів і стане вектором-прототипом відповідної групи.

Можна спрогнозувати, якому кластеру відповідатиме кожний ваговий вектор (на рис. 11.5 зображено їх кінцеві значення). Якщо мережа навчилася класифікувати задані вхідні вектори, то вона відповідно буде класифікувати і нові вектори: затемнення на рис. 11.5 показують, якій області відповідає кожний нейрон. Конкурентний шар відносить кожний вхідний вектор  $\mathbf{p}$  до певних кластерів, забезпечуючи значення виходу «1» для нейрона, ваговий вектор якого розміщений найближче до заданого вектора входу  $\mathbf{p}$ .

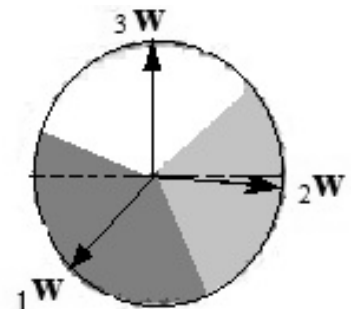


Рис. 11.5. Кінцеві значення вагових векторів (після навчання)

Таким чином, конкурентні одношарові мережі є ефективними адаптивними класифікаторами, але вони мають певні недоліки, серед яких:

- 1) проблема вибору коефіцієнта навчання  $\alpha$ , який впливає на швидкість навчання та стабільність кінцевих значень вагових векторів;
- 2) якщо кластери розміщені близько один до одного, то існує мож-

ливість неправильної кластеризації;

3) кількість кластерів у конкурентному шарі збігається з кількістю його нейронів (це може бути неприйнятним, коли кількість кластерів заздалегідь невідома);

4) початкове значення вагового вектора нейрона може бути так далеко від вхідних векторів, що він ніколи не переможе інші нейрони, у результаті цього з'являються «мертві» нейрони. Вирішити останню проблему можна, додавши до входу нейрона від'ємне значення зсуву, яке потім необхідно зменшувати у разі перемоги нейрона. Коли нейрон часто перемагає, це ускладнює процес одержання ним перемоги. Такий механізм називають «совістю» (рис. 11.6).

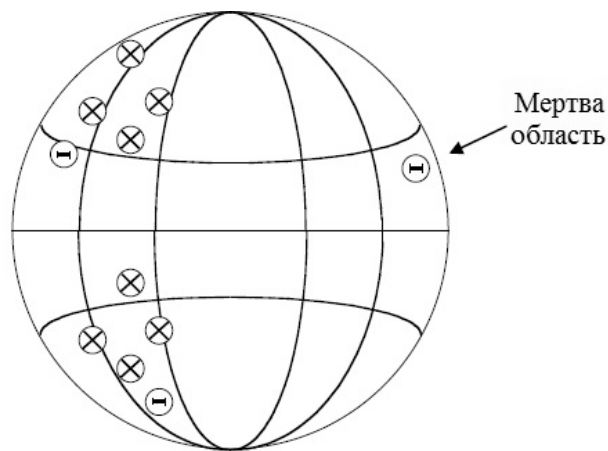


Рис. 11.6. Приклад мертвого нейрона:  
⊗ – вектори входу; Θ – вагові вектори

Приклад алгоритму навчання мережі, який враховує нечутливість «мертвих» нейронів, наведено у дод. Б. Деякі із зазначених проблем вирішуються за допомогою самоорганізованих мереж у вигляді мап Кохонена.

### 11.3. Самоорганізовані мережі: шар і мапа Кохонена

У процесі аналізу великих масивів даних виникають проблеми, пов'язані з дослідженням топологічної структури даних, їхнім об'єднанням у класи, розподілом по класах тощо. Такі задачі можна успішно розв'язати за допомогою застосування спеціального класу НМ – *самоорганізованих*. Властивість самоорганізації мають мережі, описані фінським ученим Т. Кохоненом. Нейрони самоорганізованої мережі можна навчити виявляти кластери векторів входу, які мають деякі загальні властивості. Вивчаючи мережі Кохонена, необхідно розрізняти:

1) мережі з невпорядкованими нейронами, які називають *шарами Кохонена*;

2) мережі з упорядкованими нейронами, які називають *мапами Кохонена* (вони відображають структуру даних таким чином, що близько розміщеним на мапі кластерам даних відповідають близько розміщені нейрони).

**Шар Кохонена.** Розглянемо самоорганізовану одношарову мережу, завдання якої полягає у правильному групуванні векторів входу. Шар Кохонена (рис. 11.7) є шаром конкуруючого типу, оскільки в ньому застосовується конкуруюча ФА. Він включає блок *ndist* для обчислення відстані Евкліда між векторами входу  $\mathbf{p}$  і рядками вагової матриці  $\mathbf{W}^1$ .

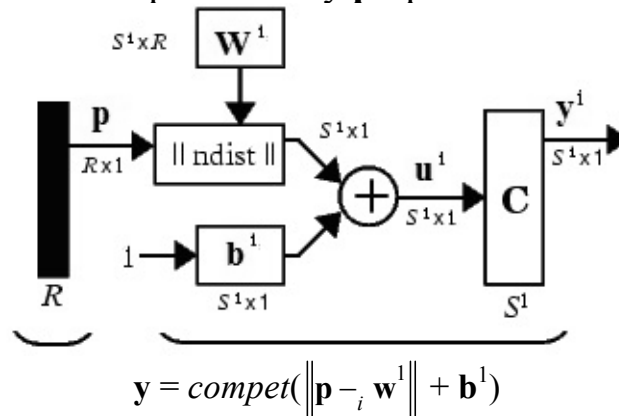


Рис. 11.7. Архітектура одношарової мережі конкуруючого типу

Вхід функції активації  $\mathbf{u}^1$  є сумою обчисленої відстані та вектора зсуву  $\mathbf{b}^1$ . Конкуруюча ФА аналізує елементи вектора  $\mathbf{u}^1$  та формує виходи нейронів: «1» – для нейрона-переможця, який має на вході максимальне значення; «0» – для всіх інших нейронів. Таким чином, вектор виходу  $\mathbf{y}^1$  має такий вигляд:  $y_i^1 = \begin{cases} 1, i = i^*; \\ 0, i \neq i^*, \end{cases}$  де  $i^* = \arg \max_{i=1, \dots, S^1} u_i^1$ . Номер активного нейрона  $i^*$  визначає кластер, до якого вхідний вектор найближчий.

Тепер, коли самоорганізована мережа сформована, потрібно навчити її вирішувати завдання кластеризації даних. Нагадаємо, що кожний нейрон блоку *C* (*compet*) змагається за право включити у свій кластер відповідний вектор входу  $\mathbf{p}$ . Якщо всі зсуви дорівнюють нулю, то нейрон, ваговий вектор якого є найближчим до вектора входу  $\mathbf{p}$ , виграє змагання та повертає на виході значення «1», усі інші нейрони повертають значення «0». Правило навчання шару Кохонена налаштовує потрібним чином елементи вагової матриці.

Припустимо, що  $i$ -й нейрон переміг у разі введення до мережі вектора входу  $\mathbf{p}(q)$  на  $q$ -му кроці самонавчання. Тоді значення  $i$ -го рядка вагової матриці  $\mathbf{W}^1$  змінюється за правилом Кохонена таким чином:

$${}_i \mathbf{w}^1(q) = {}_i \mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - {}_i \mathbf{w}^1(q-1)).$$

Отже, ваговий вектор, який є найближчим до вектора входу, модифікується таким чином, щоб відстань між ними стала меншою. В остаточному підсумку кожна група близьких векторів виявиться пов'язаною з одним із нейронів шару. У цьому і полягає *властивість самоорганізації шару Кохонена*.

Розглянута мережа Кохонена є типовим прикладом мережі, яка реалізує процедуру навчання без учителя за алгоритмом «переможець одержує все». Нейрон-переможець обирається за максимальним значенням рівня активації нейрона, яке залежить від близькості вагового вектора до вектора входу. Серед усіх нормалізованих вагових векторів максимальний рівень активації  ${}_i \mathbf{w}^T \mathbf{p}$  відповідає векторам з мінімальним значенням відстані Екліда  $\|\mathbf{p} - {}_i \mathbf{w}\|$ . Щоб більш точно відповідати вхідному вектору, ваговий вектор нейрона-переможця модифікується. Серед інших існують такі *модифікації алгоритма «переможець одержує все»*: додавання параметра «совість»; визначення не одного, а декількох нейронів-переможців; диференційоване налаштування вагових векторів усіх нейронів з околу нейрона-переможця.

**Мапа Кохонена.** У попередніх розділах не розглядалося питання щодо визначення топології мережі (як нейрони фізично організовані в межах шару). У біологічних нервових системах нейрони, зазвичай, розміщені у двовимірних шарах, в яких вони щільно пов'язані. На рис. 11.8 зображено шар із 25 нейронів, розміщених у вигляді двовимірної ґратки.

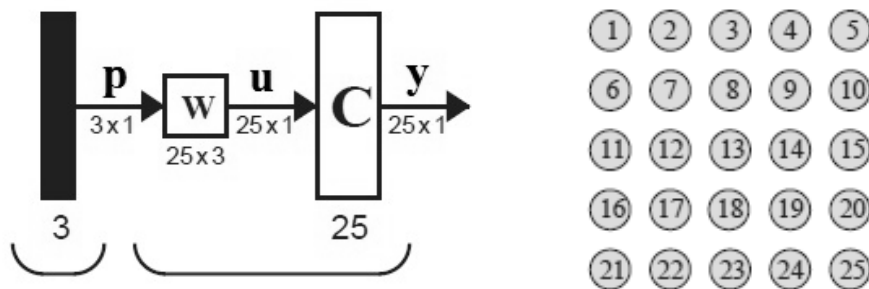


Рис. 11.8. Одношарова мапа Кохонена  $\mathbf{y} = \text{compet}(\mathbf{W}\mathbf{p})$

Зазвичай вага змінюється як функція відстані між з'єднаними нейронами. Наприклад, якщо вага другого шару мережі Хеммінга визначається як

$$w_{ij} = \begin{cases} 1, & \text{якщо } i = j; \\ -\varepsilon, & \text{якщо } i \neq j, \end{cases} \quad (11.3)$$

то ця вага з погляду відстані  $d_{ij}$  між нейронами визначається таким

чином:  $w_{ij} = \begin{cases} 1, & \text{якщо } d_{ij} = 0, \\ -\varepsilon, & \text{якщо } d_{ij} > 0. \end{cases}$  Кожному  $i$ -му нейрону відповідає значення ваги  $w_{ij}$ , яке вказує на силу зв'язку між ним та  $j$ -м нейроном.

У біології нейрон посилює не тільки себе, але й нейрони, які розміщені поруч із ним. Замість одного нейрона-переможця біологічні нервові системи зазвичай активізують групу нейронів навколо найактивнішого нейрона. Найчастіше перехід від посилення до послаблення відбувається гладко (рис. 11.9), оскільки відстань між нейронами збільшується. Термін «у центрі» часто використовують для описання схеми зв'язків між нейронами.

Графік функціональної залежності відстані між нейронами від їхньої ваги («Мексиканський капелюх») зображено на рис. 11.10. Зв'язок нейрона-переможця з іншими нейронами зменшується пропорційно збільшенню відстані між ними. Щоб моделювати активність груп нейронів у біологічних системах, Кохонен розробив деякі спрощення: його самоорганізована мережа (використовуючи ту ж процедуру, що і конкурентний шар) спочатку визначає  $i^*$ -й нейрон-переможець, а потім оновлює значення вагових векторів усіх нейронів з околу визначеного нейрона-переможця (рис. 11.11) за правилом Кохонена:  $i\mathbf{w}(q) = i\mathbf{w}(q - 1) + \alpha(\mathbf{p}(q) - i\mathbf{w}(q - 1)) = (1 - \alpha)i\mathbf{w}(q - 1) + \alpha\mathbf{p}(q)$ ,  $i \in N_{i^*}(d)$ , де окіл  $N_{i^*}(d)$  містить номери нейронів, розміщені в межах радіуса  $d$  до  $i^*$ -го нейрона-переможця:  $N_i(d) = \{j, d_{ij} \leq d\}$ . Якщо на вхід мережі подати вектор  $\mathbf{p}$ , то вагові вектори нейрона-переможця та його сусідів будуть переміщуватися у напрямку вектора  $\mathbf{p}$ . Результатом такого навчання є те, що нейрони-сусіди стануть схожими.

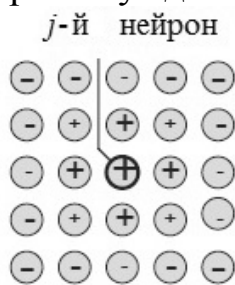


Рис. 11.9. Модель шару нейронів «у центрі»

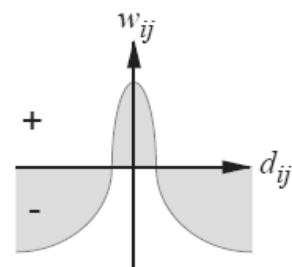


Рис. 11.10. Графік функції «Мексиканський капелюх»

У самоорганізованих мапах (self-organizing map – SOM) нейрони розміщуються у вузлах деякої ґратки (найчастіше одно- або двовимірної). Мапи з розмірністю більшою за два використовують зрідка. Під час конкуренції нейрони вибірково налаштовуються на різні вхідні образи. У такий спосіб налаштовані нейрони-переможці впорядковуються так, що на ґратці створюється система координат.

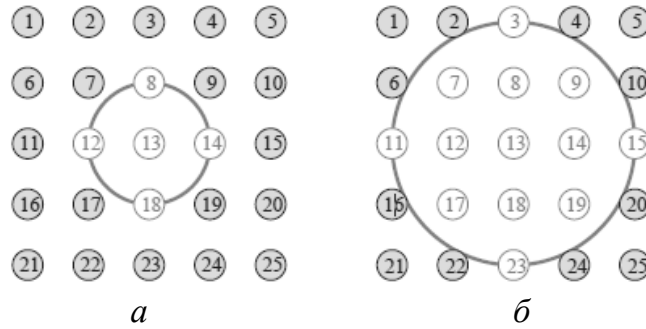


Рис. 11.11. Ілюстрація поняття «окіл нейрона»:  $a - N_{13}(1)$ ;  $b - N_{13}(2)$

Отже, самоорганізовані системи характеризуються формуванням *топографічних мап* вхідних образів. Просторове розміщення вихідних нейронів у топографічній мапі відповідає конкретній області ознак даних, виділених із вхідного простору. Самоорганізація зумовлює топологічно впорядковане відображення, при цьому розмірності вхідного й вихідного сигналів можуть бути різними. Цей принцип покладений в основу моделей відображення мапи ознак.

**Алгоритм навчання самоорганізованих мап** [12]. Основна мета самоорганізованих алгоритмів навчання полягає у виявленні у множині вхідних даних істотних ознак без участі вчителя. Для цього алгоритм реалізує множину правил локальної природи, що дозволяє навчатися обчислювати відображення вхідного сигналу на вихідний з потрібними властивостями. При цьому зміна ваги нейрона визначається тільки безпосередніми сусідами цього нейрона і виконується перетворення вхідних векторів, які мають довільну розмірність, в одно- або двовимірну дискретну мапу (таке перетворення здійснюється адаптивно, у топологічно впорядкованій формі). Алгоритм розпочинається з ініціалізації ваги мережі, що, зазвичай, відбувається за допомогою присвоєння ваговим векторам малих значень, сформованих генератором випадкових чисел. Після ініціалізації запускаються такі три процеси: *конкуренція*, *кооперація*, *синаптична адаптація*. Опишемо їх детально.

**Процес конкуренції.** Нехай  $R$  – розмірність вхідного простору, а вхідний вектор  $\mathbf{p} = [p_1 \dots p_R]^T$  обирається з цього простору випадковим чином. Позначимо ваговий вектор  $j$ -го нейрона як  ${}_j\mathbf{w} = [w_{j1} \dots w_{jR}]^T$ ,  $j = 1, \dots, S$ , де  $S$  – кількість нейронів (шару). Щоб вибрати найкращий вектор  ${}_j\mathbf{w}$ , який відповідає вхідному вектору  $\mathbf{p}$ , необхідно порівняти значення скалярних добутків  ${}_j\mathbf{w}^T \mathbf{p}$  для  $j = 1, \dots, S$  і вибрати найбільше з них. У результаті вибору нейрона з найбільшим значенням скалярного

добутку  ${}_j \mathbf{w}^T \mathbf{p}$  визначається положення центру топологічного околу нейрона, який був активований. Зазначимо, що найкращий критерій відповідності, оснований на максимізації значення скалярного добутку  ${}_j \mathbf{w}^T \mathbf{p}$ , математично еквівалентний мінімізації відстані Евкліда між векторами  $\mathbf{p}$  і  ${}_j \mathbf{w}$ . При цьому номер  $i(\mathbf{p})$  нейрона-переможця, що найкраще відповідає вхідному вектору  $\mathbf{p}$ , можна визначити таким чином:

$$i(\mathbf{p}) = \arg \min_j \|\mathbf{p} - {}_j \mathbf{w}\|, j = 1, \dots, S. \quad (11.4)$$

Отже, у результаті процесу конкуренції неперервний вхідний простір між нейронами мережі відображається в дискретний вихідний простір, а виходом мережі може бути або індекс нейрона-переможця (тобто його позиція у ґратці), або ваговий вектор, розміщений найближче до вхідного вектора відносно відстані Евкліда.

**Процес кооперації.** Нейрон-переможець перебуває у центрі топологічного околу нейронів, які взаємодіють між собою. Визначимо топологічний окіл з нейробіологічного погляду. Для цього нагадаємо нейробіологічне обґрунтування *латеральної взаємодії* між активованими нейронами: активований нейрон завжди намагається зробити активними нейрони поблизу нього. Введемо такі позначення:  $d_{ji}$  – латеральна відстань між нейроном-переможцем ( $i$ ) та вторинно активованими нейронами ( $j$ ),  $h_{ji}$  – топологічний окіл із центром в  $i$ -му нейроні-переможці, який складається із множини вторинно активованих нейронів ( $j$ ), що взаємодіють із ним. Тоді можна припустити, що топологічний окіл  $h_{ji}$  є унімодальною функцією від латеральної відстані  $d_{ji}$  і задовольняє такі умови:

1)  $h_{ji}$  є симетричною відносно точки максимуму, яка визначається умовою  $d_{ji} = 0$  (максимум функції досягається в нейроні-переможці);

2) амплітуда топологічного околу  $h_{ji}$  монотонно зменшується зі збільшенням латеральної відстані  $d_{ji}$ , прямуючи до нуля за  $d_{ji} \rightarrow \infty$  (необхідна умова збіжності).

Типовою функцією, яка задовольняє ці вимоги, є *функція Гаусса*:

$$h_{ji}(d_{ji}) = \exp\left(-\frac{d_{ji}^2}{2\sigma^2}\right) = \exp\left(-\frac{\|\mathbf{p}_i - {}_j \mathbf{w}\|^2}{2\sigma^2}\right), \quad (11.5)$$

де параметр  $\sigma$  – *ефективна ширина* топологічного околу (рис. 11.12), що визначає рівень, до якого нейрони із топологічного околу нейрона-переможця беруть участь у процесі навчання. У дискретному вихідному просторі латеральну відстань  $d_{ji}$  можна визначити таким чином: для одно

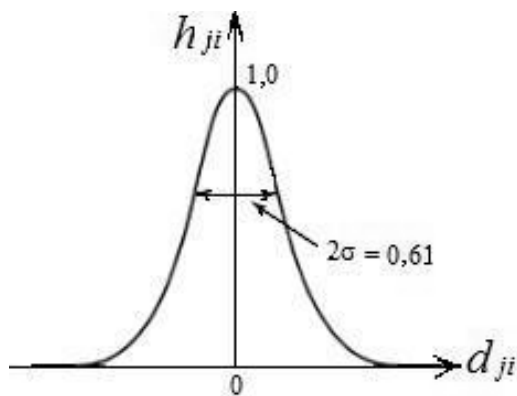


Рис. 11.12. Функція Гаусса  
топологічного околу

вимірної ґратки  $d_{ji} = |j-i|$ ; для двовимірної ґратки  $d_{ji} = \|\mathbf{r}_j - \mathbf{r}_i\|$ , де вектор  $\mathbf{r}_j$  визначає положення активованого  $j$ -го нейрона, а  $\mathbf{r}_i$  –  $i$ -го нейрона-переможця.

Важлива властивість самоорганізованого алгоритму навчання полягає у зменшенні розміру топологічного околу з часом і виконується за рахунок поступового зменшення ефективної ширини  $\sigma$  функції топологічного околу  $h_{ji}$ .

Останню найчастіше обчислюють

за формулою  $\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right)$ ,  $t = 1, 2, \dots$ , де  $\sigma_0$  – початкове значення

$\sigma$ ;  $\tau_1$  – деяка часова константа. Залежність топологічного околу від часу визначають формулою

$$h_{ji(\mathbf{p})}(t) = \exp\left(-\frac{d_{ji}^2}{2\sigma^2(t)}\right), t = 0, 1, \dots \quad (11.6)$$

Отже, зі збільшенням кількості ітерацій  $t$  ширина  $\sigma(t)$  експоненціально зменшується, а тому відповідно зменшується й топологічний окіл. Далі функцію  $h_{ji}$  будемо називати *функцією околу*.

**Процес адаптації.** Щоб мережа могла самоорганізуватися, ваговий вектор  $j$ -го нейрона  $\mathbf{w}$  повинен змінюватися відповідно до вхідного вектора  $\mathbf{p}$ . Питання зводиться до того, якою має бути ця зміна.

У правилі асоціативного навчання Хебба вага збільшується з одночасним виникненням передсинаптичної та постсинаптичної активності. Однак правило Хебба в основній формі не задовольняє вимоги навчання без учителя, оскільки зміна зв'язків відбувається тільки в одному напрямку, який зводить всі значення ваги до точки насичення. Щоб вирішити цю проблему, змінимо правило Хебба, додавши *елемент забування*  $g(y_j)\mathbf{w}$ , де  $g(y_j)$  – деяка додатна скалярна функція виходу  $y_j$ , яка повинна задовольняти таку умову:

$$g(0) = 0. \quad (11.7)$$

Тоді зміну вагового вектора  $j$ -го нейрона у ґратці можна записати у вигляді

$$\Delta_j \mathbf{w} = \alpha y_j \mathbf{p} - g(y_j) \mathbf{w}, \quad (11.8)$$

де  $\alpha$  – коефіцієнт швидкості навчання алгоритму. Як функцію  $g(y_j)$  обемо, наприклад, лінійну функцію  $g(y_j) = \alpha$ , яка задовольняє співвідношення (11.7). Вираз (11.8) можна спростити, вибравши  $y_j = h_{ji(\mathbf{p})}$ . Тоді



одержимо:  $\Delta_j \mathbf{w} = \alpha h_{ji(\mathbf{p})}(\mathbf{p} - \mathbf{w})$ . Враховуючи формалізацію дискретного часу для вагового вектора  ${}_j \mathbf{w}(t)$  у момент часу  $t$ , оновлений вектор  ${}_j \mathbf{w}(t+1)$  можна визначити таким чином:

$${}_j \mathbf{w}(t+1) = {}_j \mathbf{w}(t) + \alpha(t) h_{ji(\mathbf{p})}(t)(\mathbf{p} - {}_j \mathbf{w}(t)). \quad (11.9)$$

Вираз (11.9) застосовується до всіх нейронів ґратки, розміщених у топологічному околі  $i(\mathbf{p})$ -го нейрона-переможця  $h_{ji(\mathbf{p})}$ , та означає переміщення вагового вектора  $i(\mathbf{p})$ -го нейрона-переможця у бік вхідного вектора  $\mathbf{p}$ . Параметр  $\alpha(t)$  залежить від часу, розпочинається з деякого початкового значення  $\alpha_0$ , яке поступово зменшується зі зміною часу, наприклад, за формулою

$$\alpha(t) = \alpha_0 \exp\left(-\frac{t}{\tau_2}\right), \quad t = 0, 1, \dots, \quad (11.10)$$

де  $\tau_2$  – деяка часова константа.

Отже, *істотними характеристиками алгоритму самоорганізації map є такі:*

- 1) неперервний вхідний простір образів активації, які формуються у відповідності з деяким розподілом ймовірності;
- 2) топологія мережі у формі ґратки, яка складається з нейронів і визначає дискретний вихідний простір;
- 3) функція околу  $h_{ji(\mathbf{p})}(t)$ , яка залежить від часу та визначена в околі нейрона-переможця  $i(\mathbf{p})$ ;
- 4) параметр швидкості навчання  $\alpha(t)$ , для якого задається початкове значення  $\alpha_0$ , яке поступово зменшується з часом  $t$ , але ніколи не досягає нуля.

Для функції околу та коефіцієнта швидкості навчання на етапі впорядкування можна використовувати формули (11.6) і (11.10) відповідно. Для більшої статистичної точності на етапі збіжності параметр  $\alpha(t)$  зазвичай встановлюють на рівні  $\alpha_0 = 0,01$ . На початку алгоритму функція околу повинна містити тільки найближчих сусідів нейрона-переможця, або тільки його одного.

**Алгоритм навчання самоорганізованих map** можна описати таким чином.

*Постановка задачі (кластерний аналіз або класифікація без учителя).* Нехай задано навчальну вибірку у вигляді даних входу:  $A = \{\mathbf{p}^1, \dots, \mathbf{p}^Q\}$ , де  $\mathbf{p}^i = [p_1^i \dots p_R^i]^T$  – вектор вхідних даних з елементами  $p_j^i \in \mathfrak{R}$ . На основі початкових даних визначити таке розбиття  $P(A) = \{A_k | A_k \subseteq A\}$  множи-

ни  $A$  на задану кількість кластерів  $S$  ( $S \in N$ ,  $S > 1$ ):  $A_k$ ,  $k = 1, \dots, S$ , яке забезпечує екстремум деякої цільової функції  $f(P(A))$  серед усіх розбиттів. Як цільову функцію будемо розглядати функцію вигляду  $f(P(A), \mathbf{W}) =$

$$= \sum_{i=1}^Q \sum_{k=1}^S \exp \left( -\frac{\|\mathbf{p}^i - {}_k\mathbf{w}\|^2}{2\sigma^2} \right), \text{ де } {}_k\mathbf{w} = [{}_k w_1 \dots {}_k w_R]^T \text{ є вектором у просторі } \mathbb{R}^R;$$

$f(P(A), \mathbf{W})$  – функція, реалізована у вигляді НМ, яка описує перетворення вхідних даних у вихід. Розглянемо алгоритм.

*Крок 1. Ініціалізація.* Як початкові значення вагових векторів  $\{ {}_j\mathbf{w}(0) \}_{j=1}^{S^1}$  обираємо випадкові значення (при цьому амплітуду значень рекомендується брати малою) або значення з доступної множини вхідних векторів  $\{ \mathbf{p}^i \}_{i=1}^R$ .

*Крок 2. Вибір вхідного вектора.* Вхідний вектор  $\mathbf{p}$ , який має розмірність  $R$ , обираємо із вхідного простору з визначеною ймовірністю.

*Крок 3. Визначення нейрона-переможця.* На кроці  $t$  визначаємо значення  $i(\mathbf{p})$  нейрона-переможця, використовуючи критерій мінімуму відстані Евкліда:

$$i(\mathbf{p}) = \arg \min_{j=1, \dots, S^1} \|\mathbf{p} - {}_j\mathbf{w}\|.$$

*Крок 4. Корекція.* Значення вагових векторів усіх нейронів модифікуємо за формулою (11.9). Повертаємося до кроку 2 та продовжуємо обчислення доти, поки в мапі ознак не перестануть спостерігатися суттєві зміни.

## 11.4. Мережа квантування векторів LVQ

Мережа квантування векторів Learning Vector Quantization (LVQ – двоступінчатий адаптивний класифікатор (рис. 11.13)) – це гібридна мережа, яку використовують як для неконтрольованого, так і для контрольованого навчання.

*Квантування* – це перетворення заданої величини з неперервною шкалою значень на величину з дискретною шкалою значень. У мережі LVQ кожний нейрон відповідає певному класу (зазвичай одному класу відповідають декілька нейронів). Номер кожного класу присвоюється одному нейрону другого шару. Якщо кількість нейронів першого шару дорівнює  $S^1$ , а другого –  $S^2$ , то, як правило,  $S^1 \geq S^2$ .

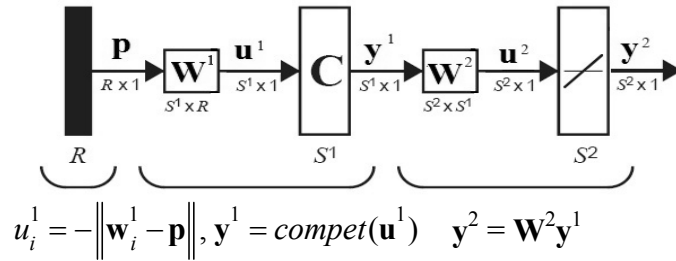


Рис. 11.13. Мережа LVQ

У першому конкурентному шарі мережі LVQ кожний нейрон вивчає вхідний вектор-прототип, який дозволяє йому класифікувати область вхідного простору, однак замість обчислення близькості входу до вагового вектора за допомогою скалярного добутку мережа LVQ обчислює безпосередньо відстань. У цьому разі вектор входу не потрібно нормалізувати. Вхід і вихід першого шару мережі LVQ, відповідно, мають

вигляд:  $\mathbf{u}^1 = -\begin{bmatrix} \|\mathbf{w}_1^1 - \mathbf{p}\| \\ \dots \\ \|\mathbf{w}_{S^1}^1 - \mathbf{p}\| \end{bmatrix}$  та  $\mathbf{y}^1 = \text{compet}(\mathbf{u}^1)$ . Таким чином, вихід ней-

рона з ваговим вектором, який є найближчим до вектора входу, дорівнює одиниці, а виходи інших нейронів – нулю.

Функціонування першого шару мережі LVQ збігається з функціонуванням одношарової мережі конкурентного типу (принаймні для нормалізованих векторів), в яких нейрон з відмінним від нуля виходом показує, до якого класу належить вхідний вектор. Однак є відмінність в інтерпретації одержаного результату:

1) у мережах конкурентного типу нейрон з ненульовим виходом показує, до якого класу належить вхідний вектор;

2) у мережі LVQ перемагає нейрон, який зображує підклас (замість класу), при цьому може бути декілька різних нейронів (підкласів), що входять до складу певного класу.

Другий шар мережі LVQ використовується для об'єднання підкласів в один клас за допомогою матриці  $\mathbf{W}^2$ , стовпці якої зображують підкласи, а рядки – класи. Матриця  $\mathbf{W}^2$  має в кожному стовпці одну одиницю, а інші її елементи дорівнюють нулю. Рядок, в якому міститься одиниця, вказує підклас, який належить до відповідного класу: якщо  $w_{ki}^2 = 1$ , то  $i$ -й підклас належить до класу  $k$ . Наприклад, матриця

вигляду  $\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$  містить підкласи 1, 3, 4 класу 1;

підклас 2 класу 2; підкласи 5, 6 класу 3. Процес об'єднання підкласів у класи дозволяє мережі LVQ утворювати множину границь поділу елементів класу. Стандартний шар конкурентного типу може утворювати лише опуклі ділянки, а мережа LVQ долає це обмеження.

**Метод навчання на основі дискретизації навчальних векторів (ДНВ).** Метод ДНВ об'єднує конкурентне навчання з керованим. Як і всі алгоритми керованого навчання, він вимагає наявності множини прикладів цільової (бажаної) поведінки мережі:  $\{\mathbf{p}_1, \mathbf{t}_1\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$ . При цьому всі елементи цільового вектора, крім одного, дорівнюють нулю, а один елемент – одиниці. Рядок, в якому міститься одиниця, позначає клас, до якого належить вектор. Наприклад, якщо необхідно класифікувати вектор  $\mathbf{p}_1$ , який складається з трьох елементів і належить класу 2 з чотирьох класів, то вхідні дані можна записати у вигляді

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} \sqrt{0,5} & 0 & \sqrt{0,5} \end{bmatrix}^T, \mathbf{t}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \right\}.$$

Перед навчанням може бути задана відповідність кожного нейрона першого шару нейрону вихідного шару, що дозволяє сформувати матрицю  $\mathbf{W}^2$ . Зазвичай однакова кількість нейронів прихованого шару відповідає одному вихідному нейрону, тому кожний клас повинен складатися з однакової кількості опуклих областей. Визначивши одного разу матрицю  $\mathbf{W}^2$ , її не можна змінювати. Значення вагової матриці  $\mathbf{W}^1$  оновлюються за допомогою різновидів правила Кохонена. Під час кожної ітерації навчання алгоритм ДНВ передбачає такі дії:

*Крок 1.* На вхід мережі подамо вектор  $\mathbf{p}$  й обчислимо відстань від  $\mathbf{p}$  до кожного вектора-прототипу  $\mathbf{w}$ .

*Крок 2.* Нейрони прихованого шару змагаються та виграве  $i^*$ -й нейрон;  $i^*$ -й елемент вектора  $\mathbf{y}^1$  набуває значення «1».

*Крок 3.* Для отримання остаточного значення виходу  $\mathbf{y}^2$  вектор  $\mathbf{y}^1$  множимо на матрицю  $\mathbf{W}^2$ , яка має тільки один ненульовий елемент  $k^*$  (тобто вектор  $\mathbf{p}$  належить до класу  $k^*$ ).

Правило Кохонена використовують для поліпшення функціонування прихованого шару мережі таким чином:

1) якщо вектор входу  $\mathbf{p}$  класифікований правильно, то ваговий вектор  $_{i^*}\mathbf{w}^1$  нейрона-переможця прихованого шару переміщується до вектора  $\mathbf{p}$ :

$$_{i^*}\mathbf{w}^1(q) = _{i^*}\mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - _{i^*}\mathbf{w}^1(q-1)), \text{ якщо } y_{k^*}^2 = t_{k^*} = 1;$$

2) якщо вектор входу  $\mathbf{p}$  класифікований неправильно, то ваговий вектор  $_{i^*}\mathbf{w}^1$  переміщується подалі від вектора  $\mathbf{p}$ :

$$_{i^*}\mathbf{w}^1(q) = _{i^*}\mathbf{w}^1(q-1) - \alpha(\mathbf{p}(q) - _{i^*}\mathbf{w}^1(q-1)), \text{ якщо } y_{k^*}^2 = 1 \neq t_{k^*} = 0.$$

Результатом функціонування (навчання) мережі LVQ є переміщення вагового вектора кожного нейрона прихованого шару до вектора, який належить певному класу, для якого створюється підклас, і віддалення від вектора, який належить іншим класам.

**Приклад 11.3.** Розглянемо приклад навчання на основі ДНВ для розв'язання задачі класифікації (рис. 11.14):

$$\text{клас 1: } \left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}; \text{ клас 2: } \left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}.$$

Нехай цільові вектори кожного класу мають такий вигляд:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}; \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\};$$

$$\left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}; \left\{ \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}.$$

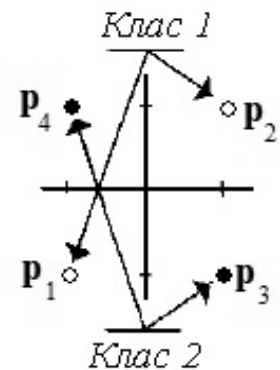


Рис. 11.14. Зображення класів

Необхідно визначити, яким чином множина підкласів утворює кожний із двох класів. Якщо припустити, що кожний клас поєднує два підкласи, то для досягнення мети достатньо чотирьох нейронів у прихованому шарі.

*Розв'язання.* Вихідна вагова матриця  $\mathbf{W}^2$  має вигляд  $\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$  і поєднує перший і другий нейрони прихованого шару з першим нейроном вихідного шару, третій і четвертий нейрони прихованого шару з другим нейроном вихідного шару. Кожний клас будується з двох опуклих областей. Припустимо, що вагова матриця  $\mathbf{W}^1$  складається з таких випадкових векторів:

$$\left\{ {}_1\mathbf{w}^1 = \begin{bmatrix} -0,543 \\ 0,840 \end{bmatrix}, {}_2\mathbf{w}^1 = \begin{bmatrix} -0,969 \\ -0,249 \end{bmatrix}, {}_3\mathbf{w}^1 = \begin{bmatrix} 0,997 \\ 0,094 \end{bmatrix}, {}_4\mathbf{w}^1 = \begin{bmatrix} 0,456 \\ 0,954 \end{bmatrix} \right\}. \text{ Позначимо}$$

білим колом ваговий вектор, який визначає перший клас і належить двом прихованим нейронам; чорним колом – ваговий вектор, який визначає другий клас (рис. 11.15). Нехай на визначеній ітерації процесу навчання на вхід мережі подається вектор  $\mathbf{p}_3$ . Тоді вихід прихованого шару мережі має вигляд

$$\mathbf{y}^1 = \text{compet}(\mathbf{u}^1) = \text{compet} \begin{pmatrix} -\|\mathbf{w}_1^1 - \mathbf{p}_3\| \\ -\|\mathbf{w}_2^1 - \mathbf{p}_3\| \\ -\|\mathbf{w}_3^1 - \mathbf{p}_3\| \\ -\|\mathbf{w}_4^1 - \mathbf{p}_3\| \end{pmatrix} = \text{compet} \begin{pmatrix} -2,40 \\ -2,11 \\ -1,09 \\ -2,03 \end{pmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Ваговий вектор третього нейрона прихованого шару найближче розміщений до вектора  $\mathbf{p}_3$ , тому вихід мережі  $\mathbf{y}^2 = \mathbf{W}^2 \mathbf{y}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . Отже, вектор  $\mathbf{p}_3$  належить до другого класу, і вага  ${}_3\mathbf{w}^1$  оновлюється, переміщуючись ближче до вектора  $\mathbf{p}_3$ :

$${}_3\mathbf{w}^1(1) = {}_3\mathbf{w}^1(0) + \alpha(\mathbf{p}_3 - {}_3\mathbf{w}^1(0)) = \begin{bmatrix} 0,997 \\ 0,094 \end{bmatrix} + 0,5 \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0,997 \\ 0,094 \end{bmatrix} \right) = \begin{bmatrix} 0,998 \\ -0,453 \end{bmatrix}.$$

Класифікацію областей вхідного простору після останньої ітерації зображено на рис. 11.16.

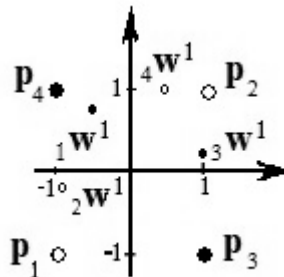


Рис. 11.15. Вектори матриці  $\mathbf{W}^1$

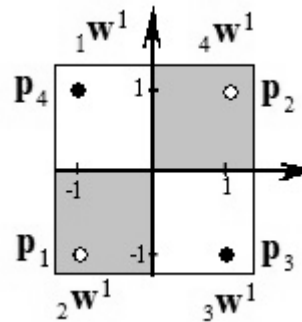


Рис. 11.16. Класифікація після останньої ітерації

Мережі ДНВ вирішують різні завдання, але мають деякі обмеження. Так, наприклад, у конкурентних шарах прихований нейрон мережі може мати початкове значення ваги, яке заважає виграти змагання, у результаті формується «мертвий» нейрон. Цю проблему вирішують за допомогою описаного раніше механізму «совість».

## **Приклади розв'язання задач\***

**Приклад 11.4.** Нехай задано нормалізовані вхідні вектори, які належать до певних класів/кластерів (рис. 11.17).

Спроекувати конкурентну мережу (див. рис. 11.2), яка класифікує задані вектори відповідно до заданих кластерів за мінімальної кількості нейронів. Нарисувати вагові вектори та границі поділу заданих класів.

*Розв'язання.* Оскільки необхідно визначити чотири кластери, то конкурентна мережа повинна мати чотири нейрони. Вага кожного нейрона є прототипом кластера, який він зображує, тому для кожного нейрона доцільно вибрати ваговий вектор, який перебуває приблизно в центрі кластера. Помітивши, що центри 1, 2 та 3-го кластерів розміщені приблизно під кутом  $45^\circ$  до осей координат, можна використати такі три нормалізовані ва-

гові вектори, які вказують потрібні напрямки:  ${}_1\mathbf{w}(0) = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$ ;

${}_2\mathbf{w}(0) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$ ;  ${}_3\mathbf{w}(0) = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^T$ . Оскільки відстань від центру

4-го кластера до вертикальної осі приблизно у два рази більша за відстань до горизонтальної осі, то нормалізований ваговий вектор, який відповідає

цьому кластеру, має вигляд  ${}_4\mathbf{w}(0) = \begin{bmatrix} -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{bmatrix}^T$ .

$$\text{Отже, вагова матриця } \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ {}_3\mathbf{w}^T \\ {}_4\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{bmatrix}.$$

Зобразивши вагові вектори, які ділять навпіл відповідні їм кластери, отримаємо чотири області (рис. 11.18).

---

\* Задачі взято з посібника [14].

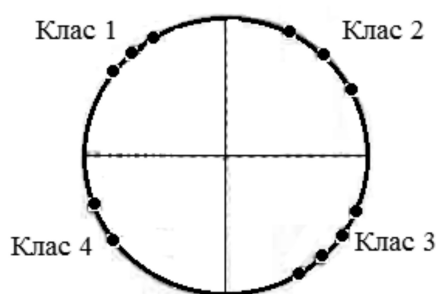


Рис. 11.17. Класи вхідних векторів

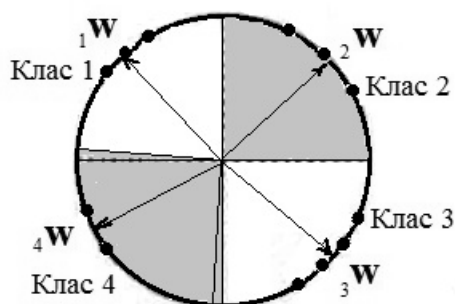


Рис. 11.18. Результуюча кластеризація

**Приклад 11.5.** На рис. 11.19 зображено вектори входу  $\mathbf{p}_1 = [-1 \ 0]^T$ ;  $\mathbf{p}_2 = [0 \ 1]^T$ ;  $\mathbf{p}_3 = \left[\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right]^T$  та початкові значення ваги  ${}_1\mathbf{w}(0) = [0 \ -1]^T$ ,  ${}_2\mathbf{w}(0) = \left[-\frac{2}{\sqrt{5}} \ \frac{1}{\sqrt{5}}\right]^T$ ;  ${}_3\mathbf{w}(0) = \left[-\frac{1}{\sqrt{5}} \ \frac{2}{\sqrt{5}}\right]^T$  для тринейронного конкурентного шару.

Визначити значення ваги після навчання конкурентного шару за правилом Кохонена з коефіцієнтом навчання  $\alpha = 0,5$ , якщо послідовність навчання має такий вигляд:  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ .

*Розв'язання.* Об'єднаємо початкові вагові вектори у матрицю  $\mathbf{W}(0)$  і подамо на вхід мережі вектор  $\mathbf{p}_1$ , одержимо:

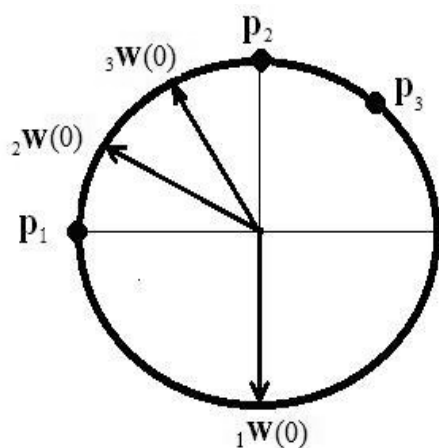


Рис. 11.19. Вхідні вектори та початкові значення ваги

$$\begin{aligned} \mathbf{y} &= \text{compet}(\mathbf{W}(0) \cdot \mathbf{p}_1) = \\ &= \text{compet} \left( \begin{bmatrix} 0 & -1 \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 0 \end{bmatrix} \right) = \\ &= \text{compet} \left( \begin{bmatrix} 0 \\ 0,894 \\ 0,447 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \end{aligned}$$

Переможцем став другий нейрон, оскільки вектор  ${}_2\mathbf{w}(0)$  є найближчим до вектора  $\mathbf{p}_1$  порівняно з іншими ваговими векторами. Обчислимо нове значення  ${}_2\mathbf{w}(1)$  за правилом Кохонена:



$${}_2\mathbf{w}(1) = {}_2\mathbf{w}(0) + \alpha(\mathbf{p}_1 - {}_2\mathbf{w}(0)) = \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ 1 \\ \frac{1}{\sqrt{5}} \end{bmatrix} + 0,5 \left( \begin{bmatrix} -1 \\ 0 \end{bmatrix} - \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ 1 \\ \frac{1}{\sqrt{5}} \end{bmatrix} \right) = \begin{bmatrix} -0,947 \\ 0,224 \end{bmatrix}.$$

Новий вектор  ${}_2\mathbf{w}(1)$  наблизився до вектора  $\mathbf{p}_1$  (рис. 11.20, а). Повторимо описану процедуру для вектора  $\mathbf{p}_2$ . Одержимо:

$$\mathbf{y} = \text{compet}(\mathbf{W}(1) \cdot \mathbf{p}_2) = \text{compet} \left( \begin{bmatrix} 0 & -1 \\ -0,947 & 0,224 \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = \text{compet} \left( \begin{bmatrix} -1 \\ 0,224 \\ 0,894 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Оскільки переміг третій нейрон, то його ваговий вектор наблизився до  $\mathbf{p}_2$ :

$${}_3\mathbf{w}(2) = {}_3\mathbf{w}(1) + \alpha(\mathbf{p}_2 - {}_3\mathbf{w}(1)) = \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ 2 \\ \frac{2}{\sqrt{5}} \end{bmatrix} + 0,5 \left( \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ 2 \\ \frac{2}{\sqrt{5}} \end{bmatrix} \right) = \begin{bmatrix} -0,224 \\ 0,947 \end{bmatrix}.$$

Подамо на вхід мережі вектор  $\mathbf{p}_3$ , одержимо:

$$\begin{aligned} \mathbf{y} = \text{compet}(\mathbf{W}(2) \cdot \mathbf{p}_3) &= \text{compet} \left( \begin{bmatrix} 0 & -1 \\ -0,947 & 0,224 \\ -0,224 & 0,947 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \right) = \\ &= \text{compet} \left( \begin{bmatrix} -0,707 \\ -0,512 \\ 0,512 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \end{aligned}$$

Знову переміг третій нейрон:

$${}_3\mathbf{w}(3) = {}_3\mathbf{w}(2) + \alpha(\mathbf{p}_3 - {}_3\mathbf{w}(2)) = \begin{bmatrix} -0,224 \\ 0,947 \end{bmatrix} + 0,5 \left( \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} - \begin{bmatrix} -0,224 \\ 0,947 \end{bmatrix} \right) = \begin{bmatrix} 0,2417 \\ 0,8272 \end{bmatrix}.$$

Після повторної подачі на вхід мережі векторів  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  і  $\mathbf{p}_3$  другий нейрон переміг один раз, а третій – два рази. Остаточне значення ваги

має вигляд (рис. 11.20, б):  $\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -0,974 & 0,118 \\ 0,414 & 0,8103 \end{bmatrix}$ . Ваговий вектор  ${}_2\mathbf{w}$

майже «вивчив»  $\mathbf{p}_1$ , вектор  ${}_3\mathbf{w}$  перебуває між  $\mathbf{p}_2$  і  $\mathbf{p}_3$ , а вектор  ${}_1\mathbf{w}$  жодного разу не змінив свого значення (перший нейрон, який жодного разу не перемагав, є «мертвим»).

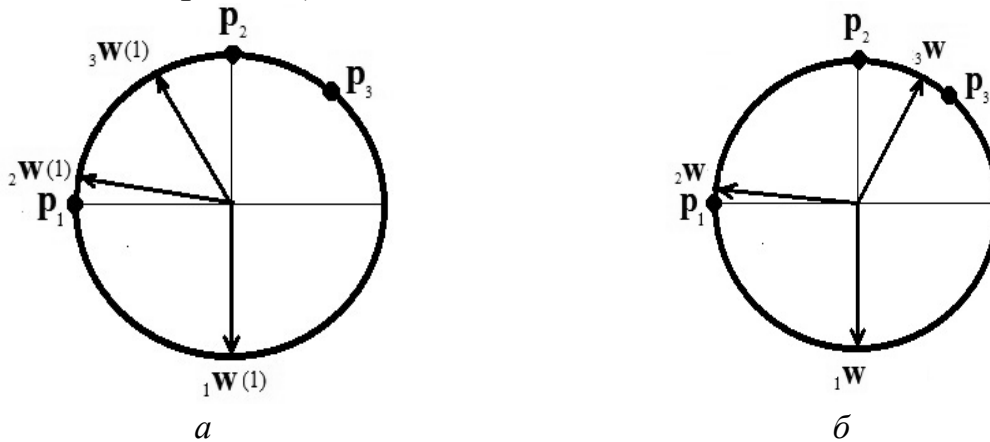


Рис. 11.20. Проміжна (а) та остаточна (б) кластеризації

**Приклад 11.6.** Нехай задано вхідні вектори та початкові значення ваги (рис. 11.21). Використовуючи правило Кохонена з коефіцієнтом навчання  $\alpha = 0,5$ , навчити конкурентну НМ кластеризувати задані вектори. Нарисувати графік розміщення вагових векторів після одноразового послідовного введення усіх вхідних векторів.

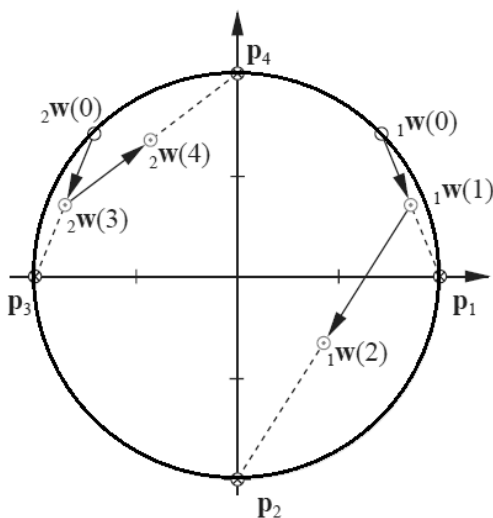


Рис. 11.21. Вхідні вектори та початкові значення ваги, розв'язання задачі

*Розв'язання.* Цю задачу можна розв'язати графічно (рис. 11.21). Маємо

$${}_1\mathbf{w}(0) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T; \quad {}_2\mathbf{w}(0) = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T;$$

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \quad \mathbf{p}_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}; \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

На початку навчання на вхід подається вектор  $\mathbf{p}_1$ . Оскільки найближчим до нього є ваговий вектор  ${}_1\mathbf{w}(0)$ , то перший нейрон перемагає, а  ${}_1\mathbf{w}(0)$  переміщується в бік  $\mathbf{p}_1$  (наближається до  $\mathbf{p}_1$  на половину відстані між  ${}_1\mathbf{w}(0)$  і  $\mathbf{p}_1$ , тому що  $\alpha = 0,5$ ). Далі на вхід подається вектор  $\mathbf{p}_2$ , знову перемагає перший нейрон і  ${}_1\mathbf{w}(1)$  наближається до  $\mathbf{p}_2$

на половину відстані між ними (значення  ${}_2\mathbf{w}(0)$  при цьому не змінюється). На третьому кроці на вхід мережі подається вектор  $\mathbf{p}_3$ , перемагає другий нейрон і вектор  ${}_2\mathbf{w}(0)$  наближається на половину відстані до  $\mathbf{p}_3$ . На четвертому кроці на вхід подається вектор  $\mathbf{p}_4$ , знову перемагає другий нейрон і ваговий вектор  ${}_2\mathbf{w}(3)$  наближається на половину відстані до  $\mathbf{p}_4$ . Якщо про-

довжити навчання, то перший нейрон буде розпізнавати вхідні вектори  $\mathbf{p}_1$  і  $\mathbf{p}_2$ , а другий – вектори  $\mathbf{p}_3$  і  $\mathbf{p}_4$ .

**Приклад 11.7.** Конкурентні шари та мапи ознак вимагають нормалізації вхідних векторів. Але що буде, якщо вхідні вектори не є нормалізованими?

*Перший спосіб* вирішення цієї проблеми полягає в нормалізації даних перед введенням у мережу (недоліком цього методу є втрата інформації про абсолютне значення величини векторів). *Другий спосіб* полягає в заміні скалярного добутку (який, зазвичай, використовують для обчислення значення входу ФА:  $\mathbf{u} = \mathbf{W}\mathbf{p}$ ) на обчислення відстані:  $u_i = -\|\mathbf{w} - \mathbf{p}\|$  і  $\mathbf{y} = \text{compet}(\mathbf{u})$ , як у мережі LVQ. При цьому зберігається інформація про абсолютне значення векторів. *Третій спосіб* полягає в додаванні елемента «1» до кожного вектора перед процедурою нормалізації. Зміна цієї константи зберігає інформацію про абсолютну величину вектора.

Нормалізувати задані вектори, використовуючи останній спосіб:  $\mathbf{p}_1 = [1 \ 1]^T$ ;  $\mathbf{p}_2 = [0 \ 1]^T$ ;  $\mathbf{p}_3 = [0 \ 0]^T$ .

*Розв'язання.* Додавши елемент «1» до кожного вектора, одержимо  $\mathbf{p}'_1 = [1 \ 1 \ 1]^T$ ;  $\mathbf{p}'_2 = [0 \ 1 \ 1]^T$ ;  $\mathbf{p}'_3 = [0 \ 0 \ 1]^T$ . Нормалізуємо кожний вектор,

$$\text{одержимо: } \mathbf{p}''_1 = \frac{[1 \ 1 \ 1]^T}{\|\mathbf{p}'_1\|} = \left[ \frac{1}{\sqrt{3}} \ \frac{1}{\sqrt{3}} \ \frac{1}{\sqrt{3}} \right]^T; \quad \mathbf{p}''_2 = \frac{[0 \ 1 \ 1]^T}{\|\mathbf{p}'_2\|} =$$

$$= \left[ 0 \ \frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \right]^T; \quad \mathbf{p}''_3 = \frac{[0 \ 0 \ 1]^T}{\|\mathbf{p}'_3\|} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

**Приклад 11.8.** Нехай задано мапу ознак (рис. 11.22), яка містить дев'ять нейронів, розміщених в одному вимірі (на прямій). Використовуючи задані початкові значення вагових векторів, зобразити нейрони на діаграмі, поєднуючи лініями вагові вектори сусідніх нейронів:

$$\mathbf{W}(0) = \begin{bmatrix} 0,41 & 0,45 & 0,41 & 0 & 0 & 0 & -0,41 & -0,45 & -0,41 \\ 0,41 & 0 & -0,41 & 0,45 & 0 & -0,45 & 0,41 & 0 & -0,41 \\ 0,82 & 0,89 & 0,82 & 0,89 & 1 & 0,89 & 0,82 & 0,89 & 0,82 \end{bmatrix}^T.$$

Провести навчання мапи ознак, подавши на вхід мережі вектор  $\mathbf{p} = [0,67 \ 0,07 \ 0,74]^T$  для коефіцієнта навчання  $\alpha = 0,1$  і радіуса околу  $r = 1$ .

*Розв'язання.* Діаграму мапи ознак для початкових значень ваги зображено на рис. 11.23. Подавши на вхід мережі вектор  $\mathbf{p}$ , одержимо:

$$\mathbf{y} = \text{compet}(\mathbf{W}(0)\mathbf{p}) = \text{compet}([0,91 \ 0,96 \ 0,85 \ 0,70 \ 0,74 \ 0,63 \ 0,36 \ 0,36 \ 0,3]^T) = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T.$$

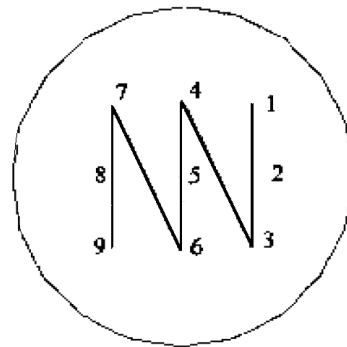


Рис. 11.22. Мапа ознак з дев'яти нейронів

Одержимо:

$$\begin{aligned} {}_1\mathbf{w}(1) &= {}_1\mathbf{w}(0) + \alpha(\mathbf{p} - {}_1\mathbf{w}(0)) = \begin{bmatrix} 0,41 \\ 0,41 \\ 0,82 \end{bmatrix} + 0,1 \cdot \left( \begin{bmatrix} 0,67 \\ 0,07 \\ 0,74 \end{bmatrix} - \begin{bmatrix} 0,41 \\ 0,41 \\ 0,82 \end{bmatrix} \right) = \begin{bmatrix} 0,43 \\ 0,37 \\ 0,81 \end{bmatrix}; \\ {}_2\mathbf{w}(1) &= {}_2\mathbf{w}(0) + \alpha(\mathbf{p} - {}_2\mathbf{w}(0)) = \begin{bmatrix} 0,45 \\ 0 \\ 0,89 \end{bmatrix} + 0,1 \cdot \left( \begin{bmatrix} 0,67 \\ 0,07 \\ 0,74 \end{bmatrix} - \begin{bmatrix} 0,45 \\ 0 \\ 0,89 \end{bmatrix} \right) = \begin{bmatrix} 0,47 \\ 0,01 \\ 0,88 \end{bmatrix}; \\ {}_3\mathbf{w}(1) &= {}_3\mathbf{w}(0) + \alpha(\mathbf{p} - {}_3\mathbf{w}(0)) = \begin{bmatrix} 0,41 \\ -0,41 \\ 0,82 \end{bmatrix} + 0,1 \cdot \left( \begin{bmatrix} 0,67 \\ 0,07 \\ 0,74 \end{bmatrix} - \begin{bmatrix} 0,41 \\ -0,41 \\ 0,82 \end{bmatrix} \right) = \begin{bmatrix} 0,43 \\ -0,36 \\ 0,81 \end{bmatrix}. \end{aligned}$$

ченнями ваги:  $\mathbf{W}^1 = \begin{bmatrix} \mathbf{w}^{1T}_1 \\ \mathbf{w}^{1T}_2 \\ \mathbf{w}^{1T}_3 \\ \mathbf{w}^{1T}_4 \\ \mathbf{w}^{1T}_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$ ;  $\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$ . Нарисувати

*Розв'язання.* Зобразимо кожний вектор  $\mathbf{w}$  матриці  $\mathbf{W}^1$  відповідно до індексу ненульових елементів у стовпцях матриці  $\mathbf{W}^2$ , який вказує на клас. Вектори-прототипи маркованих класів (область 1-го класу позначено темно-сірим кольором, 2-го – білим, 3-го – сірим) зображено на рис. 11.24, б.

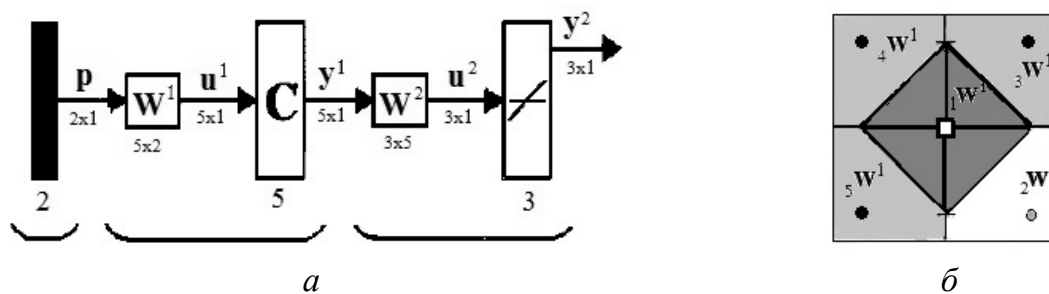


Рис. 11.24. Мережа LVQ (а) та області вхідного простору для кожного із трьох класів (б) до прикл. 11.9

**Приклад 11.10.** Спроекувати мережу LVQ для розв’язання задачі кластеризації (рис. 11.25): вектори мають належати до одного з трьох кластерів відповідно до кольору. Нарисувати області кожного кластера.

*Розв’язання.* Нехай мережа LVQ замість використання скалярного добутку обчислює безпосередньо відстань між векторами. Нехай кожному кольору відповідає певний кластер: 1-й кластер включає всі білі точки; 2-й – усі чорні; 3-й – усі сірі. Тепер можна визначити параметри мережі LVQ (рис. 11.26): оскільки задано три кластери, то вихідний шар має три нейрони; оскільки задано дев’ять підкласів, то у прихованому шарі є дев’ять нейронів.

Сформуємо вагову матрицю  $\mathbf{W}^1$  першого шару, кожний рядок якої є транспонованим вектором-прототипом для одного кластера. Вибравши вектори-прототипи, розміщені у центрі кожного кластера, одержимо:

$$\mathbf{W}^1 = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix}^T.$$

Тепер кожний нейрон 1-го шару буде реагувати на вектори окремого кластера. Далі виберемо  $\mathbf{W}^2$  таким чином, щоб кожний підклас відповідав певному кластеру: якщо  $i$ -й підклас належить класу  $k$ , то  $w_{ki}^2 = 1$ . Наприклад, точки першого підкласу на рис. 11.25, який міститься зліва зверху, мають білий колір, тому вони належать 1-му кластеру та  $w_{11}^2 = 1$ . Після виконання цієї процедури для всіх дев’яти підкласів одержимо:

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Протестуємо мережу, подавши на її вхід вектор  $\mathbf{p} = [1 \ 0]^T$ . На виході першого шару одержимо:

$$\begin{aligned} \mathbf{y}^1 &= \text{compet}(\mathbf{u}^1) = \text{compet}\left(\begin{bmatrix} -\sqrt{5} & -\sqrt{2} & -1 & -2 & -1 & 0 & -\sqrt{5} & -\sqrt{2} & -1 \end{bmatrix}^T\right) = \\ &= [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0]^T. \end{aligned}$$

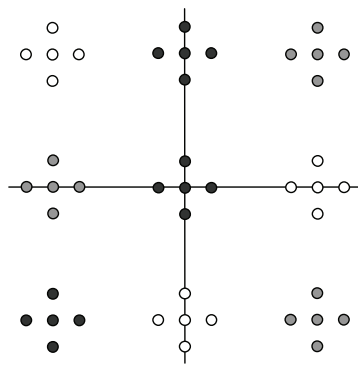


Рис. 11.25. Задача класифікації

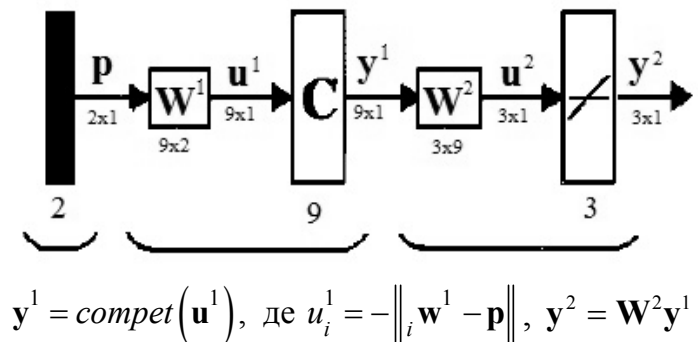


Рис. 11.26. Мережа LVQ

Отже, вектор  $p$  належить 6-му підкласу.

Наведемо *m*-файл для обчислення вектора  $u^1$ :

```

W1=[-1 0 1 -1 0 1 -1 0 1; 1 1 1 0 0 0 -1 -1 -1];
W1=W1'; p=[1 0]; n=zeros(9,2)
for i=1:1:9
    n(i,1)=W1(i,1)-p(1); n(i,2)=W1(i,2)-p(2);
end;
u=zeros(9,1), for i=1:1:9
    u(i)=abs(sum(n(i,:).^2)),
    if u(i)~=0
        u(i)=-u(i)/sqrt(u(i))
    end; end; u1=u

```

На виході 2-го шару одержимо:

$$y^2 = W^2 y^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Отже, вектор  $p$  належить 1-му кластеру, що відповідає дійсності. Області класів і границі їх поділу зображено на рис. 11.27.

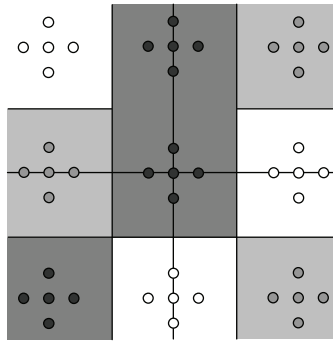


Рис. 11.27. Области вхідного простору для кожного класу і границі поділу

### Контрольні запитання

1. Опишіть нейронну мережу Хеммінга: архітектуру, принцип функціонування, сутність латерального гальмування.
2. Опишіть конкурентну одношарову мережу. В чому полягає сутність проблеми наявності «мертвих» нейронів і які існують шляхи її розв'язання?
3. Опишіть самоорганізовані мережі Кохонена (шар і мапу Кохонена), їх принципи функціонування.
4. Опишіть мережу квантування векторів (LVQ): архітектуру, принцип функціонування.

### Задачі для самостійного розв'язання

**Задача 11.1.** Нехай вагова матриця другого шару мережі Хеммінга

має вигляд  $\mathbf{W}^2 = \begin{bmatrix} 1 & -\frac{3}{4} & -\frac{3}{4} \\ -\frac{3}{4} & 1 & -\frac{3}{4} \\ -\frac{3}{4} & -\frac{3}{4} & 1 \end{bmatrix}$ . Ця матриця порушує умову для 2-го шару, оскільки  $\varepsilon = \frac{3}{4} > \frac{1}{S-1} = \frac{1}{2}$ .

Навести приклад виходу 1-го шару, для якого 2-й шар не зможе працювати.

**Задача 11.2.** Нехай задано вхідні вектори та початкові значення ваги (рис. 11.28).

**I.** Накреслити діаграму конкурентного шару, який би класифікував задані вектори (при цьому три кластери векторів утворюють окремі класи).

**II.** Графічно навчити мережу, подавши на її вхід вектори в такій

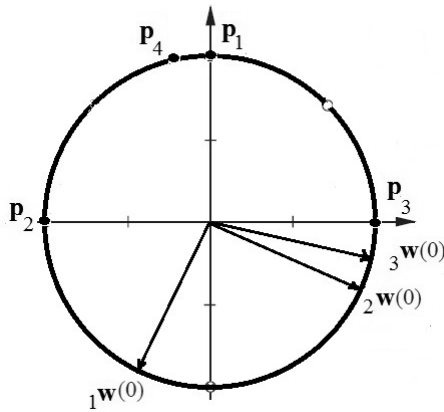


Рис. 11.28. Вектори даних та початкові значення ваги

послідовності:  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ . У разі однакових значень на виході декількох нейронів конкурентна ФА обирає нейрон з найменшим індексом. Накреслити результуючі вагові вектори та границі поділу областей, які відповідають кластерам.

**Задача 11.3.** Навчити конкурентну мережу, використовуючи такі вхідні вектори:  $\mathbf{p}_1 = [1 \quad -1]^T$ ;  $\mathbf{p}_2 = [1 \quad 1]^T$ ;  $\mathbf{p}_3 = [-1 \quad -1]^T$ .

**I.** Використати правило Кохонена з коефіцієнтом навчання  $\alpha = 0,5$  і виконати процес навчання за один прохід, подавши на вхід мережі вектори у такій послідовності:  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ . Зобразити одержаний результат графічно. Взяти початкове значення вагової матриці рівним

$$\mathbf{W}(0) = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}.$$

**II.** Визначити, як були кластеризовані вектори після одного проходу, чи зміниться кластеризація, якщо змінити послідовність введення векторів.

**III.** Повторити п. I за  $\alpha = 0,25$ . Порівняти одержані результати.

**Задача 11.4.** Поняття «совість» стосується проблеми мертвих нейронів у конкурентних шарах і мережах LVQ. Нейронам, які перебувають дуже далеко від вхідних векторів, надається можливість стати переможцями після застосування адаптивних зсувів (які стають більш від'ємними з кожною перемогою нейрона). У результаті нейрони, які часто перемагають, починають відчувати «докори совісті» доти, поки інші нейрони не матимуть можливості стати переможцями. Конкурентну мережу зі зсувом зображено на рис. 11.29.

**I.** Розглянути вектори, зображені на рис. 11.30. Визначити, чи існує порядок зображення векторів, за якого  ${}_1\mathbf{w}$  стане переможцем і переміститься в бік інших векторів. При цьому адаптивний зсув не використовується.

**II.** Задано вхідні вектори  $\mathbf{p}_1 = [-1 \quad 0]^T$ ;  $\mathbf{p}_2 = [0 \quad 1]^T$ ;  $\mathbf{p}_3 = \begin{bmatrix} 1 \\ \frac{1}{\sqrt{2}} \end{bmatrix}^T$ ,

початкові значення ваги  ${}_1\mathbf{w}(0) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ ;  ${}_2\mathbf{w}(0) = \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ 1 \\ -\frac{1}{\sqrt{5}} \end{bmatrix}$ ;  ${}_3\mathbf{w}(0) = \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ 2 \\ -\frac{1}{\sqrt{5}} \end{bmatrix}$ .



та зсув  $\mathbf{b}(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ . Визначити значення ваги, використавши правило на-

вчання Кохонена для зсуву  $i$ -го нейрона:  $b_i^{\text{нов}} = \begin{cases} 0,9b_i^{\text{стар}}, & \text{якщо } i \neq i^*; \\ b_i^{\text{стар}} - 0,2, & \text{якщо } i = i^*. \end{cases}$

Повторити введення послідовності вхідних векторів  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ , поки перший нейрон не стане переможцем.

**III.** Визначити, скільки разів цю послідовність векторів необхідно подавати на вхід для перемоги першого нейрона.

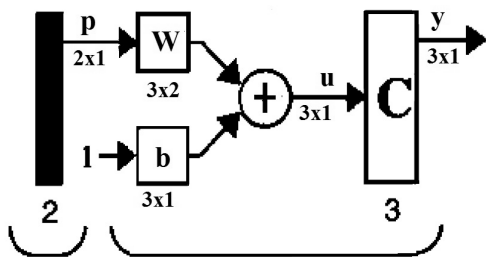


Рис. 11.29. Конкурентна мережа зі зсувом

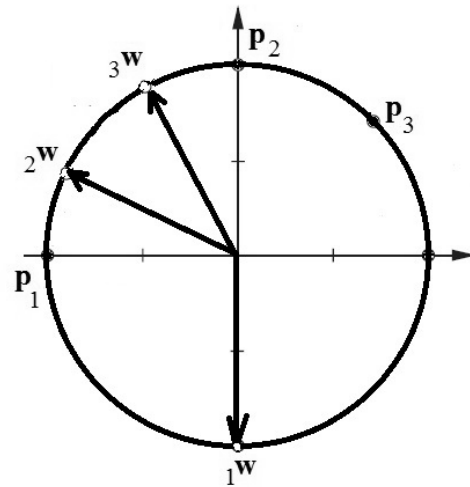


Рис. 11.30. Вхідні вектори та мертвий нейрон

**Задача 11.5.** У мережі LVQ замість використання скалярного добутку безпосередньо обчислюється відстань між вхідним і ваговим векторами. У результаті мережа LVQ (рис. 11.31) не потребує нормалізації вхідних векторів. Цей метод можна використати для класифікації ненормалізованих векторів за допомогою конкурентного шару.

**I.** Використати вказаний метод для навчання двонейронного конкурентного шару на такій послідовності ненормалізованих векторів:

$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ ;  $\mathbf{p}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ , якщо коефіцієнт  $\alpha = 0,5$ . Розглянути такі початкові значення ваги мережі:  ${}_1\mathbf{w}(0) = [0 \ 1]^T$ ;  ${}_2\mathbf{w}(0) = [1 \ 0]^T$ . На вхід подавати вектори у такій послідовності:  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_1$ .

**II.** Показати, що для нормалізованих вхідних векторів змінена конкурентна мережа (рис. 11.31), яка обчислює відстань, на виході має той самий результат, що й звичайна конкурентна мережа, яка використовує скалярний добуток.

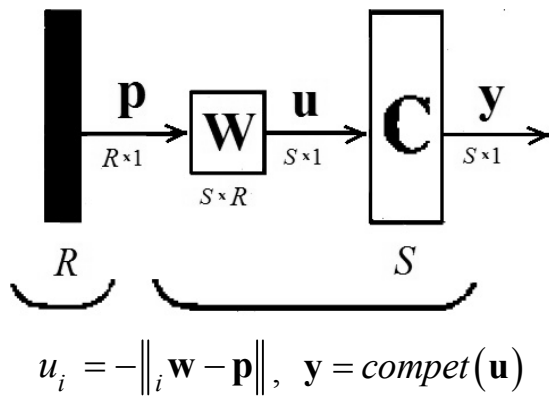


Рис. 11.31. Конкурентна мережа до задачі 11.5

**Задача 11.6.** Нехай задано мережу LVQ із такими значеннями ваги:

$$\mathbf{W}^1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix};$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

**I.** Визначити, скільки класів і підкласів містить ця мережа.

**II.** Накреслити діаграму, яка зображує вагові вектори першого шару, та межі, що поділяють вхідний простір на підкласи. Визначити належність кожного підкласу до класу.

**Задача 11.7.** Спроектувати мережу LVQ, яка класифікує вектори таким чином: 1-й клас –  $\left\{ \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}$ ; 2-й клас –  $\left\{ \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \right\}$ ;  
3-й клас –  $\left\{ \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \right\}$ .

**I.** Визначити, скільки нейронів у кожному шарі повинна мати мережа LVQ.

**II.** Визначити вагу 1 та 2-го шарів.

**III.** Перевірити роботу мережі, використавши принаймні один вектор з кожного класу.

**Задача 11.8.** Спроектувати мережу LVQ, яка класифікує вектори таким чином: 1-й клас –  $\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \right\}$ ; 2-й клас –  $\left\{ \mathbf{p}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$ .

**I.** Визначити, чи можна цю задачу розв'язати за допомогою персептрона.

**II.** Встановити, скільки нейронів повинен мати кожний шар мережі LVQ, враховуючи, що кожний клас містить два опуклі підкласи.

**III.** Визначити вагу другого шару мережі LVQ.

**IV.** Взяти початкові значення ваги першого шару мережі рівними нулю, обчислити їхні нові значення, застосувавши правило Кохонена, якщо  $\alpha = 0,5$ , послідовність навчання має вигляд  $\mathbf{p}_4, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_1, \mathbf{p}_2$ .

## Розділ 12. МЕРЕЖА ГРОССБЕРГА

### 12.1. Система зору людини

Наприкінці 1960-х рр. кількість дослідників НМ помітно зменшилася, проте деякі з них продовжували роботу (Т. Кохонен, Дж. Андерсон та інші). Найкращі результати отримав Стівен Гроссберг, який, зокрема, досліджував певні функції інтелекту та мозку людини, використовуючи нелінійну математику. Дослідження Ст. Гроссберга відзначаються високим рівнем математичної й нейрофізіологічної складності на суміжжі математики, психології та нейрофізіології. В його роботах знайшли відображення ідеї міждисциплінарних досліджень функцій мозку Гельмгольца, Максвелла та Маха. Мережа Гроссберга – це самоорганізована рекурентна конкурентна мережа, яка становить основу адаптивних мереж з використанням теорії резонансу. Розглянемо одну з мереж, спроектовану Гроссбергом. Для цього спочатку ознайомимося із системою зору людини, потім зі стандартним блоком у вигляді шунтувальної моделі як складової багатьох мереж Гроссберга та проілюструємо, як цю мережу можна використовувати для адаптивного розпізнавання образів.

Схему перших етапів функціонування системи зору зображено на рис. 12.1.

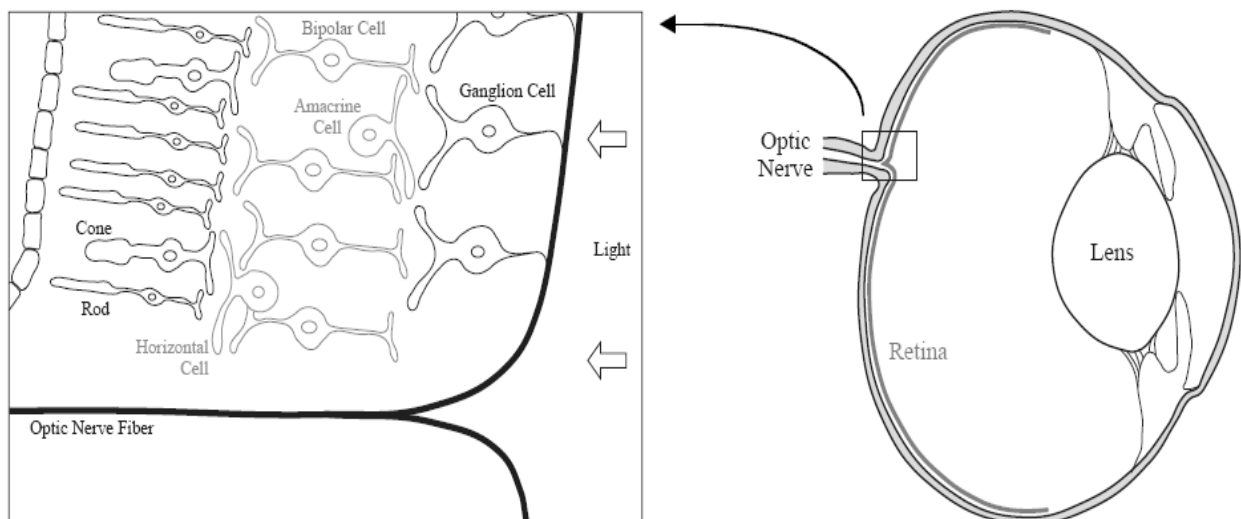


Рис. 12.1. Око людини

Світло проходить через рогівку (прозору передню частину ока) та лінзу (*lens*), яка переломлює його, щоб сфокусувати інформацію про об'єкти на сітківці (внутрішньому шарі зовнішньої стінки ока – *retina*). Після потрапляння світла на сітківку розпочинається перетворення цієї інформації на зрозуміле зображення.

Сітківка – це частина мозку, яка протягом ембріонального розвитку відокремилася від нього, але залишається пов'язаною з ним через оптичний нерв (*optic nerve*). Сітківка складається з таких трьох шарів нервових клітин:

1. *Зовнішній шар* – це набір фоторецепторів: палички (*rods*) та колбочки (*cones*), які перетворюють світло в електричні сигнали. Палички дозволяють людині бачити в темряві, тоді як колбочки – бачити дрібні деталі й кольори. Світло має пройти через інші два шари сітківки, щоб стимулювати палички та колбочки.

2. *Середній шар* сітківки складається з трьох типів клітин: біполярних (*bipolar cells*), горизонтальних (*horizontal cells*) та амакрінових (*amacrine cells* – клітин, позбавлених аксона). Біполярні клітини отримують інформацію від рецепторів і передають її в третій шар сітківки з нервовими вузлами. Горизонтальні клітини пов'язують рецептори з біполярними клітинами, а амакрінові – біполярні клітини з нейронами.

3. *Заключний шар* сітківки утворюють нейрони (*ganglion cell* – нервовий вузол), аксони яких проходять через поверхню сітківки та збираються в пучок, формуючи оптичний нерв. Кожне око містить приблизно 125 млн рецепторів і тільки 1 млн нейронів.

Аксони нейронів, пов'язаних з оптичним нервом, з'єднуються з ділянкою мозку, який називають *латеральним колінчастим центром* (рис. 12.2). Від *латерального колінчастого центру* волокна розгалужуються в первинну зорову зону кори мозку, розміщену позаду нього.

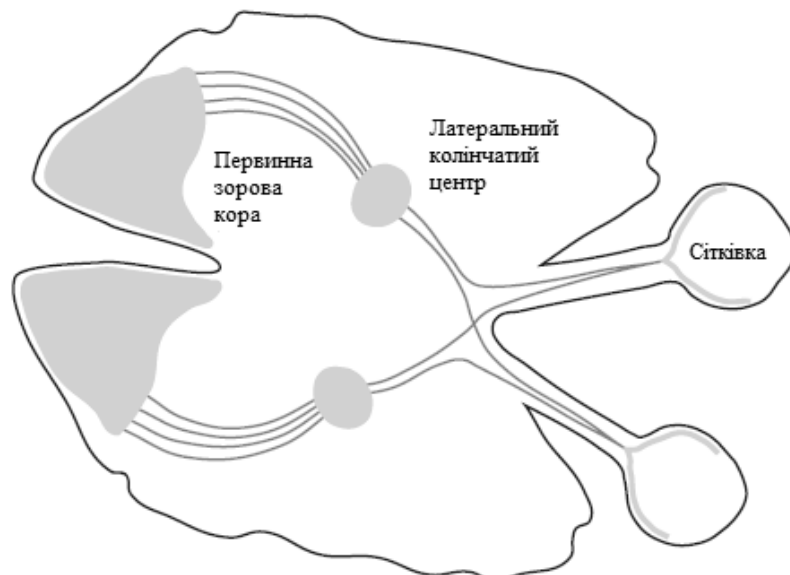


Рис. 12.2. Шлях сигналів системи зору

Аксони нейронів утворюють синапси з латеральними колінчастими клітинами, а аксони латеральних колінчастих клітин утворюють синапси

з клітинами зорової кори – ділянки мозку, яка відповідає за зорову функцію та складається з багатьох шарів клітин. Кожна півкуля мозку отримує інформацію від обох очей (рис. 12.2): при цьому ліва половина волокон кожної зорової ділянки закінчується у правій половині мозку, а права половина – у лівій.

**Нормалізація зору.** Серед процесів, які відбуваються на початковій стадії функціонування системи зору, можна виділити постійність і контрастність яскравості. Ефект постійності яскравості зображено на рис. 12.3, *а* (невелике сіре коло всередині темнішого освітленого сірого кола). Людина визначає яскравість центрального кола, дивлячись на послідовність сірих кіл, освітлених окремо, й обирає коло однакової яскравості. Далі світло, що освітлює сіре та темно-сіре кола, стає більш яскравим, і знову людина обирає коло однакової яскравості. Цей процес повторюється для декількох рівнів освітлення. Виявлено, що в кожному випадку людина вибере одне коло, яке відповідає оригіналу у вигляді центрального кола.

Інший феномен системи зору, який стосується постійності яскравості – це контрастність яскравості, проілюстрована на рис. 12.3, *б*, де в центрі обох фігур перебувають два однакових невеликих сірих кола. Коло зліва оточене темно-сірим кільцем, а коло справа – світло-сірим кільцем. Незважаючи на те, що обидва центральних кола мають однакий сірий колір, коло на темному фоні здається більш яскравим. Це можна пояснити тим, що система зору людини чутлива до відносної інтенсивності яскравості.

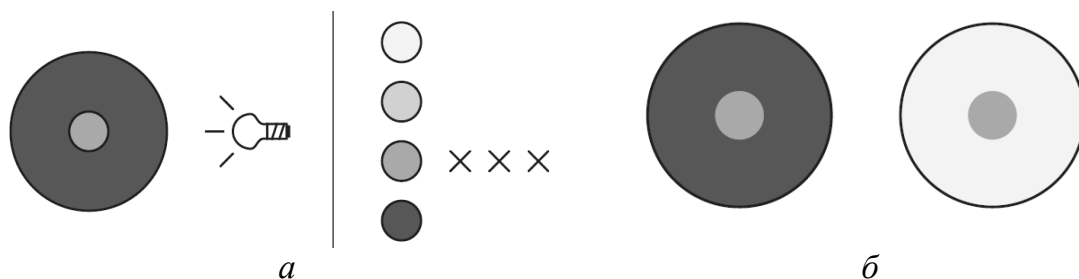


Рис. 12.3. Тестування яскравості на постійність (*а*) та контрастність (*б*)

Постійність та контрастність яскравості дуже важливі для системи зору, оскільки без них людина ніколи б не навчилася розпізнавати речі. Гроссберг назвав цей процес нормалізацією зору. Розглянемо архітектуру мережі, яка моделює систему зору.

## 12.2. Фундаментальна нелінійна шунтувальна модель

Розглянемо основні складові мережі Гроссберга [14].

Інтегратор (рис. 12.4, а) – блок, який входить до складу квазіінтегратора (рис. 12.4, б). Вихід інтегратора  $y(t)$  обчислюють за формулою  $y(t) = \int_0^t u(\tau) d\tau + y(0)$ , де  $u(t)$  – вхід інтегратора;  $y(0)$  – початкова умова.

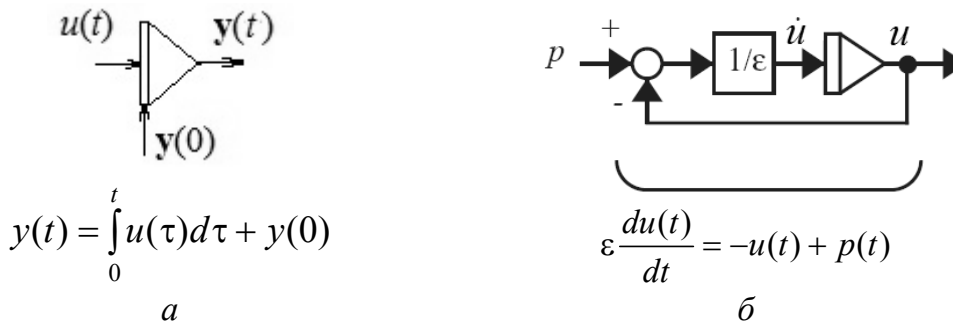


Рис. 12.4. Складові мережі Гроссберга: інтегратор (а), квазіінтегратор (б)

Поведінка квазіінтегратора описується рівнянням

$$\varepsilon \frac{du(t)}{dt} = -u(t) + p(t), \quad (12.1)$$

де  $\varepsilon$  – константа часу.

Вихід квазіінтегратора за вхідного сигналу  $p(t)$  має вигляд  $u(t) = e^{-\frac{t}{\varepsilon}} u(0) + \frac{1}{\varepsilon} \int_0^t e^{-\frac{(t-\tau)}{\varepsilon}} p(\tau) d\tau$ , оскільки відомо, що лінійне неоднорідне диференціальне рівняння

$\frac{dx(t)}{dt} = a(t)x(t) + b(t)$  має загальний

розв'язок вигляду  $x(t) = ce^{\int_0^t a(s) ds} + \int_0^t e^{\int_0^t a(s) ds} b(\tau) d\tau$ , де  $c \in \mathbb{R}$ . Наприклад,

якщо вхідний сигнал  $p(t) = p$  є константою, а початкова умова  $u(0) = 0$ , то маємо

$$u(t) = p \left( 1 - e^{-\frac{t}{\varepsilon}} \right). \quad (12.2)$$

За формулою (12.2) за  $p = 1$  та  $\varepsilon = 1$  вихід  $u(t)$  експоненційно наближається до стійкого стану, який дорівнює одиниці. Швидкість появи виходу  $u(t)$  квазіінтегратора залежить від константи часу  $\varepsilon$ : якщо  $\varepsilon$  зменшується, вихід з'являється швидше, а якщо  $\varepsilon$  збільшується – повільніше. Квазіінтегратор формує ядро шунтувальної моделі Гроссберга (рис. 12.5),

функціонування якої описується рівнянням

$$\varepsilon \frac{du(t)}{dt} = -u(t) + [(b^+ - u(t))p^+] + [-(u(t) + b^-)p^-], \quad (12.3)$$

де  $p^+$  – невід’ємна величина, що зображує збудливий вхідний сигнал (сигнал, який зумовлює збільшення виходу);  $p^-$  – невід’ємна величина, що зображує пригнічувальний вхідний сигнал (сигнал, який зумовлює зменшення виходу); зсуви  $b^+$  і  $-b^-$  позначають інтервал  $[-b^-, b^+]$  (верхню та нижню границі зміни виходу квазіінтегратора  $u(t)$ ). У правій частині рівняння (12.3) є три доданки.

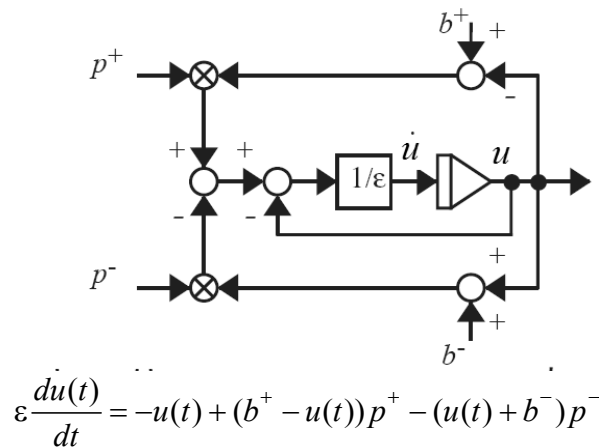


Рис. 12.5. Шунтувальна модель Гроссберга

Розглянемо кожний доданок окремо:  $[-u(t)]$  – це лінійний елемент загасання, який є від’ємним, якщо  $u(t)$  – додатний, і додатним, якщо  $u(t)$  – від’ємний;  $[(b^+ - u(t))p^+]$  – це регулятор нелінійного посилення, який є додатним, якщо  $u(t) < b^+$ , і дорівнює нулю, якщо  $u(t) = b^+$  (це дозволяє встановити значення верхньої межі  $u(t)$  на рівні  $b^+$ );  $[-(u(t) + b^-)p^-]$  – це регулятор нелінійного загасання, який дозволяє встановити значення нижньої межі  $u(t)$  на рівні  $-b^-$ . Функціонування шунтувальної мережі Гроссберга за  $b^+ = 1$ ;  $b^- = 0$ ;  $\varepsilon = 1$ ;  $p^- = 0$  зображено на рис. 12.6: на лівому графіку – відповідь мережі за  $p^+ = 1$ ; на правому – відповідь мережі за  $p^+ = 5$ . Незважаючи на те, що збуджувальний вхідний сигнал має значення  $p^+ = 5$ , вихід мережі збільшується менш ніж у два рази.

Якщо продовжити збільшувати  $p^+$ , то вихід мережі  $u(t)$  також буде збільшуватися, але його значення не буде перевищувати  $b^+ = 1$ . Якщо подати на вхід шунтувальної мережі пригнічувальний вхідний сигнал  $p^- > 0$ , то вихід мережі зменшиться, але залишиться більшим за значення  $-b^-$ . Отже, якщо  $u(0)$  належить  $[-b^-, b^+]$ , то  $u(t)$  також буде належати  $[-b^-, b^+]$ . Шунтувальна модель є основою конкурентної мережі Гроссберга.

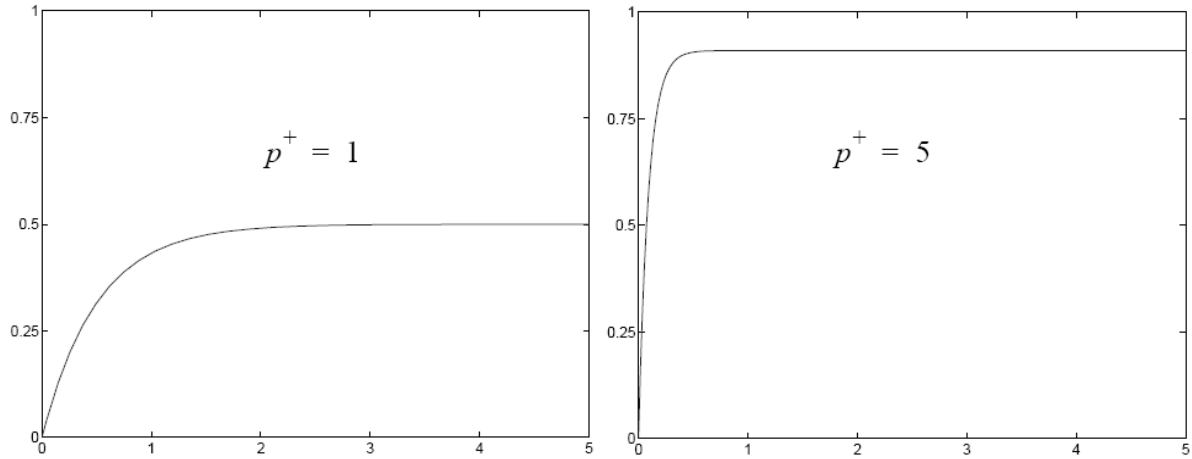


Рис. 12.6. Вихід шунтувальної мережі Гроссберга за  $b^- = 0$ ;  $b^+ = 1$ ;  $\varepsilon = 1$ ;  $p^- = 0$

### 12.3. Двошарова конкурентна мережа Гроссберга

Розглянемо конкурентну мережу Гроссберга [14], що є результатом досліджень у сфері моделювання системи зору ссавців. На жаль, ця мережа повністю не моделює системи зору людини, але вона ілюструє деякі її властивості (наприклад, механізми короткострокової STM і довгострокової LTM пам'яті), виконує адаптацію, фільтрацію, нормалізацію та посилення контрастності. Мережа Гроссберга схожа на конкурентну мережу Кохонена та складається з двох шарів: перший шар – наближена модель сітківки ока, другий шар – аналог частини кори головного мозку, яка бере участь у функціонуванні системи зору. Розглянемо кожний шар окремо.

**Перший шар** мережі Гроссберга (рис. 12.7) отримує зовнішні входні дані та нормалізує інтенсивність входного образу (нагадаємо, що мережа Кохонена працює найкраще, якщо входні образи нормалізовано).

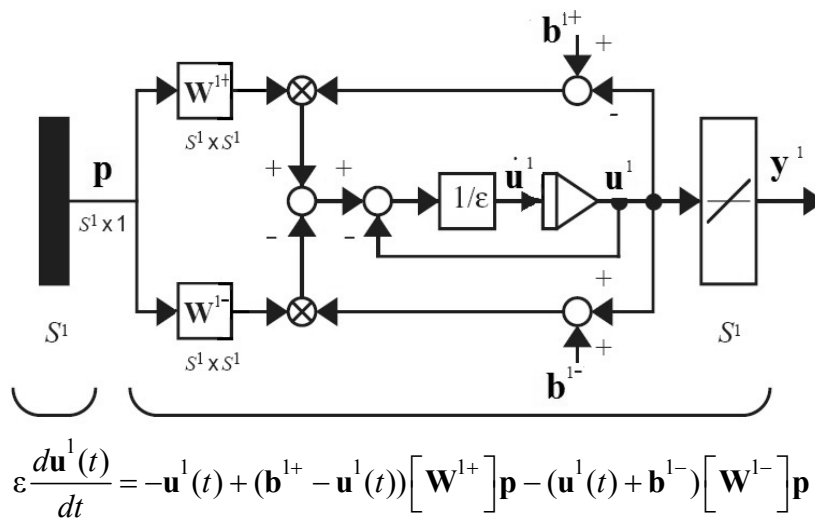


Рис. 12.7. Перший шар мережі Гроссберга



Функціонування першого шару описують рівнянням

$$\varepsilon \frac{d\mathbf{u}^1(t)}{dt} = -\mathbf{u}^1(t) + (\mathbf{b}^{1+} - \mathbf{u}^1(t))[\mathbf{W}^{1+}] \mathbf{p} - (\mathbf{u}^1(t) + \mathbf{b}^{1-})[\mathbf{W}^{1-}] \mathbf{p}. \quad (12.4)$$

Рівняння (12.4) – це шунтувальна модель зі збуджуванням входом  $p^+ = [\mathbf{W}^{1+}]p$  та пригнічуванням входом  $p^- = [\mathbf{W}^{1-}]p$  першого шару, де

$$\mathbf{W}^{1+} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}; \quad \mathbf{W}^{1-} = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 0 \end{bmatrix}. \text{ Таким чином, вхід } i\text{-го збу-}$$

джуваного нейрона – це  $i$ -й елемент вхідного вектора, а вхід  $i$ -го пригнічуваного нейрона – це сума всіх елементів вхідного вектора, крім  $i$ -го його елемента. Схему зв'язків, визначену ваговими матрицями  $\mathbf{W}^{1+}$  і  $\mathbf{W}^{1-}$ , називають «до центру/від оточення». Цей тип зв'язку виконує нормалізацію вхідного образу. Для спрощення припустимо, що всі елементи збуджуваного зсуву  $\mathbf{b}^{1+}$  мають однакове значення, тобто  $b_i^{1+} = b^{1+}$ , а для пригнічуваного –  $b_i^{1-} = 0$ ,  $i = 1, \dots, S^1$ . Щоб дослідити ефект нормалізації

у першому шарі, розглянемо вихід  $i$ -го нейрона:  $\varepsilon \frac{du_i^1(t)}{dt} = -u_i^1(t) + (b^{1+} - u_i^1(t))p_i - u_i^1(t) \sum_{j \neq i} p_j$ . У стійкому стані, коли  $\frac{du_i^1(t)}{dt} = 0$ , маємо

$$0 = -u_i^1(t) + (b^{1+} - u_i^1(t))p_i - u_i^1(t) \sum_{j \neq i} p_j, \text{ звідки } u_i^1 = \frac{b^{1+} p_i}{1 + \sum_{j=1}^{S^1} p_j}. \text{ Відносна ін-}$$

тенсивність  $i$ -го входу  $\bar{p}_i = \frac{p_i}{P}$ , де  $P = \sum_{j=1}^{S^1} p_j$ , тому функціонування  $i$ -го нейрона у стійкому стані можна описати як

$$u_i^1 = \left( \frac{b^{1+} P}{1 + P} \right) \bar{p}_i. \quad (12.5)$$

Отже, значення  $u_i^1$  є пропорційним до відносної інтенсивності  $\bar{p}_i$  і не залежить від значення  $P$ . Крім того, загальне функціонування нейрона обмежене:  $\sum_{j=1}^{S^1} u_j^1 = \sum_{j=1}^{S^1} \left( \frac{b^{1+} P}{1 + P} \right) \bar{p}_j = \left( \frac{b^{1+} P}{1 + P} \right) \leq b^{1+}$ . Зазначимо, що перший шар мережі Гроссберга пояснює постійність яскравості та контрастності системи зору людини. Мережа Гроссберга чутлива до відносної (а не до абсолютної) інтенсивності образу.

**Приклад 12.1.** Розглянемо перший шар мережі Гроссберга, який складається з двох нейронів, якщо  $b^{1+} = 1$  і  $\varepsilon = 0,1$ :

$$0,1 \frac{du_1^1(t)}{dt} = -u_1^1(t) + (1 - u_1^1(t))p_1 - u_1^1(t)p_2;$$

$$0,1 \frac{du_2^1(t)}{dt} = -u_2^1(t) + (1 - u_2^1(t))p_2 - u_2^1(t)p_1.$$

Графіки виходів нейронів  $u_1^1(t)$  та  $u_2^1(t)$  першого шару мережі для заданих вхідних векторів зображено на рис. 12.8:  $\mathbf{p}_1 = \begin{bmatrix} 2 \\ 8 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} 10 \\ 40 \end{bmatrix}$ , тобто інтенсивність яскравості вхідного образу  $\mathbf{p}_2$  у п'ять разів більша за  $\mathbf{p}_1$ . Загальний вихід шару мережі ( $u_1^1(t) + u_2^1(t)$ ) завжди є меншим за одиницю (рис. 12.8).

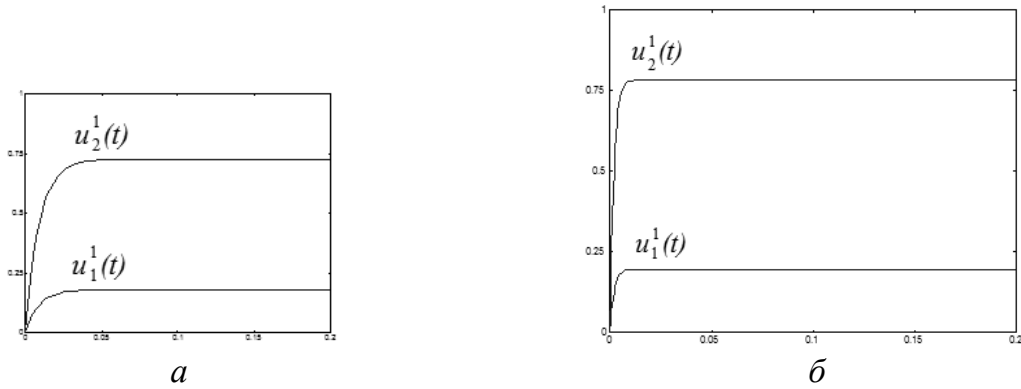


Рис. 12.8. Виходи нейронів першого шару для вхідних векторів:

$a$  — для вектора  $\mathbf{p}_1$ ;  $b$  — для вектора  $\mathbf{p}_2$

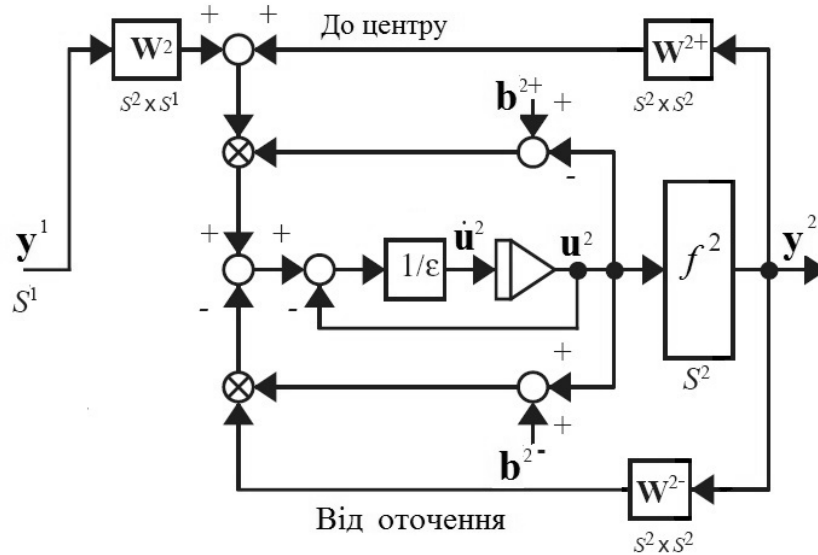
**Другий шар** мережі Гроссберга (рис. 12.9) покращує контрастність образу: нейрон, який отримав найбільші вхідні дані, впливає на вихід (подібно до конкуренції «переможець отримує все» у мережах Хеммінга та Кохонена).

Другий шар, як і перший, оснований на шунтувальній моделі, але другий шар використовує зворотний зв'язок, який викликає конкуренцію, що спричиняє поліпшення контрастності образу. Функціонування другого шару описують рівнянням

$$\varepsilon \frac{d\mathbf{u}^2(t)}{dt} = -\mathbf{u}^2(t) + (\mathbf{b}^{2+} - \mathbf{u}^2(t)) \cdot \left( [\mathbf{W}^{2+}] f^2(\mathbf{u}^2(t)) + [\mathbf{W}^2] \mathbf{y}^1 \right) - (\mathbf{u}^2(t) + \mathbf{b}^{2-}) [\mathbf{W}^{2-}] f^2(\mathbf{u}^2(t)), \quad (12.6)$$

яке відповідає шунтувальній моделі зі збуджуванням входом  $p^+ = [\mathbf{W}^{2+}] f^2(\mathbf{u}^2(t)) + [\mathbf{W}^2] \mathbf{y}^1$  і пригнічуванням входом  $p^- = [\mathbf{W}^{2-}] f^2(\mathbf{u}^2(t))$ , де  $\mathbf{W}^{2+} = \mathbf{W}^{1+}$  забезпечується зворотним зв'язком «до центру», а  $\mathbf{W}^{2-} = \mathbf{W}^{1-}$

забезпечується зворотним зв'язком «від оточення». Матриця  $\mathbf{W}^2$  складається з адаптивних вагових коефіцієнтів, аналогічних вазі у мережі Кохонена. Рядки вагової матриці  $\mathbf{W}^2$  після навчання повинні зображувати прототипи образів. Ступінь поліпшення контрастності визначається функцією активації  $f^2$ .



$$\epsilon \frac{du^2(t)}{dt} = -u^2(t) + (b^{2+} - u^2(t)) \cdot \left( [\mathbf{W}^{2+}] f^2(u^2(t)) + [\mathbf{W}^2] y^1 \right) - (u^2(t) + b^{2-}) [\mathbf{W}^{2-}] f^2(u^2(t))$$

Рис. 12.9. Другий шар мережі Гроссберга

**Приклад 12.2.** Розглянемо функціонування другого шару, в якому є два нейрони і такі параметри:  $\epsilon = 0,1$ ;  $\mathbf{b}^{2+} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{b}^{2-} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ;  $\mathbf{W}^2 = \begin{bmatrix} ({}_1\mathbf{w}^2)^T \\ ({}_2\mathbf{w}^2)^T \end{bmatrix} = \begin{bmatrix} 0,9 & 0,45 \\ 0,45 & 0,9 \end{bmatrix}$ ,  $f^2(u) = \frac{10u^2}{1+u^2}$ . Тоді рівняння функціонування другого шару

$$0,1 \frac{du_1^2(t)}{dt} = -u_1^2(t) + (1 - u_1^2(t)) \left( f^2(u_1^2(t)) + ({}_1\mathbf{w}^2)^T y^1 \right) - u_1^2(t) f^2(u_2^2(t));$$

$$0,1 \frac{du_2^2(t)}{dt} = -u_2^2(t) + (1 - u_2^2(t)) \left( f^2(u_2^2(t)) + ({}_2\mathbf{w}^2)^T y^1 \right) - u_2^2(t) f^2(u_1^2(t)).$$

Визначимо взаємозв'язок між цими рівняннями і мережами Хеммінга та Кохонена. Вхідними даними другого шару є скалярний добуток прототипів образу (стовпці вагової матриці  $\mathbf{W}^2$ ) на вихідні дані першого шару (нормалізований вхідний образ). Скалярний добуток з найбільшим значенням буде відповідати прототипу образу, найбільш близькому до вхідного образу. Другий шар згодом викликає конкуренцію для підвищення контра-

сту між нейронами, метою якої є поліпшити контрастність вихідного образу – підтримуючи великі виходи і послаблюючи малі. Це покращення контрасту, зазвичай, більш м'яке, ніж конкуренція «переможець отримує все» мереж Хеммінга і Кохонена, в яких конкуренція приводить до нульових значень усіх виходів нейронів, крім одного нейрона з найбільшим значенням входу. В мережі Гроссберга конкуренція підтримує великі величини та послаблює невеликі, перетворюючи їхні значення на нулі.

Вихід другого шару за умови, що вхідний вектор  $y^1 = [0,2 \ 0,8]^T$  (сталий режим, отриманий у прикл. 12.1) подається на вхід мережі протягом 0,25 с, а потім видаляється, зображено на рис. 12.10. Перед видаленням вхідних даних відбувається незначне покращення контрастності. Вхідні дані другого шару мають такий вигляд:

– вхід 1-го нейрона:  $({}_1w^2)^T y^1 = [0,9 \ 0,45] \cdot \begin{bmatrix} 0,2 \\ 0,8 \end{bmatrix} = 0,54$  – кореляція між входом і 1-м прототипом;

– вхід 2-го нейрона:  $({}_2w^2)^T y^1 = [0,45 \ 0,9] \cdot \begin{bmatrix} 0,2 \\ 0,8 \end{bmatrix} = 0,81$  – кореляція між входом і 2-м прототипом.

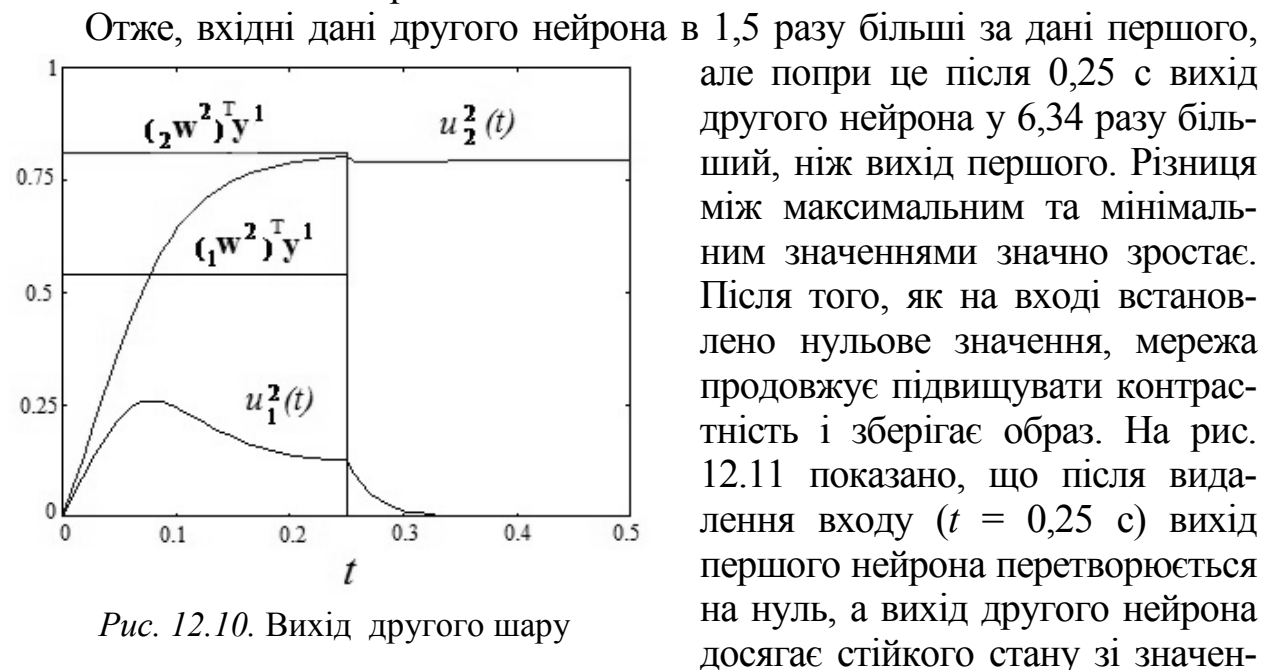



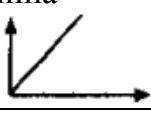
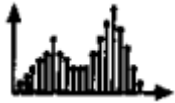
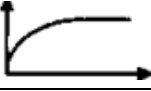



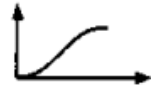
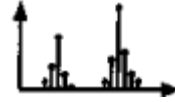
Рис. 12.10. Вихід другого шару

Вибір функції активації. Поведінка другого шару мережі Гроссберга залежить від функції активації  $f^2(u)$ . Припустимо, що вхідні дані подава-

лися протягом деякого періоду часу, і значення виходу встановилося рівним образу  (де кожна точка зображує вихід окремого нейрона). З даних табл. 12.1 видно, як вибір  $f^2(u)$  впливає на сталий стан мережі після видалення входу.

Таблиця 12.1

Вплив вибору функції активації  $f^2(u)$  на вихід мережі Гроссберга

$f^2(u)$	Вихідний браз мережі $u^2(\infty)$	Характерні риси
Лінійна 		Відмінне зберігання будь-якого образу, посилення шуму
Повільніша за лінійну 		Посилення шуму, зменшення контрастності
Швидша за лінійну 		Переможець отримує все, пригнічення шуму, дискретизація виходу
S-подібна 		Пригнічення шуму, збільшення контрастності, відсутність дискретизації

Отже, маємо такі випадки:

1. Якщо ФА є лінійною, то образ добре зберігається, при цьому шум образу збільшується та зберігається (як і значущі вхідні дані).

2. Якщо ФА повільніша за лінійну (наприклад,  $f^2(u) = 1 - e^{-u}$ ), то стійкий стан виходу не залежить від початкових умов, при цьому контрастність зменшується, а шум збільшується.

3. Якщо ФА швидша за лінійну (наприклад,  $f^2(u) = u^2$ ), це спричиняє конкуренцію «переможець отримує все». При цьому зберігаються тільки нейрони з найбільшими початковими даними, а всі інші дорівнюють нулю. Це мінімізує ефект шуму, але вихід є дискретним («все або нічого», як у мережах Хеммінга та Кохонена).

4. Якщо ФА S-подібна та швидша за лінійну для невеликих значень сигналів, приблизно лінійна для середніх і повільніша за лінійну функцію для великих значень, то контрастність образу збільшується (великі значення посилюються, а малі — послаблюються). Усі виходи нейронів другого шару, які менші за визначений поріг, дорівнюють нулю. При цьому здатність ФА пригнічувати шум поєднується зі здатністю лінійної ФА добре зберігати дані.

**Правило навчання.** Оскільки стовпці матриці адаптивної ваги  $\mathbf{W}^2$  зображують образи, які мережа повинна розпізнати, Гроссберг назвав ці адаптивні вагові коефіцієнти довгостроковою пам'яттю (LTM). Як і в мережах Кохонена та Хеммінга, збережений образ, який є найбільш близьким до вхідного образу, зумовить найбільше значення виходу другого шару мережі Гроссберга. Розглянемо зв'язок між мережами Гроссберга та Кохонена. Правило навчання для матриці  $\mathbf{W}^2$  задається рівнянням:

$$1) \text{ за правилом Хебба із загасанням } \frac{dw_{ij}^2(t)}{dt} = \alpha \left( -w_{ij}^2(t) + u_i^2(t)u_j^1(t) \right);$$

2) за правилом Інстар

$$\frac{dw_{ij}^2(t)}{dt} = \alpha u_i^2(t) \cdot \left( -w_{ij}^2(t) + u_j^1(t) \right), \quad (12.7)$$

або у векторній формі:  $\frac{d[\mathbf{w}^2(t)]}{dt} = \alpha u_i^2(t) \cdot \left( -[\mathbf{w}^2(t)] + \mathbf{u}^1(t) \right)$ , де  $\mathbf{w}^2(t)$  –  $i$ -й стовпець матриці  $\mathbf{W}^2$ .

Рівняння (12.7) є неперервною у часі реалізацією зростаючого правила навчання, описаного рівнянням (12.6).

**Приклад 12.3.** Продемонструємо правило навчання мережі Гроссберга. Розглянемо мережу з двома нейронами у кожному шарі. Рівняння

зміни ваги мають вигляд  $\frac{dw_{11}^2(t)}{dt} = u_1^2(t) \cdot \left( -w_{11}^2(t) + u_1^1(t) \right); \frac{dw_{12}^2(t)}{dt} = u_1^2(t) \times$   
 $\times \left( -w_{12}^2(t) + u_2^1(t) \right); \frac{dw_{21}^2(t)}{dt} = u_2^2(t) \cdot \left( -w_{21}^2(t) + u_1^1(t) \right); \frac{dw_{22}^2(t)}{dt} = u_2^2(t) \cdot \left( -w_{22}^2(t) +$   
 $+ u_2^1(t) \right)$  з коефіцієнтом навчання  $\alpha = 1$ . Для спрощення припустимо, що два вхідних образи по черзі подаються на вхід мережі з періодом 0,2 с, а вихід нейрона визначається за 0,2 с. Значення виходів першого та другого шарів для двох різних вхідних образів мають такий вигляд (рис. 12.11):

$$\text{– для першого образу: } \mathbf{u}^1 = \begin{bmatrix} 0,9 \\ 0,45 \end{bmatrix}; \mathbf{u}^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix};$$

$$\text{– для другого образу: } \mathbf{u}^1 = \begin{bmatrix} 0,45 \\ 0,9 \end{bmatrix}; \mathbf{u}^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Перший образ кодується першим нейроном, а другий образ – другим нейроном у другому шарі. Перший рядок вагової матриці модифікується, якщо  $u_1^2(t)$  є активним, а другий – якщо активним є  $u_2^2(t)$ .

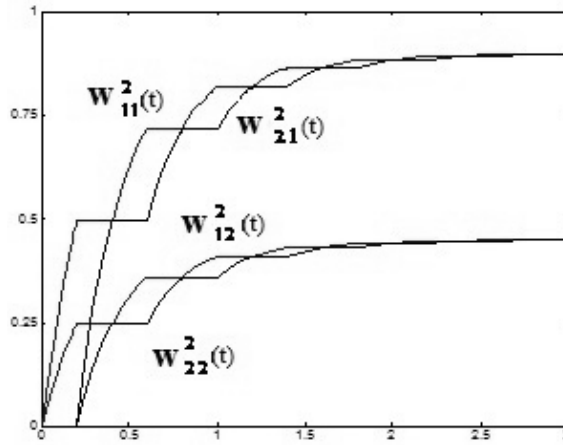


Рис. 12.11. Вихід адаптивних вагових коефіцієнтів

**Зв'язок із правилом навчання Кохонена.** Правило навчання Гроссберга є аналогом неперервного у часі правила навчання Інстар. Покажемо, що мережа Гроссберга (у найпростішій формі) є неперервною в часі версією конкурентної мережі Кохонена. Нагадаємо, що правило навчання Гроссберга має вигляд

$$\frac{d[\mathbf{w}^2(t)]}{dt} = \alpha u_i^2(t) \cdot (-[\mathbf{w}^2(t)] + \mathbf{u}^1(t)). \quad (12.8)$$

Апроксимуємо похідну  $\frac{d[\mathbf{w}^2(t)]}{dt} \approx \frac{{}_i \mathbf{w}^2(t + \Delta t) - {}_i \mathbf{w}^2(t)}{\Delta t}$ , тому рівняння (12.8) можна записати таким чином (дискретна апроксимація правила навчання Гроссберга):

$${}_i \mathbf{w}^2(t + \Delta t) = {}_i \mathbf{w}^2(t) + \alpha(\Delta t) u_i^2(t) \cdot (-[\mathbf{w}^2(t)] + \mathbf{u}^1(t)). \quad (12.9)$$

Якщо перегрупувати елементи, то рівняння (12.9) набуде вигляду

$${}_i \mathbf{w}^2(t + \Delta t) = (1 - \alpha(\Delta t) u_i^2(t)) {}_i \mathbf{w}^2(t) + \alpha(\Delta t) u_i^2(t) \mathbf{u}^1(t).$$

Припустимо, що у другому шарі використано ФА, яка швидша за лінійну. Тоді тільки один нейрон  $i^*$  у цьому шарі буде мати ненульовий вихід. Відповідно тільки  $i^*$ -й рядок матриці ваги матиме такий вигляд:  ${}_{i^*} \mathbf{w}^2(t + \Delta t) = (1 - \alpha') {}_{i^*} \mathbf{w}^2(t) + \alpha' \mathbf{u}^1(t)$ , де  $\alpha' = \alpha(\Delta t) u_{i^*}^2(t)$ , що майже збігається із правилом Кохонена для конкурентної мережі:  ${}_{i^*} \mathbf{w}(q) = (1 - \alpha) {}_{i^*} \mathbf{w}(q - 1) + \alpha \mathbf{p}(q)$ . Вектор ваги нейрона-переможця, який має ненульовий вихід, наблизиться до  $\mathbf{u}^1$  (нормалізованої версії поточного вхідного образу). Між мережею Гроссберга та конкурентною мережею Кохонена існують такі основні відмінності:

1) мережа Гроссберга є неперервною в часі (задовольняє набір нелінійних диференціальних рівнянь);

- 2) перший шар мережі Гроссберга нормалізує вхідні вектори;
- 3) другий шар мережі Гроссберга може виконувати «м'яку» конкуренцію, яка допускає можливість навчання більш ніж одного нейрона (що дозволяє використовувати мережі Гроссберга як мапи ознак), а не конкуренцію «переможець отримує все» мережі Кохонена.

### **Приклади розв'язання задач\***

**Приклад 12.4.** Нехай задано квазіінтегратор із входом  $p = 1$ ,  $\varepsilon$  – системна константа часу.

Визначити вплив величини  $\varepsilon$  на вихід квазіінтегратора.

*Розв'язання.* Рівняння, що описує поведінку квазіінтегратора, має такий вигляд:  $\varepsilon \frac{du(t)}{dt} = -u(t) + p(t)$ . Розв'язок цього диференціального

рівняння для довільного входу  $p(t)$  буде  $u(t) = e^{-\frac{t}{\varepsilon}} u(0) + \frac{1}{\varepsilon} \int_0^t e^{-\frac{(t-\tau)}{\varepsilon}} p(\tau) d\tau$ .

Наприклад, якщо вхідний сигнал  $p(t) = p$  є константою, а початкова умова  $u(0) = 0$ , то маємо  $u(t) = p \left( 1 - e^{-\frac{t}{\varepsilon}} \right)$ . Для  $p(t) = 1$  розв'язок має вигляд

$u(t) = e^{-\frac{t}{\varepsilon}} u(0) + \frac{1}{\varepsilon} \int_0^t e^{-\frac{(t-\tau)}{\varepsilon}} d\tau$ . У цьому випадку вихід квазіінтегратора є фун-

кцією від  $\varepsilon$ , тоді маємо:  $u(t) = e^{-\frac{t}{\varepsilon}} u(0) + (1 - e^{-\frac{t}{\varepsilon}}) = e^{-\frac{t}{\varepsilon}} (u(0) - 1) + 1$ . Цей вихід розпочинається з точки  $u(0)$  і зростає експоненціально (або частково експоненціально залежно від того, чи значення  $u(0)$  більше або менше за одиницю), наближаючись до стійкого стану  $u(\infty) = 1$ . На рис. 12.12 зображено вихід для  $\varepsilon = \{1(\text{«х»}), 0,5(\text{«о»}), 0,25(\text{«-»})\}$  з точки  $u(0) = 0$ . Значення стійкого стану залишається рівним одиниці для всіх випадків і змінюється тільки швидкість виходу.

**Приклад 12.5.** Нехай задано квазіінтегратор зі значенням  $\varepsilon = 1$ .

**I.** Визначити наближене різницеве рівняння для диференціального рівняння квазіінтегратора, апроксимуючи похідні за формулою  $\frac{du(t)}{dt} =$

$$= \frac{u(t + \Delta t) - u(t)}{\Delta t}.$$

---

\* Задачі взято з посібника [14].



**II.** Нехай  $\Delta t = 0,1$ . Порівняти у межах  $0 \leq t \leq 1$  розв'язок різницевого рівняння, отриманого в п. I, із розв'язком диференціального рівняння, що описує поведінку квазіінтегратора для  $p(t) = 1$  та  $u(0) = 0$ .

**III.** Використовуючи модель різницевого рівняння квазіінтегратора, показати, що розв'язок різницевого рівняння буде більшим за вихід квазіінтегратора, який можна зобразити як зважене середнє попередніх входів.

*Розв'язання. I.* Рівняння, що описує поведінку квазіінтегратора, має такий вигляд:  $\varepsilon \frac{du(t)}{dt} = -u(t) + p(t)$ . Тому, якщо апроксимувати похідну, одержимо  $\frac{u(t + \Delta t) - u(t)}{\Delta t} = -u(t) + p(t)$ , або  $u(t + \Delta t) = u(t) + \Delta t(-u(t) + p(t)) = (1 - \Delta t)u(t) + \Delta t p(t)$ .

**II.** Якщо  $\Delta t = 0,1$ , то одержимо різницеве рівняння  $u(t + 0,1) = 0,9u(t) + 0,1p(t)$ . За  $p(t) = 1$  та  $u(0) = 0$  відповідь квазіінтегратора  $u(t)$  має вигляд  $u(0,1) = 0,9u(0) + 0,1p(0) = 0,1$ ;

$$u(0,2) = 0,9u(0,1) + 0,1p(0,1) = 0,9 \cdot (0,1) + 0,1 \cdot (1) = 0,19;$$

$$u(0,3) = 0,9u(0,2) + 0,1p(0,2) = 0,9 \cdot (0,19) + 0,1 \cdot (1) = 0,271;$$

$$u(0,4) = 0,9u(0,3) + 0,1p(0,3) = 0,9 \cdot (0,271) + 0,1 \cdot (1) = 0,3439;$$

$$u(0,5) = 0,9u(0,4) + 0,1p(0,4) = 0,9 \cdot (0,3439) + 0,1 \cdot (1) = 0,4095;$$

$$u(0,6) = 0,4686; \quad u(0,7) = 0,5217; \quad u(0,8) = 0,5695; \quad u(0,9) = 0,6126;$$

$$u(1,0) = 0,6513.$$

**III.** За  $p(t) = 1$ ;  $u(0) = 0$ ,  $\varepsilon = 1$  розв'язок диференціального рівняння  $u(t) = e^{-\frac{t}{\varepsilon}}u(0) + \left(1 - e^{-\frac{t}{\varepsilon}}\right) = (1 - e^{-t})$ . Рис. 12.13 ілюструє співвідношення

між різними розв'язками рівняння: лінія зображує розв'язки диференціального рівняння, а кола – різницевого рівняння (вони дуже близько розміщені, а якщо зменшити інтервал  $\Delta t$ , то можуть бути ще ближчими). Використовуючи рівняння різницевої моделі квазіінтегратора, отримане в п. II, маємо  $u(t + 0,1) = 0,9u(t) + 0,1p(t)$ . Якщо розпочати з нульових початкових умов, одержимо:

$$u(0,1) = 0,9u(0) + 0,1p(0) = 0,1p(0);$$

$$u(0,2) = 0,9u(0,1) + 0,1p(0,1) = 0,9(0,1p(0)) + 0,1p(0,1) = 0,09p(0) + 0,1p(0,1);$$

$$u(0,3) = 0,9u(0,2) + 0,1p(0,2) = 0,081p(0) + 0,09p(0,1) + 0,1p(0,2); \dots;$$

$$u(0,1k) = 0,1((0,9)^{k-1}p(0) + (0,9)^{k-2}p(0,1) + \dots + p((k-1)0,1)).$$

Тому вихід квазіінтегратора – це зважене середнє попередніх входів  $p(0), p(0,1), \dots, p((k-1)0,1)$ , причому останні входи впливають на вихід більше за початкові входи.

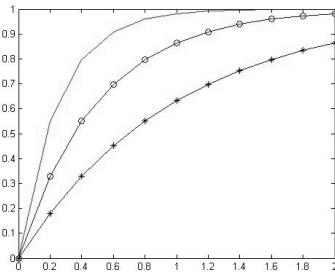


Рис. 12.12. Вплив  $\varepsilon$  на вихід  $u(t) = \frac{t}{(1 - e^{-\varepsilon})}$  квазіінтегратора

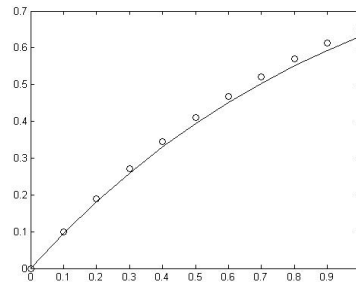


Рис. 12.13. Порівняння розв'язків диференціального та різницевого рівнянь

**Приклад 12.6.** Визначити вихід шунтувальної мережі, якщо  $\varepsilon = 1$ ;  $b^+ = 1$ ;  $b^- = 1$ ;  $p^+ = 0$ ;  $p^- = 10$ ;  $u(0) = 0,5$ .

*Розв'язання.* Рівняння шунтувальної мережі має такий вигляд:  $\varepsilon \frac{du(t)}{dt} = -u(t) + (b^+ - u(t))p^+ - (u(t) + b^-)p^-$ . Для заданих значень параметрів

одержимо:  $\frac{du(t)}{dt} = -u(t) - (u(t) + 1) \cdot 10 = -11u(t) - 10$ . Розв'язком цього рів-

няння буде  $u(t) = e^{-11t}u(0) + \int_0^t e^{-11(t-\tau)}(-10)d\tau = e^{-11t} \cdot 0,5 + \left(-\frac{10}{11}\right)(1 - e^{-11t})$ .

Графік виходу  $u(t)$  шунтувальної мережі, який ніколи не буде меншим за значення  $-b^- = -1$ , зображено на рис. 12.14. Швидкість виходу зростає зі зростанням значення входу. Наприклад, якщо вхід змінюється від  $p^- = 10$  до  $p^+ = 100$ , то вихід буде мати вигляд  $u(t) = e^{-101t}0,5 +$

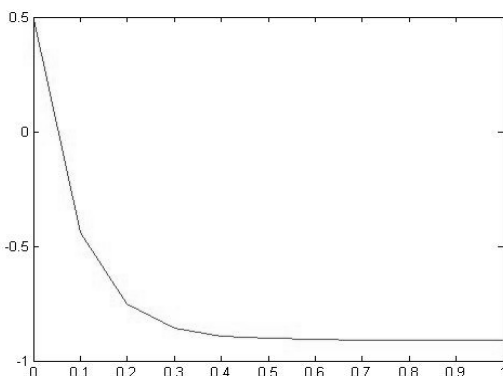


Рис. 12.14. Вихід  $u(t)$  шунтувальної мережі

$+ \left(-\frac{100}{101}\right)(1 - e^{-101t})$ . Оскільки  $e^{-101t}$  спадає швидше за  $e^{-11t}$ , то вихід формується швидше.

**Приклад 12.7.** Визначити вихід першого шару мережі Гроссберга з двома нейронами, з параметрами  $\varepsilon = 1$ ;  $b^{1+} = 1$ ;  $b^{1-} = 0$ , вектором входу  $\mathbf{p} = [c \ 2c]^T$  та початковою умовою  $u(t) = 0$ . Визначити вплив параметра  $c$  на вихід мережі.

*Розв'язання.* Диференціальні рівняння для першого шару мають такий вигляд:

$$\frac{du_1^1(t)}{dt} = -u_1^1(t) + c(1 - u_1^1(t)) - 2cu_1^1(t) = -(1 + 3c)u_1^1(t) + c;$$

$$\frac{du_2^1(t)}{dt} = -u_2^1(t) + 2c(1 - u_2^1(t)) - cu_2^1(t) = -(1 + 3c)u_2^1(t) + 2c.$$

Розв'язками цих рівнянь буде:  $u_1^1(t) = e^{-(1+3c)t}u_1^1(0) + \int_0^t e^{-(1+3c)(t-\tau)}(c)d\tau$ ;

$u_2^1(t) = e^{-(1+3c)t}u_2^1(0) + \int_0^t e^{-(1+3c)(t-\tau)}(2c)d\tau$ . Якщо початкові умови дорівнюють

нулю, то ці рівняння спрощуються до вигляду  $u_1^1(t) = \left(\frac{c}{1+3c}\right) \cdot$

$\left(1 - e^{-(1+3c)t}\right)$ ;  $u_2^1(t) = \left(\frac{2c}{1+3c}\right) \cdot \left(1 - e^{-(1+3c)t}\right)$ . Функціонування  $i$ -го нейрона у стійкому стані описується рівнянням  $u_i^1 = \left(\frac{b^{1+}P}{1+P}\right)\bar{p}_i$ ,  $i = 1, 2$ , а загальне функціонування  $u_1^1(t) + u_2^1(t)$  першого шару мережі ніколи не перевищує значення  $b^{1+} = 1$  відповідно до рівняння  $\sum_{j=1}^{S^1} u_j^1 = u_1^1 + u_2^1 =$

$= \frac{b^{1+}P}{1+P}(\bar{p}_1 + \bar{p}_2) = \frac{b^{1+}P}{1+P}(\bar{p}_1 + \bar{p}_2) = \frac{b^{1+}P}{1+P} \leq b^{1+}$ . Якщо значення  $c$  зростає, то значення стійкого стану поступово збільшується, а вихід з'являється, починаючи з моменту, коли  $e^{-(1+3c)t}$  зменшується швидше, ніж значення  $c$  зростає.

**Приклад 12.8.** Нехай задано двошарову мережу Гроссберга. На вхід другого шару подаються сигнали протягом визначеного проміжку часу, які потім обнуляються.

**I.** Визначити диференціальне рівняння, яке описує зміну загального виходу другого шару  $U^2(t) = \sum_{k=1}^{S^2} u_k^2(t)$  після видалення входу другого шару.

**II.** Визначити диференціальне рівняння, яке описує зміну відносних виходів другого шару  $\frac{u_i^2(t)}{U^2(t)}$  після видалення входу другого шару.

*Розв'язання. I.* Функціонування другого шару мережі Гроссберга описується рівнянням

$$\varepsilon \frac{d\mathbf{u}^2(t)}{dt} = -\mathbf{u}^2(t) + (\mathbf{b}^{2+} - \mathbf{u}^2(t))(\mathbf{W}^{2+}\mathbf{f}^2(\mathbf{u}^2(t)) + \mathbf{W}^2\mathbf{y}^1) -$$

$$-(\mathbf{u}^2(t) + \mathbf{b}^{2-})\mathbf{W}^{2-}\mathbf{f}^2(\mathbf{u}^2(t)).$$

Якщо входи видалити, то маємо  $\mathbf{W}^2\mathbf{y}^1 = 0$ . Для простоти встановимо, що всі елементи пригнічувального зсуву  $b_i^{2-} = 0$ , а всі елементи збуджувального зсуву  $b_i^{2+} = b^{2+}$ ,  $i = 1, \dots, S^2$ . Тоді вихід  $i$ -го нейрона

$$\varepsilon \frac{du_i^2(t)}{dt} = -u_i^2(t) + (b^{2+} - u_i^2(t))f^2(u_i^2(t)) - u_i^2(t) \sum_{k \neq i} f^2(u_k^2(t)),$$

або

$$\varepsilon \frac{du_i^2(t)}{dt} = -u_i^2(t) + b^{2+} f^2(u_i^2(t)) - u_i^2(t) \sum_{k=1}^{S^2} f^2(u_k^2(t)).$$

Якщо позначити  $F^2(t) = \sum_{k=1}^{S^2} f^2(u_k^2(t))$ , то останнє рівняння можна спростити до вигляду  $\varepsilon \frac{du_i^2(t)}{dt} = -(1 + F^2(t))u_i^2(t) + b^{2+} f^2(u_i^2(t))$ . Щоб отримати загальний вихід, просумуємо останнє рівняння за  $i$ , одержимо:  $\varepsilon \frac{dU^2(t)}{dt} = -(1 + F^2(t))U^2(t) + b^{2+} F^2(t)$ . Це рівняння описує зміну виходу другого шару в часі.

**II.** Похідна відносного виходу має вигляд

$$\frac{d}{dt} \left[ \frac{u_i^2(t)}{U^2(t)} \right] = \frac{d}{dt} \left[ \frac{u_i^2(t)}{U^2(t)} \right] = \frac{1}{U^2(t)} \frac{d}{dt} [u_i^2(t)] - \left[ \frac{u_i^2(t)}{(U^2(t))^2} \right] \frac{d}{dt} [U^2(t)].$$

Якщо використати попереднє рівняння для похідних, одержимо:

$$\varepsilon \frac{d}{dt} \left[ \frac{u_i^2(t)}{U^2(t)} \right] = \frac{1}{(U^2(t))^2} [(-(1 + F^2(t))u_i^2(t) + b^{2+} f^2(u_i^2(t))) -$$

$$-\frac{u_i^2(t)}{(U^2(t))^2} (-(1 + F^2(t))U^2(t) + b^{2+} F^2(t))],$$

$$\text{звідки } \varepsilon \frac{d}{dt} \left[ \frac{u_i^2(t)}{U^2(t)} \right] = \frac{1}{(U^2(t))^2} \left[ b^{2+} f^2(u_i^2(t)) - \frac{u_i^2(t)}{(U^2(t))^2} b^{2+} F^2(t) \right], \text{ або}$$

$$\varepsilon \frac{d}{dt} \left[ \overset{-2}{u_i}(t) \right] = \frac{b^{2+} F^2(t)}{U^2(t)} \left[ \frac{f^2(u_i^2(t))}{F^2(t)} - \frac{u_i^2(t)}{U^2(t)} \right]. \quad (12.10)$$

Перетворимо цей вираз:

$$\begin{aligned} \left[ \frac{f^2(u_i^2(t))}{F^2(t)} - \frac{u_i^2(t)}{U^2(t)} \right] &= \frac{1}{F^2(t)U^2(t)} \left[ f^2(u_i^2(t))U^2(t) - u_i^2(t)F^2(t) \right] = \\ &= \frac{1}{F^2(t)U^2(t)} \left[ g^2(u_i^2(t))u_i^2(t) \sum_{k=1}^{S^2} u_k^2(t) - u_i^2(t) \sum_{k=1}^{S^2} g^2(u_k^2(t))u_k^2(t) \right] = \\ &= \frac{u_i^2(t)}{F^2(t)U^2(t)} \left[ \sum_{k=1}^{S^2} u_k^2(t) \left[ g^2(u_i^2(t)) - g^2(u_k^2(t)) \right] \right], \end{aligned}$$

$$\text{де } g^2(u_i^2(t)) = \left[ \frac{f^2(u_i^2(t))}{u_i^2(t)} \right].$$

Підставивши останній вираз у формулу (12.10), одержимо:

$$\varepsilon \frac{d}{dt} \left[ \overset{-2}{u_i}(t) \right] = b^{2+} \overset{-2}{u_i}(t) \left[ \sum_{k=1}^{S^2} \overset{-2}{u_k}(t) \left[ g^2(u_i^2(t)) - g^2(u_k^2(t)) \right] \right].$$

Останнє диференціальне рівняння описує відношення між виходами і може бути використане для відображення характеристик 2-го шару.

**Приклад 12.9.** Нехай ФА другого шару мережі Гроссберга є лінійною.

**I.** Показати, що сусідні виходи другого шару не змінюються після видалення входу.

**II.** Визначити, за якої умови загальний вихід другого шару буде дорівнювати нулю після видалення входу.

*Розв'язання.* **I.** Із прикл. 12.8 відомо, що відносні виходи другого шару після видалення входу визначають за формулою

$$\varepsilon \frac{d}{dt} \left[ \overset{-2}{u_i}(t) \right] = b^{2+} \overset{-2}{u_i}(t) \left[ \sum_{k=1}^{S^2} \overset{-2}{u_k}(t) \left[ g^2(u_i^2(t)) - g^2(u_k^2(t)) \right] \right].$$

Функція активації другого шару  $f^2(u)$  є лінійною, тобто  $f^2(u) = cu$ , тому  $g^2(u_i^2(t)) = \frac{f^2(u_i^2(t))}{u_i^2(t)} = \frac{cu}{u} = c$ . Підставивши цей вираз у диферен-

ціальне рівняння, одержимо  $\varepsilon \frac{d}{dt} \left[ \overset{-2}{u_i}(t) \right] = b^{2+} \overset{-2}{u_i}(t) \left[ \sum_{k=1}^{S^2} \overset{-2}{u_k}(t) [c - c] \right] = 0$ .

Отже, відносні виходи не були змінені.

II. Із прикл. 12.8 відомо, що загальний вихід другого шару після видалення входу визначають за формулою  $\varepsilon \frac{dU^2(t)}{dt} = -(1 + F^2(t))U^2(t) + b^{2+}F^2(t)$ . Якщо ФА  $f^2(u) = cu$ , то маємо:

$$F^2(t) = \sum_{k=1}^{S^2} f^2(u_k^2(t)) = \sum_{k=1}^{S^2} cu_k^2(t) = c \left( \sum_{k=1}^{S^2} u_k^2(t) \right) = cU^2(t).$$

Отже, диференціальне рівняння можна записати у вигляді  $\varepsilon \frac{dU^2(t)}{dt} = -(1 + cU^2(t))U^2(t) + b^{2+}cU^2(t) = -(1 - b^{2+}c + cU^2(t))U^2(t)$ .

Щоб визначити розв'язок цього рівняння, прирівняємо похідну до нуля, одержимо:  $0 = -(1 - b^{2+}c + cU^2(t)) \cdot U^2(t)$ . Тому маємо два розв'язки:  $U^2(t) = 0$  і  $U^2(t) = \frac{b^{2+}c - 1}{c}$ . Визначимо умови, за яких загальний вихід буде збігатися до кожного з указаних розв'язків.

*Перший випадок.* За умови  $1 \geq b^{2+}c$  похідна загального виходу має вигляд  $\varepsilon \frac{dU^2(t)}{dt} = -(1 - b^{2+}c + cU^2(t)) \cdot U^2(t)$  і завжди буде від'ємною для додатних значень  $U^2(t)$ . Отже, загальний вихід збігається до нуля:

$$\lim_{t \rightarrow \infty} U^2(t) = 0.$$

*Другий випадок.* За умови  $1 < b^{2+}c$  маємо:

а) якщо  $U^2(0) > \frac{b^{2+}c - 1}{c}$ , то похідна загального виходу набуває від'ємних значень, поки вихід не стане рівним  $U^2(t) = \frac{b^{2+}c - 1}{c}$  (у цьому разі похідна дорівнює нулю). Тому маємо:

$$\lim_{t \rightarrow \infty} U^2(t) = \frac{b^{2+}c - 1}{c};$$

б) якщо  $U^2(0) < \frac{b^{2+}c - 1}{c}$ , то похідна загального виходу набуває додатних значень, поки вихід не стане рівним  $U^2(t) = \frac{b^{2+}c - 1}{c}$  (у цьому разі похідна дорівнює нулю). Тому маємо:

$$\lim_{t \rightarrow \infty} U^2(t) = \frac{b^{2+}c - 1}{c},$$

і ФА другого шару є лінійною. Загальний вихід прямує до нуля, якщо  $1 \geq b^{2+}c$ , в іншому випадку загальний вихід прямує до значення  $\frac{b^{2+}c-1}{c}$ . При цьому відносні виходи не змінюються.

Як приклад розглянемо рівняння функціонування 2-го шару:

$$\frac{du_1^2(t)}{dt} = -u_1^2(t) + (1,5 - u_1^2(t))u_1^2(t) - u_1^2(t)u_2^2(t);$$

$$\frac{du_2^2(t)}{dt} = -u_2^2(t) + (1,5 - u_2^2(t))u_2^2(t) - u_2^2(t)u_1^2(t).$$

Для  $\varepsilon = 1$ ;  $b^{2+} = 1,5$ ;  $c = 1$  маємо  $1 < b^{2+}c$ . Тому загальний вихід збігається до значення

$$\lim_{t \rightarrow \infty} U^2(t) = \frac{b^{2+}c-1}{c} = \frac{1,5-1}{1} = 0,5.$$

Вихід другого шару для двох різних початкових умов  $\mathbf{u}^2(0) = \begin{bmatrix} 0,75 \\ 0,5 \end{bmatrix}$ ;  $\mathbf{u}^2(0) = \begin{bmatrix} 0,15 \\ 0,1 \end{bmatrix}$  зображено на рис. 12.15. Загальний вихід збігається до значення 0,5 для обох цих умов. При цьому відносні виходи для обох випадків однакові.

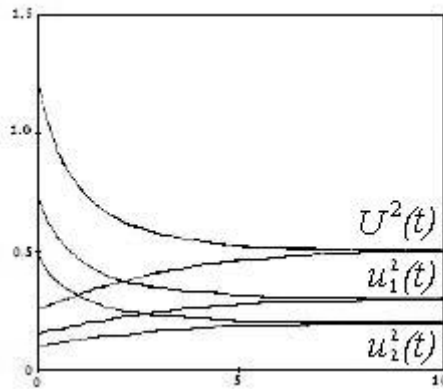


Рис. 12.15. Вихід другого шару для лінійної ФА  $f^2(u)$

**Приклад 12.10.** Показати, що неперервне за часом правило Хебба із загасанням еквівалентне дискретному за часом правилу Хебба.

*Розв'язання.* Неперервне за часом правило Хебба із загасанням має

вигляд  $\frac{dw_{ij}^2(t)}{dt} = \alpha(-w_{ij}^2(t) + u_i^2(t)u_j^1(t))$ . Апроксимувавши похідну за формулою  $\frac{dw_{ij}^2(t)}{dt} = \frac{w_{ij}^2(t + \Delta t) - w_{ij}^2(t)}{\Delta t}$ , одержимо правило Хебба вигляду

$w_{ij}^2(t + \Delta t) = w_{ij}^2(t) + \alpha \Delta t (w_{ij}^2(t) + u_i^2(t) u_j^1(t))$ , звідки маємо  $w_{ij}^2(t + \Delta t) = (1 - \alpha \Delta t) w_{ij}^2(t) + \alpha \Delta t u_i^2(t) u_j^1(t)$ , або у векторній формі:  $\mathbf{W}^2(t + \Delta t) = (1 - \alpha \Delta t) \mathbf{W}^2(t) + \alpha \Delta t \mathbf{u}^2(t) (\mathbf{u}^1(t))^T$ . Якщо порівняти одержане рівняння з дискретним за часом правилом Хебба  $\mathbf{W}(q) = (1 - \gamma) \mathbf{W}(q - 1) + \alpha \mathbf{u}(q) \mathbf{p}^T(q)$ , то можна побачити їхню подібність.

### Контрольні запитання

1. Опишіть систему зору людини та її складові.
2. Дайте визначення інтегратора та квазіінтегратора, нелінійної шунтувальної моделі Гроссберга.
3. Визначте двошарову конкурентну мережу Гроссберга та її складові.

### Задачі для самостійного розв'язання

**Задача 12.1.** Оцінити квазіінтегратор. Визначити вихід  $u(t)$ , якщо:

- а)  $\varepsilon = 1$ ,  $u(0) = 1$ ,  $p(t) = 0,5$ ; б)  $\varepsilon = 1$ ,  $u(0) = 1$ ,  $p(t) = 2$ .

Перевірити правильність виконання завдань, написавши  $m$ -файл системи MatLab, який моделює функціонування квазіінтегратора; побудувати графік виходу для кожного випадку.

**Задача 12.2.** Оцінити шунтувальну мережу. Визначити вихід мережі  $u(t)$ , якщо:

- а)  $\varepsilon = 2$ ,  $b^+ = 3$ ,  $b^- = 1$ ,  $p^+ = 0$ ,  $p^- = 5$  і  $u(0) = 1$ ;  
 б)  $\varepsilon = 2$ ,  $b^+ = 3$ ,  $b^- = 1$ ,  $p^+ = 0$ ,  $p^- = 50$  і  $u(0) = 1$ ;  
 в)  $\varepsilon = 2$ ,  $b^+ = 3$ ,  $b^- = 1$ ,  $p^+ = 50$ ,  $p^- = 0$  і  $u(0) = 1$ .

Перевірити правильність виконання завдань, написавши  $m$ -файл системи MatLab, який моделює шунтувальну мережу. Побудувати графік виходу для кожного випадку.

**Задача 12.3.** Розглянути перший шар мережі Гроссберга з двома нейронами, параметрами  $b^{1+} = 0,5$ ;  $\varepsilon = 0,5$ , вектором входу  $\mathbf{p} = [2 \ 1]^T$  та початковою умовою  $u(0) = 0$ .

**I.** Визначити вихід першого шару та розв'язок диференціального рівняння для першого шару. Перевірити, чи відповідає розв'язок виходу.

**II.** Перевірити правильність виконання завдань, написавши  $m$ -файл, який моделює перший шар мережі Гроссберга. Побудувати графік виходу.

**Задача 12.4.** Виконати завдання 12.3 для вхідного вектора  $\mathbf{p} = [20 \ 10]^T$ .

**Задача 12.5.** Визначити диференціальне рівняння, яке описує зміну загального виходу першого шару  $U^1(t) = \sum_{i=1}^S u_i^1(t)$ . Використати методи, рекомендовані для прикл. 12.8.



**Задача 12.6.** Нехай другий шар мережі Гроссберга має два нейрони з такими параметрами:  $f^2(u) = 2u$ ;  $\varepsilon = 1$ ;  $b^{2+} = 1$  і  $b^{2-} = 0$ . Входи подаються протягом визначеного часу, після чого видаляються.

**I.** Визначити встановлений стан виходу  $U^2(t)$  за  $t \rightarrow \infty$ .

**II.** Повторити п. I, якщо  $b^{2+} = 0,25$ .

**III.** Перевірити правильність виконання завдань, написавши  $m$ -файл системи MatLab, який моделює другий шар мережі Гроссберга. Побудувати графіки виходу для таких початкових умов:  $\mathbf{u}^2(0) = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ;  $\mathbf{u}^2(0) = \begin{bmatrix} 0,2 \\ 0,1 \end{bmatrix}$ .

**Задача 12.7.** Нехай ФА другого шару мережі Гроссберга має вигляд:  $f(u) = cu^2$  і  $\varepsilon = 1$ ,  $b^{2+} = 1$ .

**I.** Використовуючи результати, отримані в прикл. 12.9, показати, що після видалення входів другого шару всі відповідні виходи будуть дорівнювати нулю, крім одного з найбільшим початковим значенням («переможець отримує все»).

**II.** За яких значень  $c$  загальний вихід  $u^2(t)$  буде мати ненульову стійку точку (значення стійкого стану)?

**III.** Якщо умови п. I виконуються, то визначити: 1) яким буде значення стійкого стану  $u^2(t)$ ; 2) чи буде воно залежати від початкової умови  $u(0)$ .

**IV.** Перевірити правильність виконання завдань, написавши  $m$ -файл, який моделює другий шар мережі Гроссберга за  $c = 4$ ;  $u^2(0) = 3$ .

**Задача 12.8.** Змоделювати зміну адаптивної ваги  $\mathbf{W}^2$  мережі Гроссберга, якщо  $\varepsilon = 1$ . При цьому припустити, що: 1) два різних входні образи подаються до мережі з періодом 0,2 с; 2) обидва шари збігаються дуже швидко порівняно зі збіжністю ваги, так що виходи нейрона не змінюються протягом 0,2 с. Виходи першого і другого шарів від різних

вхідних образів мають вигляд: для першого образу –  $\mathbf{u}^1 = \begin{bmatrix} 0,8 \\ 0,2 \end{bmatrix}$ ;

$\mathbf{u}^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ; для другого образу –  $\mathbf{u}^1 = \begin{bmatrix} 0,5 \\ 0,5 \end{bmatrix}$ ;  $\mathbf{u}^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ .

## Розділ 13. ТЕОРІЯ АДАПТИВНИХ РЕЗОНАНСІВ. МЕРЕЖА ART1

### 13.1. Мережа ART1: структура

У розд. 11, 12 було показано, що однією з ключових проблем конкуруючих мереж є стійкість навчання: значення вагової матриці буде однаковим (збігатися). Розглянемо модифікований тип конкуруючого навчання, який називають теорією адаптивних резонансів (ART), що був розроблений для вирішення проблеми надання стабільності процесу навчання.

Основна проблема мереж Гроссберга та конкуруючих мереж полягає в тому, що вони не завжди формують стійкі кластери. Ст. Гроссберг показав, що якщо кількість вхідних образів не надто велика, або якщо вони утворюють кластери, кількість яких приблизно дорівнює кількості нейронів у другому шарі, то процес навчання стабілізується. Він також показав, що стандартні конкуруючі мережі не мають стабільного навчання, якщо вхідні значення є випадковими.

Нестабільність навчання виникає через здатність мережі пристосовуватися (гнучкість мережі). Ст. Гроссберг назвав цю проблему проблемою «стабільності/гнучкості». Розглянемо, як система може бути чутливою до значущих нових значень і залишатися стабільною у відповідь на значення, що не відповідають стабільності. Відомо, що біологічні системи у цьому сенсі є стабільними. Наприклад, людина може з легкістю впізнати обличчя своєї матері, навіть якщо не бачила її тривалий час і зустріла багато нових людей. Досліджуючи проблему «стабільності/гнучкості», Ст. Гроссберг і Г. Карпентер розробили «теорію адаптивних резонансів». Мережі ART ґрунтуються на мережах Гроссберга. Основна ідея мереж ART ґрунтується на використанні «очікувань», коли кожний вхідний вектор порівнюється з векторами-прототипами: якщо вектор-прототип і вхідний вектор подібні, то цей вектор-прототип є очікуванням.

Розглянемо детально одну з ART-мереж – мережу ART1, розроблену лише для бінарного вхідного вектора. На прикладі цієї архітектури можна зрозуміти особливості теорії адаптивних резонансів.

Основну архітектуру ART-мереж, яка є модифікацією мережі Гроссберга та створена для стабілізації процесу навчання, зображено на рис. 13.1. Мережа ART складається з таких підсистем [14]:

1. *Перший шар* (L1) реалізує нормалізацію, порівняння вхідного образу та очікування. Зв'язки Інстар L1–L2 реалізують операцію кластеризації (кожний рядок матриці  $\mathbf{W}^{1:2}$  є прототипом образу).

2. Другий шар (L2) реалізує змагання, підвищення контрастності. Зв'язки Оустар L2–L1 реалізують очікування та повторний виклик образу (кожний стовпець матриці  $W^{2:1}$  є прототипом образу).

3. Підсистема орієнтації реалізує скидання, коли очікування не відповідає входу, деактивацію поточного нейрона-переможця.

Нагадаємо, що з'єднання шарів L1–L2 у мережі Гроссберга має вигляд нейронів Інстар, які виконують кластеризацію. Якщо після нормалізації вхідний образ подається до мережі, то його множать на матрицю  $W^{1:2}$ . Після цього у другому шарі виконується змагання з метою визначення рядка вагової матриці, який є найближчим до вектора входу, цей рядок переміщується до вхідного вектора. Після закінчення навчання кожний рядок вагової матриці  $W^{1:2}$  є вектором-прототипом образу, який зображує кластер вхідних векторів.

У мережах ART1 навчання здійснюється на множині з'єднань зі зворотним зв'язком із другого шару до першого. Ці з'єднання мають вигляд нейронів Оутстар. Коли нейрон у другому шарі активований, він відтворює образ–прототип (очікування) у першому шарі. Потім перший шар порівнює очікування та вхідний образ: якщо вони не відповідають один одному, то підсистема орієнтації робить скидання в другому шарі, що призводить до деактивації поточного нейрона-переможця, а поточне очікування видаляється. Якщо очікування не відповідає входу, то в другому шарі виконується нове змагання: новий нейрон-переможець із другого шару викликає нове очікування першого шару завдяки з'єднанню L2–L1. Цей процес продовжується доти, поки очікування з'єднання L2–L1 не буде відповідати вхідному вектору-прототипу образу.

Розглянемо кожну підсистему мережі ART1 окремо.

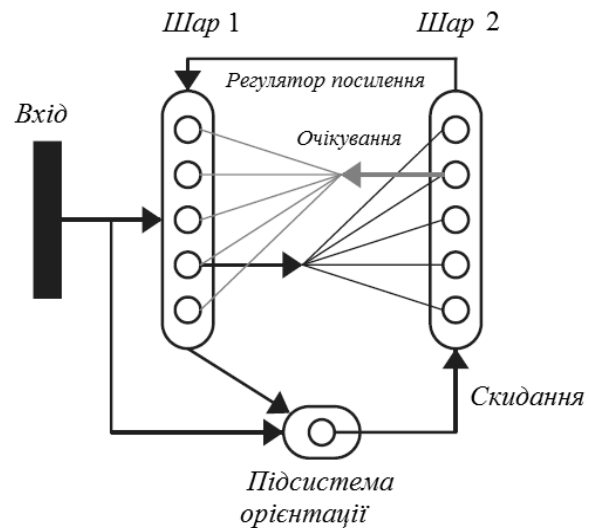
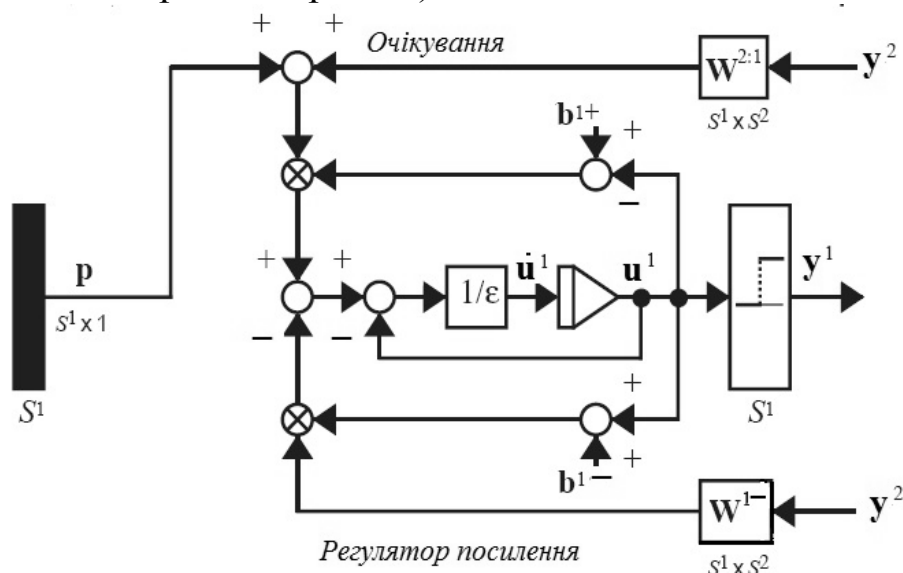


Рис. 13.1. Основна архітектура ART-мережі

## 13.2. Мережа ART1: перший шар

Головне призначення першого шару мережі ART1 (рис 13.2) полягає у порівнянні вхідного вектора з очікуваним значенням з другого шару (обидва вектори є бінарними).



$$\varepsilon \frac{du^1(t)}{dt} = -u^1(t) + (b^{1+} - u^1(t)) \left( p + W^{2:1} y^2(t) \right) - (u^1(t) + b^{1-}) [W^{1-}] y^2(t)$$

Рис. 13.2. Перший шар мережі ART1

Якщо між цими векторами немає відповідності, підсистема орієнтації виконує скидання у другому шарі (перетворює вектор очікування на нульовий вектор), інакше перший шар об'єднує очікування і вхід, формуючи новий вектор-прототип. Перший шар мережі ART1 схожий на перший шар мережі Гроссберга. Для мережі ART1 не передбачено процедури нормалізації першим шаром, більше того, немає з'єднання із вхідним вектором. Збуджувальний вхід першого шару мережі ART1 складається з комбінації вхідних образів та очікування з'єднання L2–L1. Пригнічувальний вхід складається з сигналу регулятора посилення від другого шару.

Роботу першого шару (шунтувальна модель) описують рівнянням

$$\varepsilon \frac{du^1(t)}{dt} = -u^1(t) + (b^{1+} - u^1(t)) \left( p + W^{2:1} y^2(t) \right) - (u^1(t) + b^{1-}) [W^{1-}] y^2(t). \quad (13.1)$$

Вихід першого шару має вигляд  $y^1 = \text{hardlim}^+(u^1)$ , де  $\text{hardlim}^+(u) = \begin{cases} 1, u > 0; \\ 0, u \leq 0. \end{cases}$  Рівняння (13.1) зображує шунтувальну модель зі збуджуванням входом  $p^+ = p + W^{2:1} y^2(t)$ . Якщо  $j$ -й нейрон другого шару виграв змагання, то його вихід дорівнює одиниці, а інші нейрони мають нульовий вихід:

$$\mathbf{W}^{2:1} \mathbf{y}^2 = \begin{bmatrix} \mathbf{w}_1^{2:1} & \dots & \mathbf{w}_{s^2}^{2:1} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \end{bmatrix} = \mathbf{w}_j^{2:1}, \text{ де } \mathbf{w}_j^{2:1} - j\text{-й стовпець матриці } \mathbf{W}^{2:1}, \text{ визна-}$$

ченої за правилом навчання Оутстар. Отже, збуджуваний вхід є сумою вхідного вектора та очікування зв'язку L2–L1:  $\mathbf{p} + \mathbf{W}^{2:1} \mathbf{y}^2 = \mathbf{p} + \mathbf{w}_j^{2:1}$ . Кожний стовпець матриці  $\mathbf{W}^{2:1}$  зображує різне очікування (вектор-прототип образу). Перший шар об'єднує вхідний вектор з очікуванням, використовуючи логічну операцію «І».

Пригнічуваний вхід першого шару є елементом регулятора посилення і має вигляд  $\mathbf{p}^- = [\mathbf{W}^{1-}] \mathbf{y}^2(t)$ , де  $\mathbf{W}^{1-} = \begin{bmatrix} 1 & \dots & 1 \\ \dots & & \dots \\ 1 & \dots & 1 \end{bmatrix}$ . Отже, пригнічуваний вхід

кожного нейрона першого шару є сумою всіх виходів другого шару. Оскільки в другому шарі використовується змагання типу «переможець отримує все», то кожного разу, коли другий шар буде активним, після змагання лише один елемент  $\mathbf{y}^2$  буде ненульовим і дорівнюватиме одиниці.

*Регулятор посилення* входу першого шару допомагає визначити нейрони із виходом «1», коли другий шар активний. У випадку, коли другий шар неактивний, усі нейрони мають нульовий вихід. Значення регулятора посилення стане зрозумілим, якщо проаналізувати стійкий стан першого шару.

**Аналіз стійкого стану.** Вихід  $i$ -го нейрона першого шару описують формулою

$$\varepsilon \frac{du_i^1}{dt} = -u_i^1 + (b^{1+} - u_i^1) \left( p_i + \sum_{j=1}^{s^2} w_{ij}^2 y_j^2 \right) - (u_i^1 + b^{1-}) \sum_{j=1}^{s^2} y_j^2, \quad \varepsilon \ll 1. \quad (13.2)$$

Дослідимо вихід стійкого стану системи для двох різних випадків:

1. *Другий шар є неактивним*, тобто  $y_j^2 = 0$  для кожного  $j$ . У цьому разі рівняння (13.2) спрощується до вигляду  $\varepsilon \frac{du_i^1}{dt} = -u_i^1 + (b^{1+} - u_i^1) p_i$ .

У стійкому стані, тобто за  $\frac{du_i^1(t)}{dt} = 0$ , маємо  $0 = -u_i^1 + (b^{1+} - u_i^1) p_i = -(1 + p_i) u_i^1 + b^{1+} p_i$ , звідки вихід  $i$ -го нейрона у стійкому стані  $u_i^1 = \frac{b^{1+} p_i}{1 + p_i}$ . Отже, якщо  $p_i = 0$ , то  $u_i^1 = 0$ , а якщо  $p_i = 1$ , то  $u_i^1 = \frac{b^{1+}}{2} > 0$ .

Застосувавши ФА  $hardlim^+$  для першого шару, одержимо  $\mathbf{y}^1 = \mathbf{p}$ . Таким

чином, якщо другий шар неактивний, то вихід першого шару дорівнює вхідному вектору.

2. Другий шар є активним, тобто один нейрон має вихід «1», а всі інші нейрони – вихід «0». Нехай  $j$ -й нейрон є нейроном-переможцем у другому шарі, тоді  $y_j^2 = 1$  й  $y_k^2 = 0$ , якщо  $k \neq j$ . У цьому разі рівняння

$$(13.2) \text{ спрощується до вигляду } \varepsilon \frac{du_i^1}{dt} = -u_i^1 + (b^{1+} - u_i^1)(p_i + w_{ij}^{2:1}) - (u_i^1 + b^{1-}).$$

У стабільному стані, тобто за  $\frac{du_i^1(t)}{dt} = 0$ , одержимо:

$$0 = -u_i^1 + (b^{1+} - u_i^1)(p_i + w_{ij}^{2:1}) - (u_i^1 + b^{1-}) = -(1 + p_i + w_{ij}^{2:1} + 1)u_i^1 + b^{1+}(p_i + w_{ij}^{2:1}) - b^{1-},$$

звідки вихід  $i$ -го нейрона у стійкому стані

$$u_i^1 = \frac{b^{1+}(p_i + w_{ij}^{2:1}) - b^{1-}}{2 + p_i + w_{ij}^{2:1}}. \quad (13.3)$$

Нагадаємо, що перший шар повинен поєднувати вектор входу із очікуванням другого шару  $w_j^{2:1}$ . Оскільки вхідні вектори та очікування є двійковими, то для додавання двох векторів будемо використовувати логічну операцію «І». Інакше кажучи, необхідно, щоб виконувалася одна з нерівностей:

$$1) u_i^1 < 0, \text{ якщо } p_i = 0 \text{ або } w_{ij}^{2:1} = 0;$$

$$2) u_i^1 > 0, \text{ якщо } p_i = 1 \text{ і } w_{ij}^{2:1} = 1.$$

Якщо застосувати ці умови до рівняння (13.3), одержимо такі нерівності:  $b^{1+}(2) - b^{1-} > 0$ ,  $b^{1+} - b^{1-} < 0$ , які можна об'єднати таким чином:

$$2b^{1+} > b^{1-} > b^{1+}. \quad (13.4)$$

Для виконання умови (13.4) можна використати значення  $b^{1+} = 1$  і  $b^{1-} = 1,5$ . Якщо виконується умова (13.4) та  $j$ -й нейрон другого шару активний, то на виході першого шару одержимо  $y^1 = p \cap w_j^{2:1}$ , де символ « $\cap$ » означає логічну операцію «І». Щоб виконати цю логічну операцію, необхідно керувати процесом посилення. Розглянемо чисельник виразу (13.3):  $b^{1+}(p_i + w_{ij}^{2:1}) - b^{1-}$ , де елемент  $b^{1-}$  множиться на регулятор посилення, який дорівнює одиниці. Якщо другий шар не є активним (тобто неактивовано жодних очікувань), регулятор посилення дорівнює нулю. Отже, маємо стійкий стан першого шару в таких випадках:

$$1) \text{ якщо другий шар не є активним (усі } y_j^2 = 0), \text{ то } y^1 = p;$$

2) якщо другий шар є активним (один  $y_j^2 = 1$ ), то

$$\mathbf{y}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1}. \quad (13.5)$$

Розглянемо приклад функціонування першого шару.

**Приклад 13.1.** Нехай другий шар мережі ART1 складається із двох нейронів і задано такі параметри:  $\varepsilon = 0,1$ ;  $b^{1+} = 1$ ;  $b^{1-} = 1,5$ ;  $\mathbf{W}^{2:1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ ;  $\mathbf{p} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . Розглянемо випадок, коли другий шар є активним і другий нейрон другого шару перемагає, тоді функціонування першого шару описують рівняннями

$$\begin{aligned} 0,1 \frac{du_1^1}{dt} &= -u_1^1 + (1 - u_1^1)(p_1 + w_{12}^{2:1}) - (u_1^1 + 1,5) = \\ &= -u_1^1 + (1 - u_1^1)(0 + 1) - (u_1^1 + 1,5) = -3u_1^1 - 0,5; \\ 0,1 \frac{du_2^1}{dt} &= -u_2^1 + (1 - u_2^1)(p_2 + w_{22}^{2:1}) - (u_2^1 + 1,5) = \\ &= -u_2^1 + (1 - u_2^1)(1 + 1) - (u_2^1 + 1,5) = -4u_2^1 + 0,5. \end{aligned}$$

Ці рівняння можна спростити до вигляду  $\frac{du_1^1}{dt} = -30u_1^1 - 5$ ;

$\frac{du_2^1}{dt} = -40u_2^1 + 5$ . Якщо припустити, що обидва нейрони мають нульові

початкові умови, тобто  $u_1^1(0) = u_2^1(0) = 0$ , то розв'язок матиме вигляд

(рис. 13.3):  $u_1^1(t) = -\frac{1}{6}(1 - e^{-30t})$ ;  $u_2^1(t) = \frac{1}{8}(1 - e^{-40t})$ . Зазначимо, що  $u_1^1(t)$  збігається до від'ємної величини, а  $u_2^1(t)$  – до додатної, тому  $y_1^1(t)$  збігається до нуля, а  $y_2^1(t)$  – до одиниці. Це відповідає стійкому стану:  $\mathbf{p} \cap \mathbf{w}_2^{2:1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 1 \end{bmatrix} =$

$$= \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{y}^1.$$

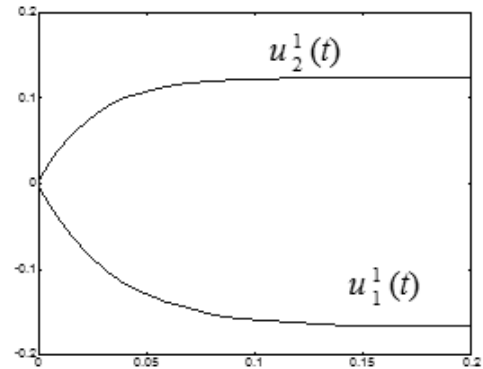


Рис. 13.3. Графік виходу першого шару мережі

### 13.3. Мережа ART1: другий шар

Основне призначення другого шару мережі ART1 полягає у збільшенні контрастності вихідного образу для нейрона-переможця, оскільки лише нейрон із найбільшим значенням входу має ненульовий вихід. Другий шар мережі ART1 використовує інтегратор (рис. 13.4), в якому всі додатні виходи перетворюються на нуль (тобто виконується скидання), якщо сигнал  $y^0$  є додатним. Після скидання виходи залишаються пригніченими протягом тривалого часу (поки не відбудеться відповідний збіг і вагові коефіцієнти не будуть оновлені) і не можуть набути значення, більшого за «0».

Другий шар мережі ART1 (рис. 13.5) схожий на другий шар мережі Гроссберга. Відмінність між ними полягає в тому, що другий шар ART1-мережі використовує:

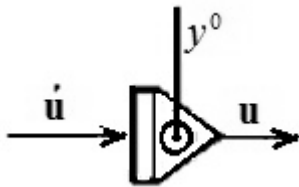
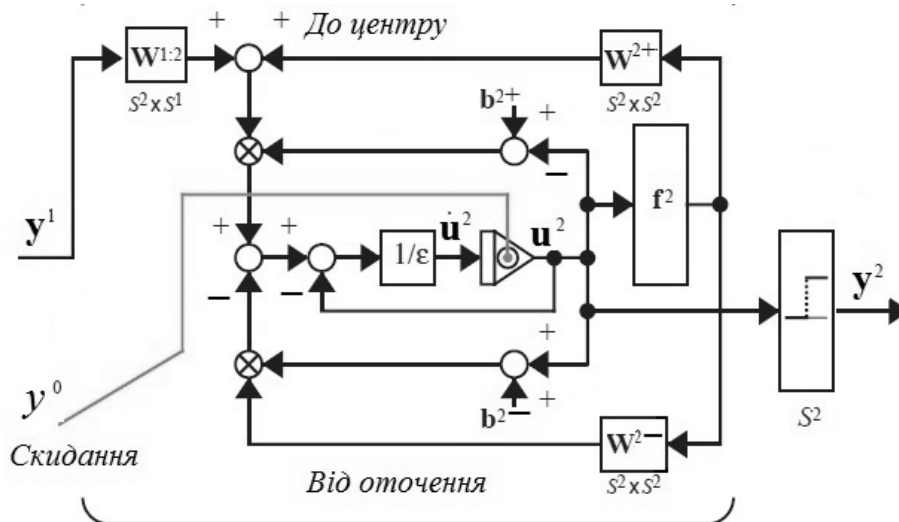


Рис.13.4. Інтегратор

1) інтегратор, який виконує скидання (на виході підсистеми орієнтації формується сигнал скидання  $y^0$ , якщо зафіксовано невідповідність у першому шарі між вхідним сигналом та очікуванням зв'язку L2-L1);

2) дві ФА:  $f^2(u^2)$  для з'єднань зворотного зв'язку «до центру / від оточення» та  $hardlim^+(u^2)$  для одержання на виході другого шару бінарного сигналу за формулою  $y^2 = hardlim^+(u^2)$ .



$$\varepsilon \frac{du^2}{dt} = -u^2 + (b^{2+} - u^2) \left( W^{2+} f^2(u^2) + W^{1:2} y^1 \right) - (u^2 + b^{2-}) W^{2-} f^2(u^2)$$

Рис. 13.5. Другий шар мережі ART1



Функціонування другого шару мережі ART1 описують рівнянням

$$\varepsilon \frac{d\mathbf{u}^2(t)}{dt} = -\mathbf{u}^2(t) + (\mathbf{b}^{2+} - \mathbf{u}^2(t))(\mathbf{W}^{2+}\mathbf{f}^2(\mathbf{u}^2(t)) + \mathbf{W}^{1:2}\mathbf{y}^1) -$$

$$-(\mathbf{u}^2(t) + \mathbf{b}^{2-})\mathbf{W}^{2-}\mathbf{f}^2(\mathbf{u}^2(t)).$$

Це рівняння зображує шунтувальну модель зі збуджуванням входом  $\mathbf{p}^+ = \mathbf{W}^{2+}\mathbf{f}^2(\mathbf{u}^2(t)) + \mathbf{W}^{1:2}\mathbf{y}^1$ , де матриця  $\mathbf{W}^{2+}$  забезпечує з'єднання зворотного зв'язку типу «до центру» і складається з адаптивних вагових коефіцієнтів, які є аналогами вагових коефіцієнтів мережі Кохонена. Елементи матриці  $\mathbf{W}^{1:2}$  одержані за правилом Інстар, а її рядки після навчання відображають вектори-прототипи. Пригнічуваний вхід шунтувальної моделі має вигляд  $\mathbf{p}^- = \mathbf{W}^{2-}\mathbf{f}^2(\mathbf{u}^2(t))$ , де  $\mathbf{W}^{2-}$  забезпечує з'єднання зворотного зв'язку «від оточення».

Розглянемо приклад функціонування другого шару ART1-мережі.

**Приклад 13.2.** Нехай другий шар ART1-мережі складається з двох нейронів і має такі параметри:  $\varepsilon = 0,1$ ;  $\mathbf{b}^{2+} = \mathbf{b}^{2-} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{W}^{1:2} = \begin{bmatrix} (\mathbf{w}^{1:2})^T \\ (\mathbf{w}^{1:2})^T \end{bmatrix} =$

$$= \begin{bmatrix} 0,5 & 0,5 \\ 1 & 0 \end{bmatrix}; f^2(u) = \begin{cases} 10u^2, & \text{якщо } u \geq 0; \\ 0, & \text{якщо } u < 0. \end{cases}$$

Функціонування другого шару описується рівняннями

$$0,1 \frac{du_1^2(t)}{dt} = -u_1^2(t) + (1 - u_1^2(t))\left(f^2(u_1^2(t)) + (\mathbf{w}^{1:2})^T \mathbf{y}^1\right) - (u_1^2(t) + 1)f^2(u_2^2(t));$$

$$0,1 \frac{du_2^2(t)}{dt} = -u_2^2(t) + (1 - u_2^2(t))\left(f^2(u_2^2(t)) + (\mathbf{w}^{1:2})^T \mathbf{y}^1\right) - (u_2^2(t) + 1)f^2(u_1^2(t)).$$

Ці рівняння збігаються з рівняннями функціонування другого шару мережі Гроссберга за винятком параметра  $b^{2-} = 1$ , тому значення  $u_1^2(t)$  і  $u_2^2(t)$  належать інтервалу  $[-1; 1]$ . Вхідні значення другого шару є результатом скалярного добутку образів-прототипів (рядків вагової матриці  $\mathbf{W}^{1:2}$ ) на вихід першого шару  $\mathbf{y}^1$ . Рядки вагової матриці  $\mathbf{W}^{1:2}$  є нормованими. Найбільше значення скалярного добутку відповідає образу-прототипу, який є найближчим до значення вектора входу першого шару. Потім другий шар виконує конкуренцію (змагання) між нейронами.

Функцію активації  $f^2(u)$  обрано з міркувань надлінійності (див. у розд. 12 вплив вибору ФА  $f^2(u)$  на вихід мережі Гроссберга). Цей вибір зумовить додатні значення  $u$  для нейрона з найбільшим входом та від'ємні значення  $u$  для інших нейронів. Змагання нейронів дозволяє

одержати на виході другого шару такі значення: одне значення «1», інші – «0» (нагадаємо, що для обчислення виходу другого шару застосовують ФА *hardlim*<sup>+</sup>). Графік виходу другого шару з вектором входу  $\mathbf{y}^1 = [1 \ 0]^T$  зображено на рис. 13.6. Скалярний добуток другого рядка матриці  $\mathbf{W}^{1:2}$  на  $\mathbf{y}^1$  має більше значення, ніж скалярний добуток першого рядка на  $\mathbf{y}^1$ , тому другий нейрон перемагає у змаганні. У стабільному стані  $u_2^2(t)$  має додатне значення, а  $u_1^2(t)$  – від’ємне. Стан рівноваги виходу другого шару має вигляд  $\mathbf{y}^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ .

Отже, стан рівноваги як результат функціонування другого шару ART1-мережі має вигляд:  $y_i^2 = \begin{cases} 1, & \text{якщо } (\mathbf{w}^{1:2})^T \mathbf{y}^1 = \max_j ((\mathbf{w}^{1:2})^T \mathbf{y}^1); \\ 0 & \text{в інших випадках.} \end{cases}$

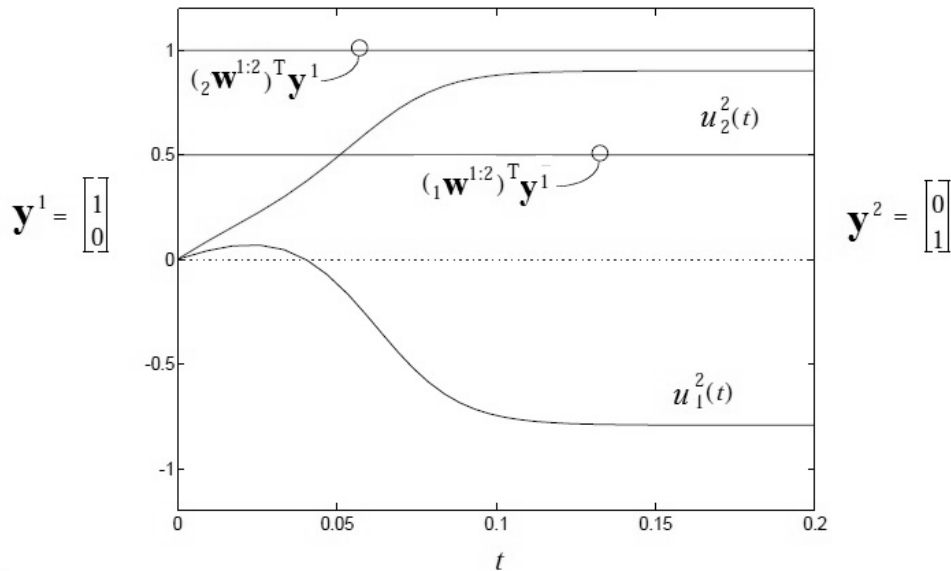


Рис 13.6. Вихід другого шару (до прикл. 13.2)

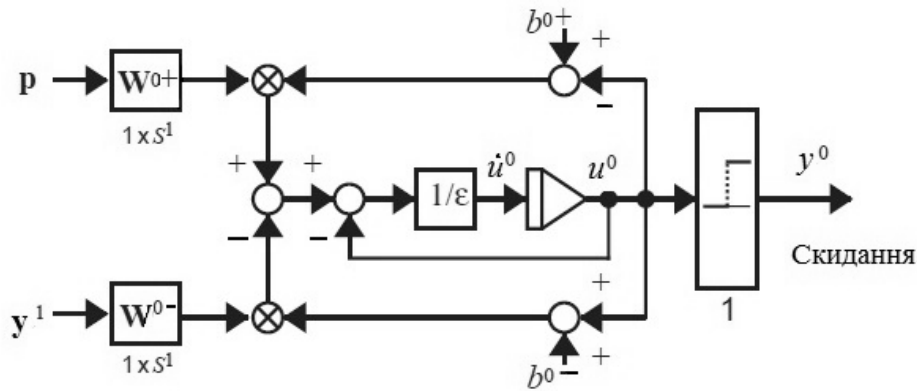
### 13.4. Мережа ART1: підсистема орієнтації

Підсистема орієнтації (рис. 13.7) визначає, чи відповідає очікування  $\mathbf{y}^1$  зв'язку L2–L1 вхідному образу  $\mathbf{p}$ . Якщо відповідності немає, то підсистема орієнтації передає другому шару сигнал скидання, який встановлює довготривалу заборону на функціонування нейрона-переможця, надаючи можливість іншим нейронам стати переможцями у процесі змагання.

Функціонування підсистеми орієнтації описують рівнянням

$$\varepsilon \frac{du^0(t)}{dt} = -u^0(t) + (b^{0+} - u^0(t)) \mathbf{W}^{0+} \mathbf{p} - (u^0(t) + b^{0-}) \mathbf{W}^{0-} \mathbf{y}^1,$$

яке зображує шунтувальну модель зі збуджуванням входом  $\mathbf{p}^+ = \mathbf{W}^{0+} \mathbf{p}$ , де  $\mathbf{W}^{0+} = [\alpha \ \dots \ \alpha]$ . Збуджуваний вхід підсистеми орієнтації можна записати у вигляді  $\mathbf{W}^{0+} \mathbf{p} = [\alpha \ \dots \ \alpha] \mathbf{p} = \alpha \sum_{j=1}^{S^1} p_j = \alpha \|\mathbf{p}\|^2$ , остання рівність виконується, оскільки  $\mathbf{p}$  є двійковим вектором. Пригнічуваний вхід має вигляд  $\mathbf{p}^- = \mathbf{W}^{0-} \mathbf{y}^1$ , де  $\mathbf{W}^{0-} = [\beta \ \dots \ \beta]$ , тому  $\mathbf{W}^{0-} \mathbf{y}^1 = [\beta \ \dots \ \beta] \mathbf{y}^1 = \beta \sum_{j=1}^{S^1} y_j^1(t) = \beta \|\mathbf{y}^1\|^2$ . Підсистема орієнтації використовується кожного разу, коли значення збуджуваного входу перевищує значення пригнічуваного.



$$\epsilon \frac{du^0}{dt} = -u^0 + (b^{0+} - u^0) \mathbf{W}^{0+} \mathbf{p} - (u^0 + b^{0-}) \mathbf{W}^{0-} \mathbf{y}^1$$

Рис. 13.7. Підсистема орієнтації мережі ART1

Розглянемо стійкий стан підсистеми орієнтації:

$$\begin{aligned} 0 &= -u^0 + (b^{0+} - u^0) \alpha \|\mathbf{p}\|^2 - (u^0 + b^{0-}) \beta \|\mathbf{y}^1\|^2 = \\ &= -(1 + \alpha \|\mathbf{p}\|^2 + \beta \|\mathbf{y}^1\|^2) u^0 + b^{0+} \alpha \|\mathbf{p}\|^2 - b^{0-} \beta \|\mathbf{y}^1\|^2, \end{aligned}$$

звідси маємо: 
$$u^0 = \frac{b^{0+} \alpha \|\mathbf{p}\|^2 - b^{0-} \beta \|\mathbf{y}^1\|^2}{1 + \alpha \|\mathbf{p}\|^2 + \beta \|\mathbf{y}^1\|^2}.$$

Нехай  $b^{0+} = b^{0-} = 1$ , тоді  $u^0 > 0$ , якщо  $\alpha \|\mathbf{p}\|^2 - \beta \|\mathbf{y}^1\|^2 > 0$ , або

$$\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} < \frac{\alpha}{\beta} = \rho. \quad (13.6)$$

Одержано умову, за якої виконується скидання другого шару починаючи з моменту, коли  $y^0 = \text{hardlim}^+(u^0)$ . Параметр  $0 < \rho < 1$  називають

параметром пильності, він визначає похибку класифікації (кластеризації): якщо значення  $\rho$  є близьким до одиниці, то виконується скидання (крім випадку, коли  $\mathbf{y}^1$  є близьким до  $\mathbf{p}$ ); якщо значення  $\rho$  є близьким до нуля, то, щоб уникнути скидання,  $\mathbf{y}^1$  не має бути близьким до  $\mathbf{p}$ . Кожного разу, коли другий шар активний, виконується умова (13.4):  $\mathbf{y}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1}$ , тому завжди справджується нерівність  $\|\mathbf{p}\|^2 \geq \|\mathbf{y}^1\|^2$  (рівність справджується, якщо елементи вектора очікування  $\mathbf{w}_j^{2:1}$  і входу  $\mathbf{p}$  дорівнюють одиниці). Якщо між  $\mathbf{p}$  і  $\mathbf{w}_j^{2:1}$  існує невідповідність, підсистема орієнтації виконує скидання. Потрібний для скидання рівень невідповідності визначається параметром пильності  $\rho$ .

**Приклад 13.3.** Розглянемо функціонування підсистеми орієнтації.

Припустимо, що  $\varepsilon = 0,1$ ;  $\alpha = 3$ ;  $\beta = 4$  (тому  $\rho = 0,75$ );  $\mathbf{p} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{y}^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ .

Функціонування підсистеми орієнтації описують рівнянням

$$0,1 \frac{du^0(t)}{dt} = -u^0(t) + (1 - u^0(t))3(p_1 + p_2) - (u^0(t) + 1)4(y_1^1 + y_2^1),$$

або

$$\frac{du^0(t)}{dt} = -110u^0(t) + 20.$$

Результат її функціонування зображено на рис. 13.8.

Сигнал скидання спрямований у напрямку другого шару, поки значення  $u^0(t)$  є додатним. Якщо  $\rho = 0,75$ , то за формулою (13.6) маємо скидання, оскільки  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = 0,5$ . Стійкий стан підсистеми орієнтації визначається як  $y^0 = \begin{cases} 1, & \text{якщо } \frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} < \rho; \\ 0 & \text{в іншому випадку.} \end{cases}$

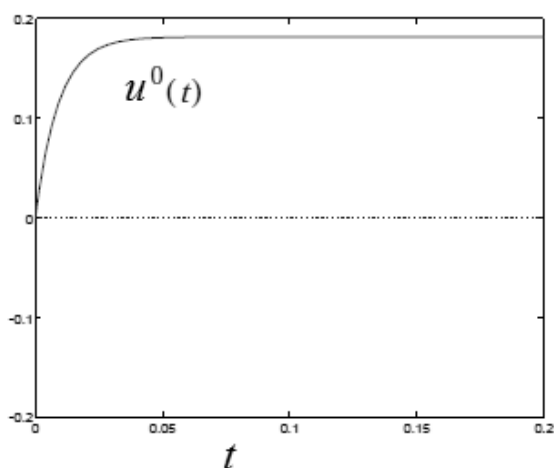


Рис. 13.8. Результат функціонування підсистеми орієнтації

ня, оскільки  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = 0,5$ . Стійкий стан підсистеми орієнтації визначається як  $y^0 = \begin{cases} 1, & \text{якщо } \frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} < \rho; \\ 0 & \text{в іншому випадку.} \end{cases}$

**Правила навчання з'єднання L1–L2 і L2–L1.** Мережа ART1 реалізує два незалежних правила навчання:

- 1) правило навчання Інстар для з'єднання типу L1–L2;

2) правило навчання Оутстар для з'єднання типу L2-L1. Зазначимо, що вагові матриці  $\mathbf{W}^{1:2}$  з'єднання L1-L2 і  $\mathbf{W}^{2:1}$  з'єднання L2-L1 поновлюються одночасно за допомогою підсистеми орієнтації, коли вхідний образ відповідає очікуванню. Такий процес називають *резонансом*.

**Проблема «підмножина / надмножина».** Навчання у з'єднаннях L1-L2 мережі ART1 схоже на навчання мережі Гроссберга з однією істотною відмінністю: у мережі Гроссберга вхідні образи нормалізуються у першому шарі, а тому всі вектори-прототипи мають однакову довжину; у мережі ART1 образи не нормалізуються у першому шарі, тому може виникнути проблема, коли один вектор-прототип є підмножиною іншого.

**Приклад 13.4.** Нехай матриця з'єднання L1-L2 має такий вигляд:

$$\mathbf{W}^{1:2} = \begin{bmatrix} \left( {}_1\mathbf{w}^{1:2} \right)^T \\ \left( {}_2\mathbf{w}^{1:2} \right)^T \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \text{ Оскільки перші два елементи цих векторів}$$

збігаються (дорівнюють одиниці), то вектор-прототип  ${}_1\mathbf{w}^{1:2}$  є підмножиною  ${}_2\mathbf{w}^{1:2}$ . Якщо вихід першого шару  $\mathbf{y}^1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ , то вхід другого шару має

$$\text{вигляд } \mathbf{W}^{1:2} \mathbf{y}^1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}. \text{ Обидва вектори-прототипи мають}$$

однакове значення скалярного добутку на  $\mathbf{y}^1$  (навіть якщо перший вектор-прототип відповідає  $\mathbf{y}^1$ , а другий не відповідає), що називають проблемою «підмножина / надмножина». Одним із шляхів її вирішення є нормалізація вектора-прототипа, що ефективно, коли вектор-прототип має велику кількість ненульових елементів і розмір кожного входу має бути зменшений.

**Приклад 13.5.** Розглянемо приклад 13.4, замінивши вигляд матриці

$$\mathbf{W}^{1:2} \text{ на } \mathbf{W}^{1:2} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}. \text{ Вхідний сигнал другого шару } \mathbf{W}^{1:2} \mathbf{y}^1 =$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{2}{3} \end{bmatrix}. \text{ Перший вектор-прототип має більше значення}$$

скалярного добутку на  $y^1$ , тому перший нейрон другого шару активується. У мережі Гроссберга нормалізація векторів-прототипів відбувається у першому шарі, у мережі ART1 нормалізація векторів-прототипів використовує змагання «до центру / від оточення» у правилі навчання з'єднання L1–L2.

**Правило навчання з'єднання L1–L2.** Правило навчання для матриці  $\mathbf{W}^{1:2}$  має такий вигляд:

$$\frac{d_i \mathbf{w}^{1:2}(t)}{dt} = y_i^2(t) \left[ \left( \mathbf{b}^+ -_i \mathbf{w}^{1:2}(t) \right) \zeta \mathbf{W}^+ \mathbf{y}^1(t) - \left( \mathbf{w}^{1:2}(t) + \mathbf{b}^- \right) \mathbf{W}^- \mathbf{y}^1(t) \right], \quad (13.7)$$

$$\text{де } \mathbf{b}^+ = \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}; \mathbf{b}^- = \begin{bmatrix} 0 \\ \dots \\ 0 \end{bmatrix}; \mathbf{W}^+ = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}; \mathbf{W}^- = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 0 \end{bmatrix}.$$

Формула (13.7) – це змінена форма навчання за правилом Інстар. Якщо  $j$ -й нейрон другого шару є активним, то  $j$ -й рядок  $j\mathbf{w}^{1:2}$  матриці  $\mathbf{W}^{1:2}$  переміщується у напрямку  $y^1$ . Відмінність між формулою (13.7) і стандартним навчанням Інстар полягає в тому, що елементи  $j\mathbf{w}^{1:2}$  змагаються між собою, а тому вектор  $j\mathbf{w}^{1:2}$  є нормалізованим. Формула (13.7) описує шунтувальну модель із вхідним з'єднанням від  $y^1$  типу «до центру / від оточення». Збуджуваний зсув  $b^+ = 1$  і пригнічуваний зсув  $b^- = 0$  гарантують, що елементи  $j\mathbf{w}^{1:2}$  залишаються у межах  $[0; 1]$ .

Щоб перевірити, чи виконує формула (13.7) нормалізацію векторів-прототипів, дослідимо стійкість стану цієї операції. Нехай вихідні сигнали першого та другого шарів залишаються постійними, поки вага досягає стійкого стану. Рівняння для елементів  $j\mathbf{w}^{1:2}$  має вигляд

$$\frac{dw_{ij}^{1:2}(t)}{dt} = y_i^2(t) \left( (1 - w_{ij}^{1:2}(t)) \zeta y_j^1(t) - w_{ij}^{1:2}(t) \sum_{k \neq j} y_k^1(t) \right). \quad (13.8)$$

Якщо припустити, що  $i$ -й нейрон у другому шарі активний (тобто  $y_i^2(t) = 1$ ) і похідні у формулі (13.8) дорівнюють нулю, то одержимо

$$0 = (1 - w_{ij}^{1:2}(t)) \zeta y_j^1(t) - w_{ij}^{1:2}(t) \sum_{k \neq j} y_k^1(t). \quad (13.9)$$

Для визначення стійкого стану  $j\mathbf{w}^{1:2}$  розглянемо два випадки:

1) якщо  $y_j^1 = 1$ , маємо

$$0 = (1 - w_{ij}^{1:2}(t)) \zeta - w_{ij}^{1:2}(t) \left( \|\mathbf{y}^1\|^2 - 1 \right) = - \left( \zeta + \|\mathbf{y}^1\|^2 - 1 \right) w_{ij}^{1:2}(t) + \zeta,$$

звідки

$$w_{ij}^{1,2} = \frac{\zeta}{\zeta + \|\mathbf{y}^1\|^2 - 1}, \quad (13.10)$$

де  $\|\mathbf{y}^1\|^2 = \sum_{k=1}^{S^1} y_k^1$ ,  $\mathbf{y}^1$  – бінарний вектор;

2) якщо  $y_j^1 = 0$ , формула (13.9) має вигляд  $0 = -w_{ij}^{1,2} \|\mathbf{y}^1\|^2$ ,

або

$$w_{ij}^{1,2} = 0. \quad (13.11)$$

Із формул (13.10) і (13.11) маємо:  ${}_j\mathbf{w}^{1,2} = \frac{\zeta \mathbf{y}^1}{\zeta + \|\mathbf{y}^1\|^2 - 1}$ , де параметр  $\zeta > 1$

гарантує, що знаменник ніколи не буде дорівнювати нулю. Отже, вектори-прототипи нормалізовані, що вирішує проблему «підмножина / надмножина». «Нормалізація» не означає, що всі вектори-прототипи мають довжину, яка дорівнює одиниці за формулою Евкліда, натомість рядки матриці  $\mathbf{W}^{1,2}$  мають більшість ненульових входів з невеликими значеннями елементів. У цьому випадку вектори з більшою кількістю ненульових входів можуть мати меншу довжину, ніж вектори з їх меншою кількістю.

**Правила навчання з'єднання L2–L1.** В ART1-мережі з'єднання типу L2–L1 (або матриця  $\mathbf{W}^{2:1}$ ) під час навчання використовує правило Оутстар. Метою з'єднання L2–L1 є повторний виклик відповідного вектора-прототипа (очікування), який можна порівняти й об'єднати у першому шарі із вхідним вектором. Коли очікування і вхідний вектор не збігаються, то у другий шар передається сигнал скидання. Це означає, що можна вибирати новий вектор-прототип. Правило навчання для матриці  $\mathbf{W}^{2:1}$  – це правило Оутстар вигляду

$$\frac{d {}_j\mathbf{w}^{2:1}(t)}{dt} = y_j^2(t)(-{}_j\mathbf{w}^{2:1}(t) + \mathbf{y}^1(t)). \quad (13.12)$$

Якщо  $j$ -й нейрон у другому шарі було активовано (виграв змагання), то  $j$ -й стовпець матриці  $\mathbf{W}^{2:1}$  під час навчання переміщується до вектора  $\mathbf{y}^1$ .

Розглянемо операції одержання стійкого стану в рівнянні (13.12), а саме сценарій швидкого навчання, коли виходи першого і другого шарів залишаються постійними, поки вага перебуває в межах стійкого стану. Нехай  $j$ -й нейрон у другому шарі активований, тобто  $y_j^2 = 1$ . Прирівнявши похідну у формулі (13.12) до нуля, одержимо:  $0 = -{}_j\mathbf{w}^{2:1} + \mathbf{y}^1$ , звідки  ${}_j\mathbf{w}^{2:1} = \mathbf{y}^1$ . Тому  $j$ -й стовпець матриці  $\mathbf{W}^{2:1}$  збігається з виходом першого шару  $\mathbf{y}^1 = \mathbf{p} \cap_j \mathbf{w}^{2:1}$ .

Матриці  $\mathbf{W}^{1:2}$  і  $\mathbf{W}^{2:1}$  оновлюються одночасно: коли  $j$ -й нейрон другого шару активний і відповідність між очікуванням і вхідним вектором достатня (явище резонансу), то  $j$ -й рядок матриці  $\mathbf{W}^{1:2}$  та  $j$ -й стовпець матриці  $\mathbf{W}^{2:1}$  оновлюються. У разі швидкого навчання  $j$ -й стовпець матриці  $\mathbf{W}^{2:1}$  збігається зі значенням  $\mathbf{y}^1$ , тоді як  $j$ -й рядок матриці  $\mathbf{W}^{1:2}$  – з нормалізованим значенням  $\mathbf{y}^1$ .

### 13.5. Мережа ART1: алгоритм навчання

Алгоритм розпочинається з ініціалізації вагових матриць  $\mathbf{W}^{1:2}$  і  $\mathbf{W}^{2:1}$ . Початкові значення елементів матриці  $\mathbf{W}^{2:1}$  вважають рівними одиниці. Таким чином, на початку алгоритму нейрон у другому шарі виграє змагання,

відбувається резонанс, оскільки  $\mathbf{y}^1 = \mathbf{p} \cap_j \mathbf{w}^{2:1} = \mathbf{p}$ , звідки  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = 1 > \rho$ .

Це означає, що будь-який ненавчений стовпець матриці  $\mathbf{W}^{2:1}$  можна об'єднати із вхідним вектором. Оскільки рядки матриці  $\mathbf{W}^{1:2}$  повинні збігатися з нормалізованими стовпцями матриці  $\mathbf{W}^{2:1}$ , то для кожного елемента початкової матриці  $\mathbf{W}^{1:2}$  встановлюється значення  $\frac{\zeta}{(\zeta + S^1 - 1)}$ .

Алгоритм навчання включає такі етапи:

*Крок 1.* До мережі подаються вхідні вектори. На етапі ініціалізації, поки другий шар неактивований, тобто всі  $y_i^2 = 0$ , вихід першого шару  $\mathbf{y}^1 = \mathbf{p}$ .

*Крок 2.* Обчислюють входи другого шару  $\mathbf{W}^{1:2} \mathbf{y}^1$ , і нейрон другого шару з найбільшим входом стає активним:

$$y_i^2 = \begin{cases} 1, & \text{якщо } ({}_i \mathbf{w}^{1:2})^T \mathbf{y}^1 = \max_k \left( ({}_k \mathbf{w}^{1:2})^T \mathbf{y}^1 \right); \\ 0 & \text{в інших випадках.} \end{cases}$$

Якщо перемагає декілька нейронів, то переможцем оголошується нейрон з найменшим значенням індексу (номера).

*Крок 3.* Обчислюють очікування зв'язку L2–L1; якщо  $j$ -й нейрон другого шару активний, одержимо  $\mathbf{W}^{2:1} \mathbf{y}^2 = {}_j \mathbf{w}^{2:1}$ .

*Крок 4.* Якщо другий шар активований, то вихід першого шару пристосовується до очікування:  $\mathbf{y}^1 = \mathbf{p} \cap_j \mathbf{w}^{2:1}$ .

*Крок 5.* Підсистема орієнтації визначає, наскільки вхідний вектор відповідає очікуванню:  $y^0 = \begin{cases} 1, & \text{якщо } \frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} < \rho; \\ 0 & \text{в інших випадках.} \end{cases}$



*Крок 6.* Якщо  $y^0 = 1$ , то встановлюється значення  $y_j^2 = 0$ , яке затримуємо, поки не трапиться відповідний збіг між очікуванням і вхідним вектором (резонанс), тоді повертаємося до *кроку 1*. Якщо  $y^0 = 0$ , переходимо до *кроку 7*.

*Крок 7.* У разі одержання резонансу оновлюється  $j$ -й рядок матриці  $\mathbf{W}^{1:2}$ :

$${}_j \mathbf{w}^{2:1}(t) = \frac{\zeta y^1}{\zeta + \|y^1\|^2 - 1}.$$

*Крок 8.* Оновлюється  $j$ -й стовпець матриці  $\mathbf{W}^{2:1}$ :  ${}_j \mathbf{w}^{2:1} = y^1$ .

*Крок 9.* Вхід видаляється, відновлюються всі пригнічені нейрони другого шару, переходимо до *кроку 1* з новим входом. Вхідні вектори продовжують надходити до мережі, поки вага не припинить змінюватися.

Г. Карпентер і Ст. Гроссберг показали, що ART1-алгоритм завжди формує стабільні кластери для будь-яких множин вхідних образів.

**Інші види мереж ART.** Недолік мережі ART1 полягає в тому, що її можна використовувати тільки для бінарних вхідних векторів. Г. Карпентер і Ст. Гроссберг розробили мережу ART2 (різновид мережі ART1) для обробки аналогових або бінарних образів. Структура мережі ART2 дуже схожа на мережу ART1. У мережі ART2 є декілька підшарів у першому шарі. Ці підшари необхідні, оскільки аналогові вектори відрізняються від бінарних і можуть бути достатньо близькими один до одного. Підшари виконують одночасно нормалізацію та пригнічення шумів, а також визначення вихідних векторів і очікувань, потрібних для підсистеми орієнтації.

У 1991 р. Г. Карпентер, Ст. Гроссберг і Дж. Рейнольдс запропонували мережу ARTMAP, яка, на відміну від всіх інших ART-мереж, є контрольованою. Архітектура ARTMAP складається з двох ART-модулів, які з'єднані асоціативною пам'яттю «інтер-ART». Перший ART-модуль отримує вхідний вектор, у той час як другий ART-модуль отримує бажаний вихідний вектор. Г. Карпентер, Ст. Гроссберг, Н. Маркузон, Дж. Рейнольдс і Д. Росен модифікували архітектуру ARTMAP, включивши до неї нечітку логіку. Нечітку ARTMAP найчастіше використовують, коли вхідні вектори мають шуми.

## Приклади розв'язання задач\*

**Приклад 13.6.** Нехай перший шар ART1-мережі має такі параметри:  $\varepsilon = 0,01$ ;  $b^{1-} = 2$ ;  $b^{1+} = 3$ . Другий шар має два нейрони, вагову матрицю та вхід вигляду  $\mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ ;  $\mathbf{p} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . Перший нейрон другого шару є активним.

**I.** Визначити і побудувати вихід  $u^1$ .

**II.** Перевірити, що отриманий у п. I вихід задовольняє умову стійкого стану:  $\mathbf{p} \cap_1 \mathbf{w}^{2:1} = \mathbf{y}^1$ .

*Розв'язання.* **I.** Оскільки другий шар є активним і перший нейрон другого шару виграє змагання, то рівняння для виходу першого шару має вигляд

$$0,01 \frac{du_1^1}{dt} = -u_1^1 + (2 - u_1^1)(p_{1+11} w^{2:1}) - (u_1^1 + 3) =$$

$$= -u_1^1 + (2 - u_1^1)(1 + 0) - (u_1^1 + 3) = -3u_1^1 - 1;$$

$$0,01 \frac{du_2^1}{dt} = -u_2^1 + (2 - u_2^1)(p_{2+21} w^{2:1}) - (u_2^1 + 3) =$$

$$= -u_2^1 + (2 - u_2^1)(1 + 1) - (u_2^1 + 3) = -4u_2^1 + 1.$$

Спростивши ці рівняння, одержимо  $\frac{du_1^1}{dt} = -300u_1^1 - 100$ ;  $\frac{du_2^1}{dt} = -400u_2^1 + 100$ . Якщо припустити, що обидва нейрони мають нульові початкові умови, то розв'язки будуть такими (рис. 13.9):  $u_1^1 = -\frac{1}{3}(1 - e^{-300t})$ ;  $u_2^1 = -\frac{1}{4}(1 - e^{-400t})$ . Зазначимо, що  $u_1^1(t)$  збігається до від'ємного значення, а  $u_2^1(t)$  – до додатного значення, тому  $y_1^1(t)$  збігається до нуля, а  $y_2^1(t)$  – до одиниці. Це узгоджується з аналізом стійкого стану, оскільки  $\mathbf{p} \cap_1 \mathbf{w}^{2:1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cap \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{y}^1$ .

**Приклад 13.7.** Нехай другий шар ART1-мережі має такі параметри:

$$\varepsilon = 0,1; \mathbf{b}^{2+} = \mathbf{b}^{2-} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}; \mathbf{W}^{1:2} = \begin{bmatrix} (\mathbf{w}^{1:2})^T \\ (\mathbf{w}^{1:2})^T \end{bmatrix} = \begin{bmatrix} 0,5 & 0,5 \\ 1 & 0 \end{bmatrix} \text{ і } f^2(u) = \begin{cases} 10u^2, & \text{якщо } u \geq 0; \\ 0, & \text{якщо } u < 0. \end{cases}$$

---

\* Задачі взято з посібника [14].

Вихід першого шару має такий вигляд:  $\mathbf{y}^1 = [1 \ 0]^T$ .

**I.** Написати рівняння функціонування другого шару та побудувати графіки виходів. Пояснити, чому відбувається збільшення значення зсуву.

**II.** Перевірити правильність стійкого стану для другого шару.

*Розв'язання. I.* Функціонування другого шару відбувається за такими формулами:

$$0,1 \frac{du_1^2}{dt} = -u_1^2(t) + (2 - u_1^2(t)) \left( f^2(u_1^2(t)) + ({}_1\mathbf{w}^{1:2})^T \mathbf{y}^1 \right) - (u_1^2(t) + 2) f^2(u_2^2(t));$$

$$0,1 \frac{du_2^2}{dt} = -u_2^2(t) + (2 - u_2^2(t)) \left( f^2(u_2^2(t)) + ({}_2\mathbf{w}^{1:2})^T \mathbf{y}^1 \right) - (u_2^2(t) + 2) f^2(u_1^2(t)).$$

Вихід другого шару, якщо вектор входу  $\mathbf{y}^1 = [1 \ 0]^T$ , зображено на рис. 13.10.

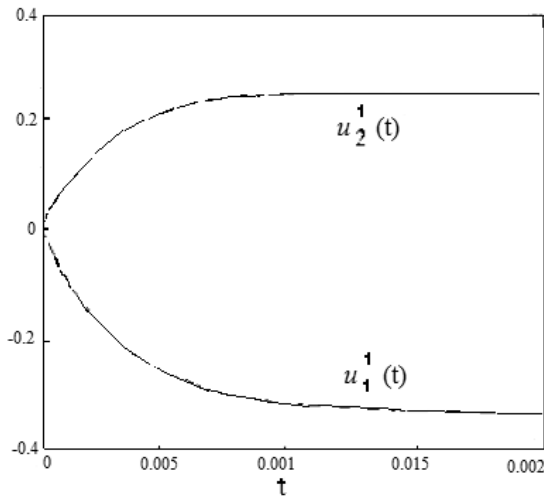


Рис. 13.9. Вихід першого шару  
(до прикл. 13.6)

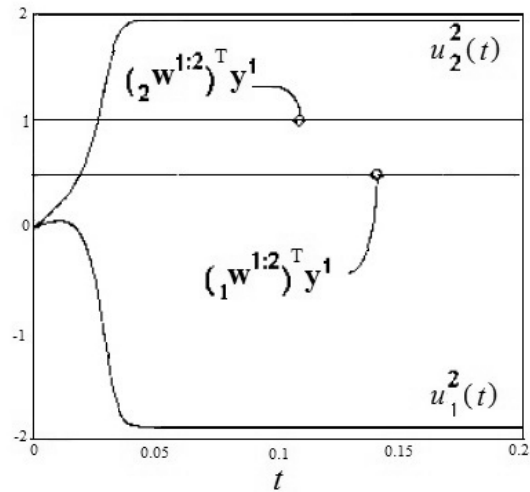


Рис. 13.10. Вихід другого шару  
(до прикл. 13.7)

Оскільки значення добутку  $\mathbf{y}^1$  на другий рядок матриці  $\mathbf{W}^{1:2}$  більше за значення добутку  $\mathbf{y}^1$  на перший рядок вказаної матриці, то другий нейрон викрає змагання.

**II.** У стійкому стані  $u_1^2(t)$  є додатним, а  $u_2^2(t)$  – від'ємним. Стійкий стан для другого шару має вигляд  $\mathbf{y}^2 = [1 \ 0]^T$ , що відповідає очікуваним характеристикам стійкого стану виходу другого шару:

$$\mathbf{y}_i^2 = \begin{cases} 1, & \text{якщо } ({}_i\mathbf{w}^{1:2})^T \mathbf{y}^1 = \max_j \left( ({}_j\mathbf{w}^{1:2})^T \mathbf{y}^1 \right); \\ 0 & \text{в інших випадках.} \end{cases}$$

**Приклад 13.8.** Нехай підсистема орієнтації ART1-мережі має такі параметри:  $\varepsilon = 0,1$ ;  $b^{0+} = b^{0-} = 0,5$ ;  $\alpha = 0,5$ ;  $\beta = 2$  ( $\rho = 0,25$ ) та входи

вигляду  $\mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{y}^1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ .

**I.** Визначити та побудувати графік виходу  $u^0(t)$  підсистеми орієнтації.

**II.** Перевірити, чи виконуються умови стійкого стану.

*Розв'язання.* **I.** Рівняння для визначення виходу  $u^0(t)$  підсистеми орієнтації має вигляд

$$0,1 \frac{du^0(t)}{dt} = -u^0(t) + (0,5 - u^2(t)) \cdot 0,5(p_1 + p_2 + p_3) - (u^0(t) + 0,5) \cdot 2(y_1^1 + y_2^1 + y_3^1),$$

або  $\frac{du^0(t)}{dt} = -65u^0(t) - 12,5$ , звідки вихід має вигляд  $u^0(t) = -0,1923(1 - e^{-65t})$ .

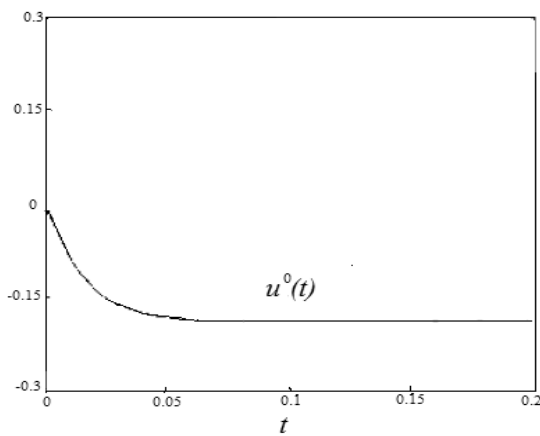


Рис. 13.11. Вихід підсистеми орієнтації

На рис. 13.11 зображено графік виходу підсистеми орієнтації: якщо  $u^0(t)$  є від'ємним, то  $y^0 = \text{hardlim}^+(u^0) = 0$ . Отже, другому шару не передається сигнал скидання.

**II.** Значення стійкого стану підсистеми орієнтації можна зобразити у вигляді

$$y^0 = \begin{cases} 1, & \text{якщо } \frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} < \rho; \\ 0 & \text{в інших випадках.} \end{cases}$$

Маємо  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = \frac{2}{3} < \rho = 0,25$ , тому  $y^0 = 0$ , що узгоджується з результатами, одержаними в п. I.

**Приклад 13.9.** Показати, що правило навчання для з'єднання L2–L1 є еквівалентним правилу навчання Оутстар вигляду

$${}_j \mathbf{w}(q) = {}_j \mathbf{w}(q-1) + \alpha(\mathbf{y}(q) - {}_j \mathbf{w}(q-1))\mathbf{p}_j(q).$$

*Розв'язання.* Правило навчання для з'єднання L2–L1 має вигляд

$$\frac{d {}_j \mathbf{w}^{2:1}(t)}{dt} = y_j^2(t) \left( -{}_j \mathbf{w}^{2:1}(t) + \mathbf{y}^1(t) \right).$$

Апроксимувавши похідну за формулою  $\frac{d {}_j \mathbf{w}^{2:1}(t)}{dt} = \frac{{}_j \mathbf{w}^{2:1}(t + \Delta t) - {}_j \mathbf{w}^{2:1}(t)}{\Delta t}$ , одержимо  ${}_j \mathbf{w}^{2:1}(t + \Delta t) = {}_j \mathbf{w}^{2:1}(t) +$

$\Delta t y_j^2(t) (-{}_j \mathbf{w}^{2:1}(t) + \mathbf{y}^1(t))$ , що еквівалентно правилу навчання Оутстар.

**Приклад 13.10.** Навчити ART1-мережу розпізнавати вектор, використовуючи такі вектори входу:  $\mathbf{p}_1 = [0 \ 1 \ 0]^T$ ;  $\mathbf{p}_2 = [1 \ 0 \ 0]^T$ ;  $\mathbf{p}_3 = [1 \ 1 \ 0]^T$  та параметри  $\zeta = 2$ ;  $\rho = 0,4$ . Вибрати  $S^2 = 3$  (три кластери) та початкові

значення ваги вигляду  $\mathbf{W}^{1:2} = \begin{bmatrix} 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix}$ ;  $\mathbf{W}^{2:1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ .

*Розв'язання.* Використаємо алгоритм роботи ART1-мережі.

*Перший етап. Крок 1.* Обчислити вихід першого шару:  $\mathbf{y}^1 = \mathbf{p}_1 = [0 \ 1 \ 0]^T$ .

*Крок 2.* Обчислити вхід другого шару:  $\mathbf{W}^{1:2} \mathbf{y}^1 = \begin{bmatrix} 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \end{bmatrix}$ . Оскільки всі нейрони мають однакові виходи, то надамо пере-

могу першому нейрону:  $\mathbf{y}^2 = [1 \ 0 \ 0]^T$ .

*Крок 3.* Обчислити очікування для L2–L1:  $\mathbf{W}^{2:1} \mathbf{y}^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} =_1 \mathbf{w}^{2:1}$ .

*Крок 4.* Визначити вихід першого шару таким чином, щоб він відповідав очікуванням для L2–L1:  $\mathbf{y}^1 = \mathbf{p}_1 \cap_1 \mathbf{w}^{2:1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ .

*Крок 5.* Підсистема орієнтації визначає ступінь відповідності між очікуванням і вхідним вектором:  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = \frac{1}{1} > \rho = 0,4$ , тому  $y^0 = 0$  (скидання немає).

*Крок 6.* Якщо  $y^0 = 0$ , перейти на крок 7.

*Крок 7.* Оскільки виникає резонанс, то перший рядок матриці

$$\mathbf{W}^{1:2} \text{ оновлюється за формулою } {}_1\mathbf{w}^{1:2} = \frac{2\mathbf{y}^1}{2 + \|\mathbf{y}^1\|^2 - 1} = \mathbf{y}^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{1:2} =$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix}, \text{ а перший стовпець матриці } \mathbf{W}^{2:1} \text{ – за формулою } {}_1\mathbf{w}^{2:1} =$$

$$= \mathbf{y}^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

*Крок 8.* Видалити  $\mathbf{p}_1$  та повернутися до кроку 1, подавши на вхід вектор  $\mathbf{p}_2$ .

*Другий етап. Крок 1.* Обчислити новий вихід першого шару (другий шар не активовано):  $\mathbf{y}^1 = \mathbf{p}_2 = [1 \ 0 \ 0]^T$ .

*Крок 2.* Обчислити вхід другого шару:  $\mathbf{W}^{1:2}\mathbf{y}^1 =$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0,5 & 0,5 & 0,5 \\ 0,5 & 0,5 & 0,5 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0,5 \\ 0,5 \end{bmatrix}.$$

Оскільки виходи 2-го та 3-го нейронів однакові, то надамо перемогу 2-му нейрону:  $\mathbf{y}^2 = [0 \ 1 \ 0]^T$ .

*Крок 3.* Обчислити очікування для L2–L1:  $\mathbf{W}^{2:1}\mathbf{y}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} =$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = {}_2\mathbf{w}^{2:1}.$$

*Крок 4.* Визначити вихід першого шару таким чином, щоб він відповідав очікуванням для L2–L1:  $\mathbf{y}^1 = \mathbf{p}_2 \cap {}_2\mathbf{w}^{2:1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$

*Крок 5.* Підсистема орієнтації визначає ступінь відповідності між очікуванням і вхідним вектором:  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = \frac{1}{1} > \rho = 0,4$ , тому  $y^0 = 0$  (скидання немає).

*Крок 6.* Якщо  $y^0 = 0$ , перейти на *крок 7*.

Крок 7. Оскільки виникає резонанс, то другий рядок матриці

$$\mathbf{W}^{1:2} \text{ оновлюється за формулою } {}_2\mathbf{w}^{1:2} = \frac{2\mathbf{y}^1}{2 + \|\mathbf{y}^1\|^2 - 1} = \mathbf{y}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{1:2} =$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0,5 & 0,5 & 0,5 \end{bmatrix}, \text{ а другий стовпець матриці } \mathbf{W}^{2:1} \text{ – за формулою } {}_2\mathbf{w}^{2:1} =$$

$$= \mathbf{y}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Крок 8. Повернутися до кроку 1, подавши на вхід вектор  $\mathbf{p}_3$ .

Третій етап. Крок 1. Обчислити вихід першого шару:  $\mathbf{y}^1 = \mathbf{p}_3 = [1 \ 1 \ 0]^T$ .

$$\text{Крок 2. Обчислити вхід другого шару: } \mathbf{W}^{1:2} \mathbf{y}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0,5 & 0,5 & 0,5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \text{ Оскільки виходи 1-го, 2-го та 3-го нейронів однакові, то надамо}$$

перемогу 1-му нейрону:  $\mathbf{y}^2 = [1 \ 0 \ 0]^T$ .

Крок 3. Обчислити очікування для L2–L1:

$$\mathbf{W}^{2:1} \mathbf{y}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = {}_1\mathbf{w}^{2:1}.$$

Крок 4. Визначити вихід першого шару таким чином, щоб він від-

$$\text{повідав очікуванням для L2–L1: } \mathbf{y}^1 = \mathbf{p}_3 \cap {}_1\mathbf{w}^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Крок 5. Підсистема орієнтації визначає ступінь відповідності між

очікуванням і вхідним вектором:  $\frac{\|\mathbf{y}^1\|^2}{\|\mathbf{p}\|^2} = \frac{1}{2} > \rho = 0,4$ , тому  $y^0 = 0$  (скидання немає).

Крок 6. Якщо  $y^0 = 0$ , перейти на крок 7.

Крок 7. Оскільки виникає резонанс, то перший рядок матриці

$$\mathbf{W}^{1:2} \text{ оновлюється за формулою } {}_1\mathbf{w}^{1:2} = \frac{2\mathbf{y}^1}{2 + \|\mathbf{y}^1\|^2 - 1} = \mathbf{y}^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{1:2} =$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0,5 & 0,5 & 0,5 \end{bmatrix}, \text{ а другий стовпець матриці } \mathbf{W}^{2:1} \text{ – за формулою}$$

$${}_2\mathbf{w}^{2:1} = \mathbf{y}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Процес навчання завершено, оскільки під час повторного введення одного з наведених образів вага не зміниться. Образи успішно кластеризовані.

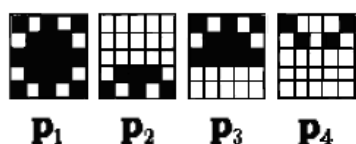


Рис. 13.12. Образи у вигляді векторів-прототипів до прикл. 13.11

**Приклад 13.11.** Навчити ART1-мережу розпізнавати задані образи (у вигляді векторів-прототипів, зображених на рис. 13.12). Вектори подавати на вхід мережі у такій послідовності:  $\mathbf{p}_1 - \mathbf{p}_2 - \mathbf{p}_3 - \mathbf{p}_1 - \mathbf{p}_4$ . Використати параметри  $\zeta = 2$  і  $\rho = 0,6$  та вибрати  $S^2 = 3$  (три кластери).

Процес навчання продовжити, поки вагова матриця не припинить змінюватися.

**Розв'язання.** Розпочнемо з ініціалізації вагової матриці. Початкова матриця  $\mathbf{W}^{2:1}$  має розмірність  $[S^1 \times S^2] = [25 \times 3]$ , а матриця  $\mathbf{W}^{1:2}$  з розмірністю  $[S^2 \times S^1] = [3 \times 25]$  нормалізована, і кожний її елемент становить  $\frac{\zeta}{\zeta + S^1 - 1} = \frac{2}{2 + 25 - 1} = 0,0769$ . Для створення вектора входу пот-

рібно сканувати образ по рядках: кожний чорний квадрат відповідає значенню «1», а білий – значенню «0». Оскільки вхідні образи мають розмірність  $[5 \times 5]$ , то вхідний вектор має розмірність  $[25]$ . Результат навчання (кожний рядок описує один крок алгоритму навчання ART1) зображено на рис. 13.13.

На кожному кроці зірочка позначає точку резонансу – стовпець матриці  $\mathbf{W}^{2:1}$ , який збігається із вхідним вектором-прототипом. Коли відбувається скидання, число біля символу «✓» зображає етап, на якому виникло скидання. Після виконання 10 кроків вага нестабільна. Результат виконання алгоритму можна змінити, змінивши параметр  $\rho$ .



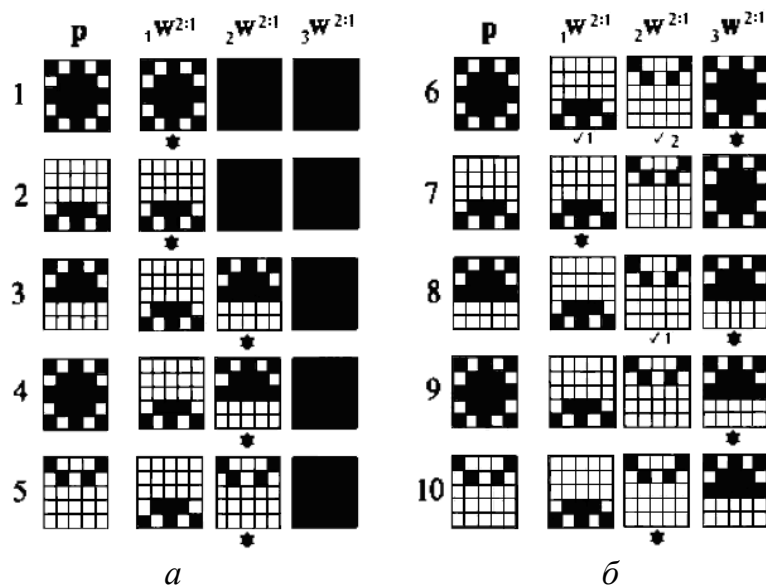


Рис. 13.13. Виконання алгоритму ART1:  
а – перший прохід  $p_1-p_2-p_3-p_1-p_4$ ; б – другий прохід

### Контрольні запитання

1. Опишіть ART1-мережу: перший шар (рівняння функціонування, умову стійкого стану); другий шар (рівняння функціонування).
2. Опишіть підсистему орієнтації ART1-мережі: рівняння функціонування, параметр пильності.
3. Опишіть правила навчання з'єднань L1–L2 і L2–L1 (рівняння функціонування, умову стійкого стану).
4. Опишіть алгоритм навчання ART1-мережі.

### Задачі для самостійного розв'язання

**Задача 13.1.** Нехай перший шар мережі ART1 має параметр  $\varepsilon = 0,02$ , другий шар містить два нейрони, причому другий нейрон є активним.

Вагова матриця та вхід  $\mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ ;  $\mathbf{p} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . Визначити та побудувати

графік виходу  $u^1$ , якщо:

I.  $b^{1+} = 2$  і  $b^{1-} = 3$ .

II.  $b^{1+} = 4$  і  $b^{1-} = 5$ .

III.  $b^{1+} = 4$  і  $b^{1-} = 4$ .

IV. Перевірити, чи всі відповіді п. I, II, III задовольняють умову виходу стійкого стану. Пояснити невідповідності.

V. Перевірити правильність відповідей п. I, II, III написавши *m*-файл для моделювання функціонування першого шару ART1-мережі.

**Задача 13.2.** Нехай другий шар ART1-мережі має такі параметри:

$$\varepsilon = 0,1; \mathbf{W}^{2:1} = \begin{bmatrix} (\mathbf{w}^{1:2})^T \\ (\mathbf{w}^{1:2})^T \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ 1 & 0 \end{bmatrix} \text{ та } f^2(u) = \begin{cases} 10u^2, & \text{якщо } u \geq 0; \\ 0, & \text{якщо } u < 0. \end{cases} \text{ При-}$$

пустимо, що вихід першого шару  $\mathbf{y}^1 = [1 \ 1]^T$ .

**I.** Написати рівняння функціонування другого шару, змодельовати та побудувати вихід для таких векторів зсуву:  $\mathbf{b}^{2+} = \mathbf{b}^{2-} = [2 \ 2]^T$ .

**II.** Виконати п. I для векторів зсуву:  $\mathbf{b}^{2+} = [3 \ 3]^T$ ;  $\mathbf{b}^{2-} = [0 \ 0]^T$ .

**III.** Перевірити, чи всі відповіді п. I, II задовольняють умову виходу стійкого стану. Пояснити невідповідності.

**Задача 13.3.** Підсистема орієнтації мережі ART1 має такі параметри:

$$\varepsilon = 0,01; b^{0-} = b^{0+} = 2, \text{ та входи: } \mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; \mathbf{y}^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

**I.** Визначити та побудувати графік виходу  $u^0(t)$  підсистеми орієнтації за:

а)  $\alpha = 0,5$ ;  $\beta = 4$  ( $\rho = 0,125$ );

б)  $\alpha = 0,5$ ;  $\beta = 2$  ( $\rho = 0,25$ ).

**II.** Перевірити, чи задовольняють результати, одержані в п. I (а, б), умову стійкого стану.

**III.** Перевірити правильність відповідей, одержаних у п. I (а, б), написавши *m*-файл для моделювання функціонування підсистеми орієнтації.

**Задача 13.4.** Навчити ART1-мережу розпізнавати такі вхідні векто-

$$\text{ри: } \mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}; \mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}; \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \text{ Використати значення } \zeta = 2$$

та  $S^2 = 3$  (три кластери).

**I.** Навчати мережу до моменту збіжності вагової матриці, якщо  $\rho = 0,3$ .

**II.** Повторити п. I, якщо  $\rho = 0,6$ ;  $\rho = 0,9$ .

**Задача 13.5.** Алгоритм ART1 можна змінити, додавши один нейрон до другого шару, якщо немає відповідності між існуючим вектором-прототипом і вхідним вектором. При цьому створюється новий рядок матриці  $\mathbf{W}^{1:2}$  та новий стовпець матриці  $\mathbf{W}^{2:1}$ . Описати реалізацію цього процесу.

**Задача 13.6.** Написати *m*-файл системи MatLab для застосування модифікованого алгоритму ART1, описаного в задачі 13.5, для навчання ART1-мережі розпізнавати задані нижче образи у вигляді векторів-прототипів (рис. 13.12). Вводити вектори в такій послідовності:  $\mathbf{p}_1 - \mathbf{p}_2 - \mathbf{p}_3 - \mathbf{p}_1 - \mathbf{p}_4$ , використати значення  $\zeta = 2$ ;  $\rho = 0,9$  і  $S^2 = 3$  (три кластери). Навчання виконувати до моменту збіжності вагової матриці.

**Задача 13.7.** Навчити ART1-мережу розпізнавати такі вектори входу:  $\mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ ;  $\mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ , використовуючи параметри:  $\zeta = 2$ ;  $\rho = 0,6$ . Вибрати  $S^2 = 3$  (три кластери) та початкові значення ваги вигляду  $\mathbf{W}^{1:2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0,5 & 0,5 & 0,5 \end{bmatrix}$ ;  $\mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ .

## Розділ 14. МЕРЕЖА ХОПФІЛДА

### 14.1. Нелінійні динамічні системи у вигляді рекурентних мереж

У поведінці динамічних систем розрізняють сталий та перехідний рух. *Сталим* називають рух, який виник у системі після деякого проміжку часу і має властивості стійкості та повторюваності. *Перехідним* називають несталий рух, у процесі якого система з деякого початкового стану наближається до сталого руху.

Завданням дослідження математичних моделей динамічних систем є *локалізація сталих рухів*, тобто аналітичне оцінювання у фазовому просторі (багатовимірному просторі змінних динамічної системи) компактної множини, яка містить усі траєкторії сталих рухів. Під час вирішення цього завдання суттєву роль відіграє теорема Ла–Салля про локалізацію траєкторій сталих рухів за допомогою узагальнених функцій Ляпунова (ФЛ), похідна яких недодатна (дод. Д). Множину точок фазового простору, в яких похідна ФЛ дорівнює нулю, називають *нейтральною*. За теоремою Ла–Салля нейтральна множина притягує до себе будь-який рух із передкомпактних додатних напівтраєкторій і містить усі траєкторії сталих рухів. Про стійкість (збіжність) алгоритмів найшвидшого спуску та алгоритму LMS йшлося відповідно в розд. 5 та 6, а про неперервні рекурентні мережі Гроссберга – у розд. 12.

Мережі, які мають зворотний зв'язок виходів із входами, називають *рекурентними*. Рекурентні мережі більш потужні, ніж мережі з прямим зв'язком, оскільки вони здатні розпізнати як часові, так і просторові образи. Поведінка рекурентних мереж набагато складніша, ніж мереж із прямим зв'язком: вихід мережі із прямим зв'язком є функцією, яка залежить тільки від входу мережі; вихід рекурентних мереж є функцією від часу. Для заданих вхідних параметрів мережі та початкових значень вихідних параметрів вихід мережі може збігатися або здійснювати коливальні рухи (прямувати до нескінченності або поводитися хаотично). Розглянемо рекурентні мережі (рис. 14.1), які можна описати нелінійним диференціальним рівнянням вигляду  $\frac{dy(t)}{dt} = g(y(t), p(t), t)$ , де  $p(t)$  – вхід мережі;  $y(t)$  – вихід мережі;  $t$  – час. У конкретний момент часу  $t$  спостерігається стан системи  $y(t)$ , зображений точкою в  $n$ -вимірному просторі станів (наприклад,  $\mathbb{R}^n$ ).

Визначимо поведінку системи у стійкому стані (а саме, як виходи мережі збігаються в точці стійкої рівноваги). Нелінійна система може

мати багато точок рівноваги. Для деяких мереж точки стійкості мають вигляд збережених векторів-прототипів образів. Визначимо, де розміщені точки стійкості системи та які початкові умови дозволяють отримати на виході ці точки. У дод. Д розглянуто поняття «стійкість» та визначено точку (або траєкторію), в якій певні групи нелінійних рівнянь збігаються до своїх розв'язків. Теореми, описані в цьому додатку, є важливим інструментом для аналізу рекурентних НМ (мереж Гроссберга і Хопфілда).

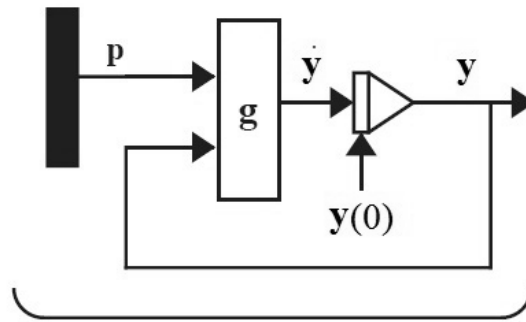


Рис. 14.1. Нелінійна рекурентна мережа:  $\frac{dy(t)}{dt} = g(y(t), p(t), t)$

На початку професійної діяльності Дж. Хопфілд вивчав взаємодію між світлом і твердими тілами, а пізніше зосередив зусилля на дослідженні механізму перенесення електронів між біологічними молекулами. У 1982–1984 рр. Дж. Хопфілд написав дві статті, в яких висвітлив ідеї, основані на розробках інших дослідників, серед яких нейронна модель Маккаллока–Пітса, адитивна модель Гроссберга, лінійний асоціатор Андерсона і Кохонена. Ці статті Хопфілда інтегрували важливі ідеї у вигляді чіткого математичного аналізу з використанням теорії стійкості Ляпунова. Дж. Хопфілд установив аналогію між своєю мережею та ізінговським методом магнітних матеріалів, який використовується у статистичній фізиці. На початку 1980-х років це спонукало багатьох вчених зосередити зусилля на дослідженнях з теорії НМ (дослідження почали відроджуватися).

Розглянемо модель Хопфілда дискретного часу.

## 14.2. Мережа Хопфілда дискретного часу

Рекурентну мережу Хопфілда зображено на рис. 14.2. У цій мережі нейрони ініціюються вхідним вектором, потім мережа рекурсивно функціонує, поки вихід не буде збігатися до однієї точки. Якщо мережа функціонує правильно, то вона одержує на виході результат, який є одним із векторів-прототипів образів. Функціонування мережі Хопфілда описується рівнянням  $y(0) = p$ ,  $y(t+1) = \text{satlins}(Wy(t) + b)$ , де *satlins* – лінійна ФА, визначена на інтервалі  $[-1; 1]$ , яка дорівнює «1» для входів, більших за «1», і «-1» – для входів, менших за «-1». Проілюструємо функціонування мережі Хопфілда, визначивши вагову матрицю та вектор зсуву як

$$W = \begin{bmatrix} 0,2 & 0 & 0 \\ 0 & 1,2 & 0 \\ 0 & 0 & 0,2 \end{bmatrix}; b = \begin{bmatrix} 0,9 \\ 0 \\ -0,9 \end{bmatrix}. \text{ Це не єдиний можливий варіант вибору}$$

$W$  і  $b$ , їх визначення для мережі Хопфілда є складною процедурою.

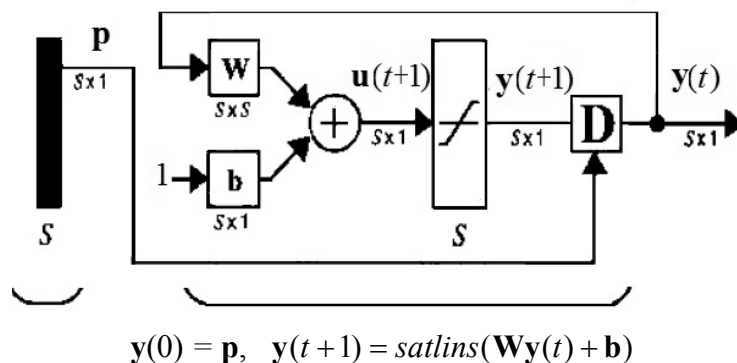


Рис. 14.2. Мережа Хопфілда дискретного часу

Нехай вектор-прототип апельсина має вигляд  $p_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$ , а вектор-

прототип яблука –  $p_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$ . Необхідно, щоб вихід мережі збігався до

однієї точки: до вектора-прототипа апельсина  $p_1$  або вектора-прототипа яблука  $p_2$ . В обох випадках перший та третій елементи дорівнюють «1» та «-1» відповідно, різняться лише другі елементи векторів-прототипів. Функціонування мережі Хопфілда має вигляд  $y_1(t+1) = \text{satlins}(0,2y_1(t) + 0,9)$ ;  $y_2(t+1) = \text{satlins}(1,2y_2(t))$ ;  $y_3(t+1) = \text{satlins}(0,2y_3(t) - 0,9)$ . Незважаючи на початкове значення  $y^1(0)$ , перший елемент буде збільшуватися, поки він

не стане дорівнювати «1», а третій елемент буде зменшуватися, поки він не стане дорівнювати «-1». Оскільки другий елемент множать на число, яке більше «1», то, якщо він спочатку був від'ємним, в остаточному підсумку він буде дорівнювати «-1», а якщо його значення було додатним, то «1».

Нехай апельсин має еліптичну форму, яка задається вхідним вектором вигляду  $\mathbf{p} = [-1 \quad -1 \quad -1]^T$ . Тоді вихід мережі Хопфілда для перших трьох ітерацій  $\mathbf{y}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$ ;  $\mathbf{y}(1) = \begin{bmatrix} 0,7 \\ -1 \\ -1 \end{bmatrix}$ ;  $\mathbf{y}(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$ ;  $\mathbf{y}(3) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$

і збігається з вектором-прототипом апельсина.

Розглянемо модель Хопфілда неперервного часу [14] (з неперервними змінними станів), до якої застосуємо теорію стійкості Ляпунова та теорему про інваріантність Ла-Салля (дод. Д). Наприкінці проілюструємо можливість використання правила Хебба під час проектування мережі Хопфілда як асоціативної пам'яті.

### 14.3. Модель Хопфілда неперервного часу

Хопфілд запропонував модель у вигляді електричної схеми (рис. 14.3). Кожний нейрон мережі має вигляд підсилювача, пов'язаного з резисторами та конденсаторами. Нейрони мають дві множини входів:

1) множину електричних струмів  $I_1, I_2, \dots$  у вигляді констант зовнішніх входів;

2) множину, яка складається зі з'єднань зворотного зв'язку. Наприклад, другий вихід  $y_2$  живиться від резистора  $R_{s2}$ , який, у свою чергу, з'єднаний із входом підсилювача  $S$ . Резистори використовуються тільки додатні, але від'ємний вхід нейрона можна отримати, вибравши інверсний вихід певного підсилювача (на рис. 14.3 інверсний вихід першого підсилювача з'єднаний із входом другого підсилювача через резистор). Нагадаємо, що резистор – це пасивний елемент електричного ланцюга, який в ідеалі характеризується тільки опором електричному струму (тобто для ідеального резистора у будь-який момент часу повинен виконуватися закон Ома – миттєве значення напруги на резисторі пропорційне струму, що проходить через нього:  $U(t) = RI(t)$ ).

Запишемо рівняння функціонування моделі Хопфілда, отримане за допомогою правила Кірхгофа:

$$C \frac{du_i(t)}{dt} = \sum_{j=1}^S T_{ij} y_j(t) - \frac{u_i(t)}{R_i} + I_i, \quad (14.1)$$

де  $u_i$  – вхідна напруга  $i$ -го підсилювача;  $y_i$  – вихідна напруга  $i$ -го підси-

лювача;  $C$  – вхідна ємність підсилювача;  $I_i$  – фіксований вхідний струм  $i$ -го підсилювача. Крім того, маємо  $|T_{ij}| = \frac{1}{R_{ij}}$ ;  $\frac{1}{R_i} = \frac{1}{\rho} + \sum_{j=1}^S \frac{1}{R_{i,j}}$ ;  $u_i = f^{-1}(y_i)$  (або  $y_i = f(u_i)$ ), де  $f(u)$  – характеристика підсилювача). Припустимо, що схема є симетричною, тому  $T_{ij} = T_{ji}$ . Функція активації підсилювача  $y_i = f(u_i)$  зазвичай має вигляд сигмоїдної функції. Помноживши обидві сторони рівняння (14.1) на  $R_i$ , отримаємо

$$R_i C \frac{du_i(t)}{dt} = \sum_{j=1}^S R_i T_{ij} y_j(t) - u_i(t) + R_i I_i. \quad (14.2)$$

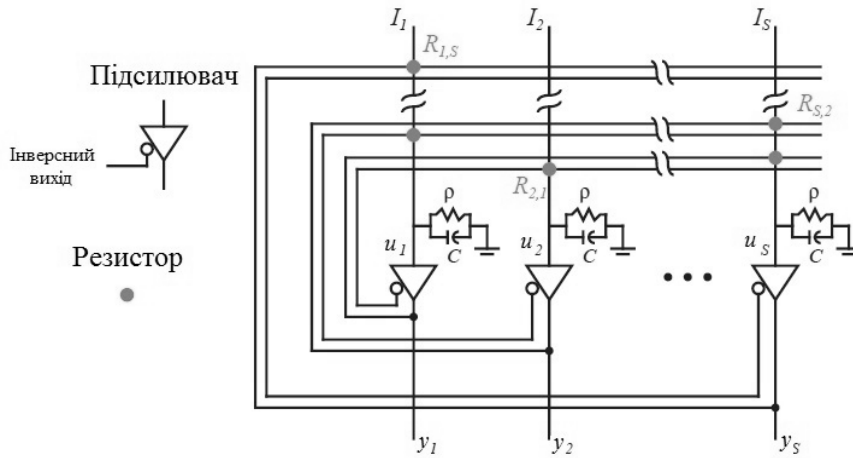
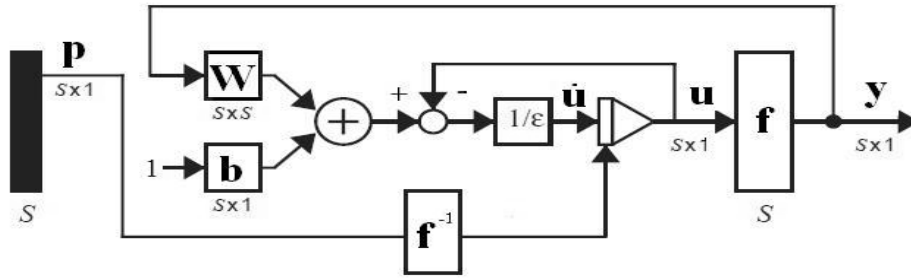


Рис. 14.3. Модель Хопфілда



$$\mathbf{u}(0) = f^{-1}(\mathbf{p}) \quad (\mathbf{y}(0) = \mathbf{p}); \quad \epsilon \frac{d\mathbf{u}(t)}{dt} = -\mathbf{u}(t) + \mathbf{W}f(\mathbf{u}(t)) + \mathbf{b}$$

Рис. 14.4. Мережа Хопфілда неперервного часу

Якщо визначити  $\epsilon = R_i C$ ;  $w_{ij} = R_i T_{ij}$  і  $b_i = R_i I_i$ , то рівняння (14.2) можна переписати у вигляді  $\epsilon \frac{du_i(t)}{dt} = -u_i(t) + \sum_{j=1}^S w_{ij} y_j(t) + b_i$ , або у векторній формі:

$$\epsilon \frac{d\mathbf{u}(t)}{dt} = -\mathbf{u}(t) + \mathbf{W}\mathbf{y}(t) + \mathbf{b}; \quad \mathbf{y}(t) = f(\mathbf{u}(t)). \quad (14.3)$$



У мережі Хопфілда неперервного часу, зображеній на рис. 14.4, вхідний вектор  $\mathbf{p}$  визначає початковий вихід мережі. Таку форму мережі Хопфілда використовують для мереж асоціативної пам'яті.

#### 14.4. Функція Ляпунова та її особливості

Хопфілд застосував теорію стійкості Ляпунова до аналізу рекурентної НМ. Продемонструємо, як теорему Ла–Салля про інваріантність можна використати до мережі Хопфілда. Спочатку визначимо ФЛ – Хопфілд запропонував такий її вигляд:

$$V(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} + \sum_{i=1}^S \left( \int_0^{y_i} f^{-1}(x)dx \right) - \mathbf{b}^T\mathbf{y}. \quad (14.4)$$

Зазначимо, що перший і третій елементи ФЛ формують квадратичну функцію. Визначимо особливості ФЛ (14.4).

Щоб використати теорему Ла–Салля, необхідно обчислити похідну від ФЛ  $V(\mathbf{y})$ . Розглянемо окремо кожний із трьох елементів функції  $V(\mathbf{y})$ . Відповідно до рівняння  $\nabla(\mathbf{x}^T\mathbf{Q}\mathbf{x}) = 2\mathbf{Q}\mathbf{x}$  (для симетричної матриці  $\mathbf{Q}$ ) похідна першого елемента має вигляд

$$\frac{d}{dt} \left( -\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} \right) = -\frac{1}{2}\nabla[\mathbf{y}^T\mathbf{W}\mathbf{y}]^T \frac{d\mathbf{y}}{dt} = -[\mathbf{W}\mathbf{y}]^T \frac{d\mathbf{y}}{dt} = -\mathbf{y}^T\mathbf{W} \frac{d\mathbf{y}}{dt}.$$

Другий елемент функції  $V(\mathbf{y})$  складається із суми інтегралів, розглянемо один із них:  $\frac{d}{dt} \left( \int_0^{y_i} f^{-1}(x)dx \right) = \frac{d}{dy_i} \left( \int_0^{y_i} f^{-1}(x)dx \right) \frac{dy_i}{dt} = f^{-1}(y_i) \frac{dy_i}{dt} = u_i \frac{dy_i}{dt}$ . Повна похідна другого елемента функції  $V(\mathbf{y})$  має такий вигляд:

$$\frac{d}{dt} \left[ \sum_{i=1}^S \left( \int_0^{y_i} f^{-1}(x)dx \right) \right] = \mathbf{u}^T \frac{d\mathbf{y}}{dt}.$$

Використавши рівняння  $\nabla(\mathbf{h}^T\mathbf{x}) = \nabla(\mathbf{x}^T\mathbf{h}) = \mathbf{h}$ , можна визначити похідну третього елемента функції  $V(\mathbf{y})$ :  $\frac{d}{dt}(-\mathbf{b}^T\mathbf{y}) = -\nabla[\mathbf{b}^T\mathbf{y}]^T \frac{d\mathbf{y}}{dt} = -\mathbf{b}^T \frac{d\mathbf{y}}{dt}$ .

Отже, повна похідна ФЛ  $V(\mathbf{y})$  має такий вигляд:

$$\frac{d}{dt}V(\mathbf{y}) = -\mathbf{y}^T\mathbf{W} \frac{d\mathbf{y}}{dt} + \mathbf{u}^T \frac{d\mathbf{y}}{dt} - \mathbf{b}^T \frac{d\mathbf{y}}{dt} = [-\mathbf{y}^T\mathbf{W} + \mathbf{u}^T - \mathbf{b}^T] \frac{d\mathbf{y}}{dt}. \quad (14.5)$$

Порівнявши вираз (14.5) з рівнянням (14.3), отримаємо:  $-\mathbf{y}^T\mathbf{W} + \mathbf{u}^T - \mathbf{b}^T = -\varepsilon \left[ \frac{d\mathbf{u}(t)}{dt} \right]^T$ , що дозволяє переписати рівняння (14.5) у вигляді

$$\frac{d}{dt}V(\mathbf{y}) = -\varepsilon \left[ \frac{d\mathbf{u}(t)}{dt} \right]^T \frac{d\mathbf{y}}{dt} = -\varepsilon \sum_{i=1}^S \left( \frac{du_i}{dt} \right) \left( \frac{dy_i}{dt} \right). \quad (14.6)$$

Враховуючи, що  $u_i = f^{-1}(y_i)$ , похідну  $\frac{du_i}{dt}$  можна записати таким чином:  $\frac{du_i}{dt} = \frac{d}{dt}[f^{-1}(y_i)] = \frac{d}{dy_i}[f^{-1}(y_i)] \frac{dy_i}{dt}$ . Тоді рівняння (14.6) набуде вигляду

$$\frac{d}{dt}V(\mathbf{y}) = -\varepsilon \sum_{i=1}^S \left( \frac{du_i}{dt} \right) \left( \frac{dy_i}{dt} \right) = -\varepsilon \sum_{i=1}^S \left( \frac{d}{dy_i}[f^{-1}(y_i)] \right) \left( \frac{dy_i}{dt} \right)^2. \quad (14.7)$$

Припустивши, що  $f^{-1}(y_i)$  є зростаючою функцією (як для функціонуючого підсилювача), маємо  $\frac{d}{dy_i}[f^{-1}(y_i)] > 0$ , звідки, враховуючи рівняння (14.7), одержимо:

$$\frac{d}{dt}V(\mathbf{y}) \leq 0. \quad (14.8)$$

Отже, якщо  $f^{-1}(y_i)$  – зростаюча функція, то  $\frac{dV(\mathbf{y})}{dt}$  – від’ємно напіввизначена функція. Функція  $V(\mathbf{y})$  справді є ФЛ.

**Інваріантна множина.** Застосуємо теорему про інваріантність Ла-Салля для визначення точок рівноваги мережі Хопфілда. Спочатку визначимо множину  $Z = \left\{ \mathbf{y} : \frac{dV(\mathbf{y})}{dt} = 0, \mathbf{y} \in \bar{G} \right\}$ , що містить всі точки, в яких похідна ФЛ дорівнює нулю. Припустимо, що  $G = \mathfrak{R}^S$ . Із рівняння (14.7) видно, що  $\frac{dV(\mathbf{y})}{dt} = 0$ , якщо похідні всіх виходів нейрона дорівнюють нулю, тобто:

$$\frac{dy}{dt} = 0. \quad (14.9)$$

У цьому випадку мережа є стійкою. Таким чином, точки, в яких енергія системи не змінюється, є точками стійкості мережі. Це означає, що множина  $L$  – найбільша інваріантна підмножина  $Z$  збігається із  $Z$ , тобто  $L = Z$ , і всі точки із  $Z$  є потенційними атракторами.

**Приклад 14.1.** Розглянемо приклад, наведений у статті Дж. Хопфілда, у вигляді системи, яка має ФА підсилювача вигляду  $y = f(u) = \frac{2}{\pi} \tan^{-1} \left( \frac{\gamma \pi u}{2} \right)$ , звідки  $u = \frac{2}{\gamma \pi} \tan \left( \frac{\pi}{2} y \right)$ . Візьмемо два підсилювача,

з'єднані таким чином, що вихід одного підключений до входу іншого через резистор, причому  $R_{12} = R_{21} = 1$  і  $T_{12} = T_{21} = 1$ . Одержимо вагову матрицю  $\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ . Якщо вхідна ємність підсилювача теж дорівнює одиниці, то маємо  $\varepsilon = R_i C = 1$ . Нехай  $\gamma = 1,4$  і  $I_1 = I_2 = 0$ , звідки  $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ .

Розглянемо ФЛ вигляду

$$V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} + \sum_{i=1}^S \left( \int_0^{y_i} f^{-1}(x) dx \right) - \mathbf{b}^T \mathbf{y}. \quad (14.10)$$

Перший елемент ФЛ (14.10)

$$-\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} = -\frac{1}{2} \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -y_1 y_2.$$

Третій елемент дорівнює нулю, оскільки зсув дорівнює нулю, а  $i$ -та частина другого елемента має вигляд

$$\begin{aligned} \int_0^{y_i} f^{-1}(x) dx &= \frac{2}{\gamma\pi} \int_0^{y_i} \tan\left(\frac{\pi}{2}x\right) dx = \frac{2}{\gamma\pi} \left[ -\log\left[\cos\left(\frac{\pi}{2}x\right)\right] \frac{2}{\pi} \right]_0^{y_i} = \\ &= -\frac{4}{\gamma\pi^2} \log\left[\cos\left(\frac{\pi}{2}y_i\right)\right]. \end{aligned}$$

Підставивши всі три елементи в рівняння (14.10), отримаємо ФЛ вигляду  $V(\mathbf{y}) = -y_1 y_2 - \frac{4}{1,4\pi^2} \left[ \log\left(\cos\left(\frac{\pi}{2}y_1\right)\right) + \log\left(\cos\left(\frac{\pi}{2}y_2\right)\right) \right]$ . Записавши рівняння мережі (14.3) за  $\varepsilon = 1$  і  $\mathbf{b} = 0$ , отримаємо:

$$\frac{d\mathbf{u}}{dt} = -\mathbf{u} + \mathbf{W}f(\mathbf{u}) = -\mathbf{u} + \mathbf{W}\mathbf{y}. \quad (14.11)$$

Підставивши у (14.11) значення вагової матриці  $\mathbf{W}$ , одержимо систему рівнянь  $\frac{du_1}{dt} = y_2 - u_1$ ;  $\frac{du_2}{dt} = y_1 - u_2$ . Отже, виходи нейрона мають вигляд  $y_1 = \frac{2}{\pi} \tan^{-1}\left(\frac{1,4\pi}{2}u_1\right)$ ;  $y_2 = \frac{2}{\pi} \tan^{-1}\left(\frac{1,4\pi}{2}u_2\right)$ .

Розглянемо поведінку мережі Хопфілда: на рис. 14.5 зображено контурний графік ФЛ (контури позначають постійні значення цієї функції) та приклади траєкторій функціонування мережі. Мережа має дві точки-атрактори (притягання) відповідно в нижньому лівому та у верхньому правому кутах. Розпочавши роботу з верхнього лівого кута, система прямує до стійкої точки в нижньому лівому куті (рис. 14.5, а). Якщо для мережі визначити початко-

ву умову  $y(0)$  на діагональній лінії, проведеній з верхнього лівого кута до нижнього правого кута, то розв'язок прямує до початку координат (рис. 14.5, б). Початкові умови, які не потрапляють на цю лінію, будуть прямувати до однієї з точок притягання у нижньому лівому або правому верхньому куті. Початок координат є сідловою точкою функції Ляпунова, а не точкою локального мінімуму. На рис. 14.6, а зображено графіки залежності двох виходів нейрона від часу, а на рис. 14.6, б – графік залежності ФЛ від часу (з наближенням до точок рівноваги значення ФЛ зменшується).

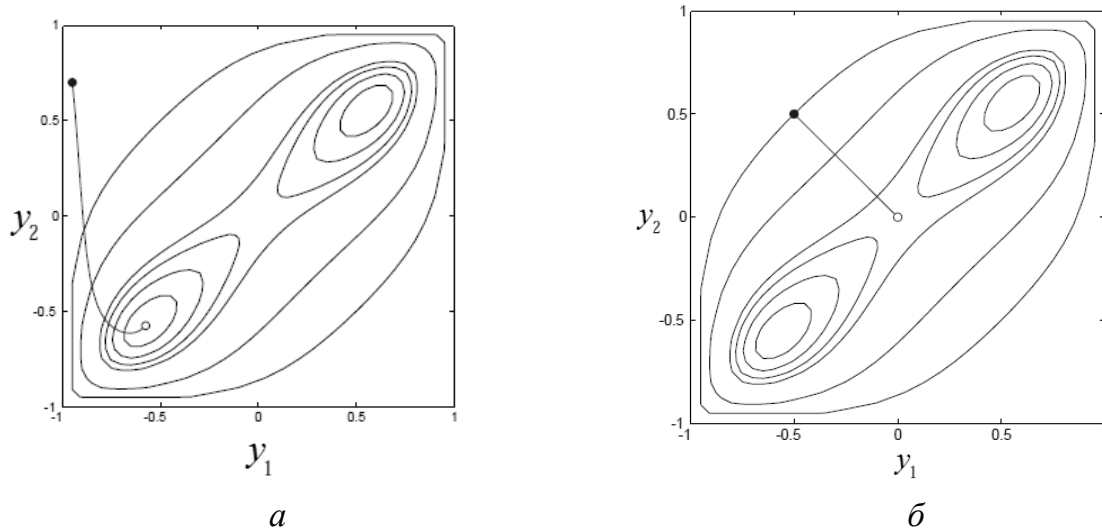


Рис. 14.5. Контурні графіки ФЛ та приклади траєкторій функціонування мережі: збіжність у точці в нижньому лівому куті (а) та в сідловій точці (б)

**Точки-атрактори мережі Хопфілда.** У прикл. 14.1 точки притягання мережі Хопфілда були точками спокою ФЛ. Покажемо, що це виконується і в загальному випадку. Нагадаємо, що потенційні точки притягання мережі Хопфілда задовольняють умову

$$\frac{dy}{dt} = 0. \quad (14.12)$$

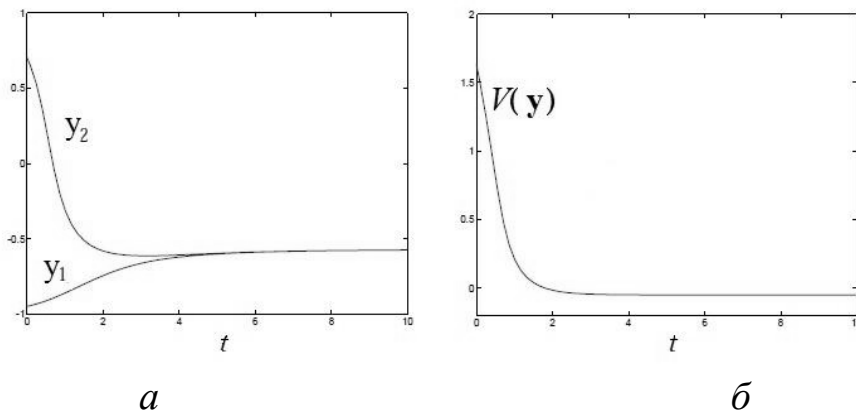


Рис. 14.6. Графіки залежності виходів мережі Хопфілда (а) та ФЛ (б) від часу

Покажемо, як ці точки пов'язані з точками мінімуму ФЛ  $V(\mathbf{y})$ . Раніше було показано, що точка мінімуму  $x^*$  функції  $F(x)$  є стаціонарною точкою (спокою), тобто  $\nabla F(x)|_{x=x^*} = 0$ . Точки спокою функції

$V(\mathbf{y})$  задовольняють умову  $\nabla V(\mathbf{y}) = \left[ \frac{\partial V}{\partial y_1} \quad \dots \quad \frac{\partial V}{\partial y_s} \right]^T = 0$ , де

$V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} + \sum_{i=1}^S \left( \int_0^{y_i} f^{-1}(x) dx \right) - \mathbf{b}^T \mathbf{y}$ . Якщо виконати алгоритм, який

використовувався під час отримання рівняння (14.5), то одержимо такий вираз для градієнта ФЛ:  $\nabla V(\mathbf{y}) = -\mathbf{W} \mathbf{y} + \mathbf{u} - \mathbf{b} = -\varepsilon \frac{d\mathbf{u}(t)}{dt}$ . Отже,  $i$ -й елемент градієнта має вигляд

$$\frac{\partial}{\partial y_i} V(\mathbf{y}) = -\varepsilon \frac{du_i}{dt} = -\varepsilon \frac{d}{dt} [f^{-1}(y_i)] = -\varepsilon \frac{d}{dy_i} [f^{-1}(y_i)] \frac{dy_i}{dt}. \quad (14.13)$$

Зазначимо, що функція  $f^{-1}(\mathbf{y})$  є лінійною, тому рівняння (14.13) передбачає, що:  $\frac{d\mathbf{y}}{dt} = -\alpha \nabla V(\mathbf{y})$ , а вихід мережі Хопфілда є лінією най-

швидшого спуску. Таким чином, якщо точка перебуває в області, де  $f^{-1}(\mathbf{y})$  є наближено лінійною, то мережа апроксимує розв'язок лінією найшвидшого спуску. Припустивши, що ФА та обернена до неї функція монотонно зростають, отримаємо  $\frac{d}{dy_i} [f^{-1}(y_i)] > 0$ . Із рівняння (14.13)

маємо, що точки, для яких  $\frac{d\mathbf{y}(t)}{dt} = 0$ , є також точками, в яких  $\nabla V(\mathbf{y}) = \mathbf{0}$ . Отже, точки-атрактори, які належать множині  $L$  і задовольняють рівняння (14.12), є також точками рівноваги ФЛ  $V(\mathbf{y})$ .

**Ефект посилення.** Функцію Ляпунова для мережі Хопфілда можна спростити, розглянувши випадок, коли підсилювач одержує досить велике значення  $\gamma$ . Нагадаємо, що у прикл. 14.1 було використано нелінійну ФА підсилювача (рис. 14.7) вигляду  $y = f(u) = \frac{2}{\pi} \tan^{-1} \left( \frac{\gamma \pi u}{2} \right)$ . Величина при-

росту  $\gamma$  визначає крутизну кривої за  $u = 0$ :

- 1) якщо  $\gamma$  зростає, то кут нахилу кривої до початку координат зростає;
- 2) якщо  $\gamma$  прямує до нескінченності, то  $f(u)$  наближається до функції з чітким порогом *hardlims*.

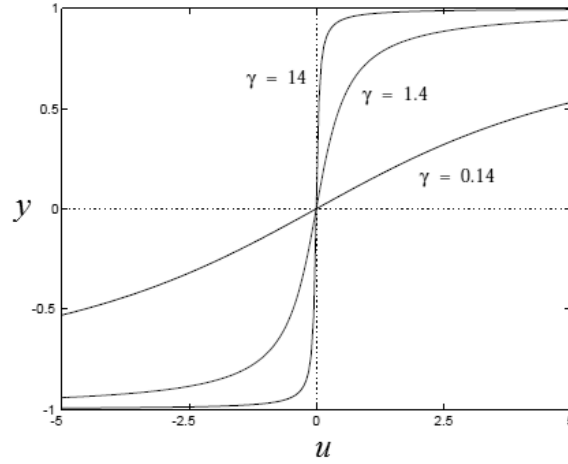


Рис. 14.7. Графік ФА підсилювача  $y = f(u) = \frac{2}{\pi} \tan^{-1} \left( \frac{\gamma \pi u}{2} \right)$

Відповідно до прикл. 14.1 ФЛ  $V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} + \sum_{i=1}^S \left( \int_0^{y_i} f^{-1}(x) dx \right) - \mathbf{b}^T \mathbf{y}$ .

Враховуючи, що  $f^{-1}(x) = \frac{2}{\gamma \pi} \tan \left( \frac{\pi x}{2} \right)$ , другий елемент ФЛ має вигляд

$$\int_0^{y_i} f^{-1}(x) dx = \frac{2}{\gamma \pi} \left[ \frac{2}{\pi} \log \left( \cos \left( \frac{\pi y_i}{2} \right) \right) \right] = -\frac{4}{\gamma \pi^2} \log \left[ \cos \left( \frac{\pi y_i}{2} \right) \right].$$

Графік другого елемента ФЛ для трьох різних значень посилення  $\gamma$  зображено на рис. 14.8. Якщо значення  $\gamma$  збільшується, то функція згладжується і досить довго є близькою до нуля. Таким чином, якщо приріст  $\gamma$  прямує до нескінченності, то інтеграл у другому елементі ФЛ наближається до нуля, якщо  $-1 < y_i < 1$ . Це дозволяє не розглядати другий

елемент ФЛ. Тому ФЛ з великим посиленням  $\gamma$  ( $\gamma \rightarrow \infty$ ) має вигляд

$$V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} - \mathbf{b}^T \mathbf{y}. \quad (14.14)$$

Порівнявши рівняння (14.14) із КФ  $F(\mathbf{x}) = -\frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + \mathbf{c}$ , можна побачити, що ФЛ з великим посиленням  $\gamma$  є квадратичною, оскільки

$$V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} - \mathbf{b}^T \mathbf{y} = \quad (14.15)$$

$$= \frac{1}{2} \mathbf{y}^T \mathbf{A} \mathbf{y} + \mathbf{d}^T \mathbf{y} + \mathbf{c},$$

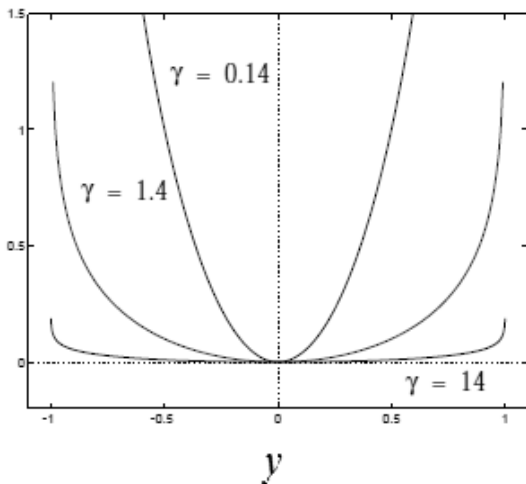


Рис. 14.8. Графік другого елемента функції Ляпунова

де  $\nabla^2 V(\mathbf{y}) = \mathbf{A} = -\mathbf{W}$ ;  $\mathbf{d} = -\mathbf{b}$ ;  $c = 0$ . Нагадаємо, що форма поверхні КФ визначається власними числами та власними векторами матриці Гессе. Матриця Гессе (прикл. 14.1) для ФЛ має вигляд  $\nabla^2 V(\mathbf{y}) = \mathbf{A} = -\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$ , тому власні числа визначаються як  $|\nabla^2 V(\mathbf{y}) - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & -1 \\ -1 & -\lambda \end{vmatrix} = \lambda^2 - 1 = (\lambda + 1)(\lambda - 1) = 0$  і мають вигляд  $\lambda_1 = -1$  та  $\lambda_2 = 1$ , а власні вектори  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$  і  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ .

Покажемо, як виглядає поверхня ФЛ з великим посиленням. Оскільки матриця Гессе має одне додатне та одне від'ємне власне число, то є сідлова точка, а поверхня має від'ємне викривлення вздовж першого власного вектора і додатне – уздовж другого вектора (рис. 14.9).

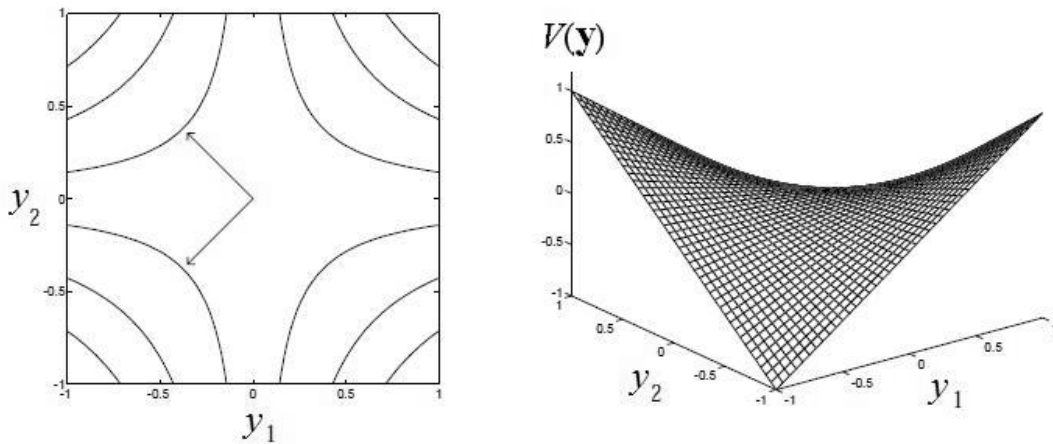


Рис. 14.9. Функція Ляпунова з великим посиленням

Хоча у такої ФЛ немає точки мінімуму, проте вона обмежена гіперкубом  $\{\mathbf{y}: -1 < y_i < 1, i = 1, 2\}$ . Отже, точки мінімуму є у двох кутах гіперкуба:

$$\mathbf{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}. \quad (14.16)$$

Якщо посилення  $\gamma$  має мале значення, то існує єдиний мінімум у початку координат. Зі збільшенням посилення мінімуми переміщуються з початку координат у напрямку точок (14.16). На рис. 14.9 зображено перехідний (проміжний) випадок для  $\gamma = 1,4$ , при цьому мінімуми розміщені в точках  $\mathbf{y} = \begin{bmatrix} 0,57 \\ 0,57 \end{bmatrix}$  і  $\mathbf{y} = \begin{bmatrix} -0,57 \\ -0,57 \end{bmatrix}$ .

## 14.5. Модель Хопфілда як асоціативна пам'ять

Для мережі Хопфілда не існує правила навчання. Замість цього для визначення вагової матриці використовують процедуру проектування, оснований на ФЛ. Метод проектування Хопфілда полягає у визначенні вагової матриці  $\mathbf{W}$  і вектора зсуву  $\mathbf{b}$ , які мінімізують ФЛ вигляду  $V(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T \mathbf{W} \mathbf{y} - \mathbf{b}^T \mathbf{y}$ . Таким чином, будь-яка задача зводиться до мінімізації квадратичної функції (коли мережа Хопфілда мінімізує  $V(\mathbf{y})$ , вона також розв'язує задачу).

Основна проблема полягає у зведенні  $V(\mathbf{y})$  до квадратичної форми, яка не завжди є очевидною.

**Пам'ять, адресована за змістом.** Опишемо, як мережу Хопфілда можна використати як асоціативну пам'ять. Змодельована пам'ять адресується за змістом, оскільки вона відновлює збережені записи на основі частини змісту. Відмінність звичайної комп'ютерної пам'яті полягає в тому, що в ній відновлення відбувається на основі адреси об'єкта. Наприклад, за наявності асоціативної бази даних з іменами, адресами і телефонами роботодавців можна отримати весь набір даних, ввівши лише ім'я роботодавця (або навіть частину імені). Пам'ять, адресована за змістом, схожа на автоасоціативну пам'ять, за винятком того, що тут використовується рекурентна мережа Хопфілда замість лінійного асоціатора.

Нехай у мережі Хопфілда збережено набір векторів-прототипів образів. Отримавши вектор на вході, мережа повинна визначити вектор-прототип, який найбільше схожий на нього. Для цього вектори-прототипи мають бути точками-мінімумами ФЛ. Нехай вектори-прототипи мають вигляд  $\{\mathbf{p}_1, \dots, \mathbf{p}_Q\}$  і кожний із них складається із  $S$  елементів (тобто  $\mathbf{p}_i \in \mathcal{R}^S$ ), значення яких дорівнює «1» або «-1». Припустимо, що  $Q \ll S$ , тому простір станів є досить великим і вектори-прототипи розміщені в цьому просторі не дуже близько один до одного. Оскільки ФЛ з великим значенням посилення є квадратичною, необхідно, щоб вектори-прототипи були точками мінімуму відповідної квадратичної функції. Нехай функція, яка визначає функціонування асоціативної пам'яті, має такий квадратичний показник якості:

$$J(\mathbf{y}) = -\frac{1}{2} \sum_{q=1}^Q \left( (\mathbf{p}_q)^T \mathbf{y} \right)^2. \quad (14.17)$$

Якщо елементи вектора  $\mathbf{y}$  належать проміжку  $[-1; 1]$ , то точками мінімуму функції  $J(\mathbf{y})$  є вектори-прототипи. Припустимо, що вектори-прототипи є ортогональними. Визначимо показник якості для одного



$$\text{з них, одержимо: } J(\mathbf{p}_j) = -\frac{1}{2} \sum_{q=1}^Q \left( (\mathbf{p}_q)^T \mathbf{p}_j \right)^2 = \frac{1}{2} \left( (\mathbf{p}_j)^T \mathbf{p}_j \right)^2 = \frac{S}{2}.$$

Визначимо показник якості для випадкового вхідного вектора  $\mathbf{p}$ , який не є близьким до жодного з векторів-прототипів. Кожний елемент суми в рівнянні (14.17) є скалярним добутком вектора-прототипа на вектор входу, який зростає з наближенням вхідного вектора до відповідного вектора-прототипа. Якщо вводиться вектор, який не є близьким до жодного вектора-прототипа, то всі елементи суми у рівнянні (14.17) будуть мати малі значення. Отже, функція  $J(\mathbf{y})$  буде мати найбільше значення (найменше від'ємне) у разі мінімального наближення до всіх векторів-прототипів, найменше значення (найбільше від'ємне) у разі максимального наближення до одного з векторів-прототипів. Таким чином, КФ безпомилково визначає функціонування асоціативної пам'яті.

Наступний крок полягає у виборі такої матриці ваги  $\mathbf{W}$  і вектора зсуву  $\mathbf{b}$ , щоб ФЛ  $V(\mathbf{y})$  була еквівалентна показнику якості  $J(\mathbf{y})$ . Використавши контрольоване правило навчання Хебба для визначення вагової матриці (якщо шукані значення збігаються із введеними), маємо  $\mathbf{W} = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q)^T$ .

Якщо зсув  $\mathbf{b} = \mathbf{0}$ , то ФЛ

$$V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} = -\frac{1}{2} \mathbf{y}^T \left( \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q)^T \right) \mathbf{y} = -\frac{1}{2} \left( \sum_{q=1}^Q \mathbf{y}^T \mathbf{p}_q (\mathbf{p}_q)^T \mathbf{y} \right).$$

Останню формулу можна переписати у вигляді

$$V(\mathbf{y}) = -\frac{1}{2} \sum_{q=1}^Q \left( (\mathbf{p}_q)^T \mathbf{y} \right)^2 = J(\mathbf{y}).$$

Отже, ФЛ дорівнює квадратичному показнику якості для асоціативної пам'яті, адресованої за змістом. Вихід мережі Хопфілда буде прямувати до збережених векторів-прототипів. Як було показано під час розгляду автоасоціативної пам'яті (розд. 4), контрольоване правило навчання Хебба не працює у випадку наявності значної кореляції між векторами-прототипами, тому було запропоновано псевдоінверсний метод. Якщо вектори-прототипи є ортогональними, то кожний із них є точкою рівноваги мережі. Однак мережа має й багато інших точок рівноваги і може прямувати до вектора-прототипа, який не є вектором-прототипом заданого образу. Зазначимо, що у разі використання правила Хебба кількість збережених векторів-прототипів не повинна перевищувати 15 % від кількості нейронів.

**Правило Хебба.** Розглянемо функціонування мережі Хопфілда у разі використання правила навчання Хебба для визначення вагової матриці

з ортогональністю векторів-прототипів. Контрольоване правило Хебба має вигляд  $\mathbf{W} = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q)^T$ . Подавши на вхід мережі вектор  $\mathbf{p}_j$ , одержимо:

$$\mathbf{W}\mathbf{p}_j = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q)^T \mathbf{p}_j = \mathbf{p}_j (\mathbf{p}_j)^T \mathbf{p}_j = S\mathbf{p}_j, \quad (14.18)$$

де друга рівність пояснюється ортогональністю векторів-прототипів, а третя тим, що кожний елемент вектора  $\mathbf{p}_j$  дорівнює «1» або «-1». Порівнявши формулу (14.18) з виразом  $\mathbf{W}\mathbf{p}_j = \lambda\mathbf{p}_j$ , одержимо, що кожний вектор-прототип є власним вектором вагової матриці із власним числом  $\lambda = S$ . Таким чином, власний простір  $X$  для власного числа  $\lambda = S$  має вигляд  $X = \text{span}\{\mathbf{p}_1, \dots, \mathbf{p}_Q\}$ . Цей простір містить усі вектори, які можна записати у вигляді лінійної комбінації векторів-прототипів. Будь-який вектор  $\mathbf{y}$ , який є лінійною комбінацією векторів-прототипів, є власним вектором:

$$\begin{aligned} \mathbf{W}\mathbf{y} &= \mathbf{W}(\alpha_1\mathbf{p}_1 + \dots + \alpha_Q\mathbf{p}_Q) = \alpha_1\mathbf{W}\mathbf{p}_1 + \dots + \alpha_Q\mathbf{W}\mathbf{p}_Q = \\ &= \alpha_1 S\mathbf{p}_1 + \dots + \alpha_Q S\mathbf{p}_Q = S(\alpha_1\mathbf{p}_1 + \dots + \alpha_Q\mathbf{p}_Q) = S\mathbf{y}. \end{aligned}$$

Власний простір  $X$  є  $Q$ -вимірним (якщо припустити, що вектори-прототипи є незалежними). Простір  $\mathcal{R}^S$  можна поділити на множини, що не перетинаються:  $\mathcal{R}^S = X \cup X^\perp$ , де  $X^\perp$  – ортогональне доповнення до  $X$ . Кожний вектор із  $X^\perp$  є ортогональним до будь-якого вектора з  $X$ , тобто для будь-якого вектора  $\mathbf{y} \in X^\perp$ :  $(\mathbf{p}_q)^T \mathbf{y} = 0$ ,  $q = 1, \dots, Q$ . Отже, якщо  $\mathbf{y} \in X^\perp$ , то маємо  $\mathbf{W}\mathbf{y} = \sum_{q=1}^Q \mathbf{p}_q (\mathbf{p}_q)^T \mathbf{y} = \sum_{q=1}^Q (\mathbf{p}_q \cdot 0) = \mathbf{0} = 0 \cdot \mathbf{y}$ . Таким чином,

$X^\perp$  визначає власний простір для власного числа  $\lambda = 0$ .

Підводячи підсумок, зазначимо, що вагова матриця  $\mathbf{W}$  має два власних числа:  $S$  та  $0$ . Власний простір для власного числа із значенням « $S$ » є лінійною оболонкою векторів-прототипів, а для власного числа із значенням « $0$ » є ортогональним доповненням простору лінійної оболонки векторів-прототипів.

Оскільки з рівняння (14.15) матриця Гессе для ФЛ  $V(\mathbf{y})$  із великим посиленням має вигляд  $\nabla^2 V = -\mathbf{W}$ , то власними числами для  $\nabla^2 V$  є « $-S$ » і « $0$ ». Власні числа матриці Гессе визначають форму квадратичної ФЛ з великим посиленням: оскільки перше власне число є від'ємним, то  $V$  має від'ємне викривлення на  $X$ , а оскільки друге власне число дорівнює нулю, то  $V$  має нульове викривлення на  $X^\perp$ . Оскільки функція  $V$  має від'ємне викривлення на  $X$ , то траєкторія функціонування мережі Хопфілда (вихід мережі) буде спадати в кутах гіперкуба  $\{\mathbf{y}: -1 < y_i < 1\}$ ,

який міститься у просторі  $X$ . Зазначимо, що у разі використання правила Хебба для визначення вагової матриці буде одержано принаймні два мінімуми ФЛ для кожного вектора-прототипа. Для кожного вектора-прототипа  $\mathbf{p}_q$ , вектор  $-\mathbf{p}_q$  теж буде міститися у просторі  $X$ , який є лінійною оболонкою векторів-прототипів. Вектор, протилежний до вектора-прототипа, буде розміщений в одному з кутів гіперкуба  $\{\mathbf{y}: -1 < y_i < 1\}$ , який міститься в  $X$ . Існують також інші мінімуми ФЛ, які не відповідають векторам-прототипам образів. Таким чином, мінімуми функції  $V$  розміщені в кутах гіперкуба  $\{\mathbf{y}: -1 < y_i < 1\}$  у просторі  $X$ . Ці кути, крім векторів-прототипів, містять їхні лінійні комбінації. Мінімуми, які не є векторами-прототипами, часто називають підробками. Метою побудови мережі Хопфілда є мінімізація кількості підробок та максимізація розміру області притягання (атрактора) для кожного вектора-прототипа.

Щоб проілюструвати ці принципи, знову розглянемо прикл. 14.1, в якому вагова матриця має вигляд  $\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ . Припустимо, що цю вагову матрицю отримано на основі використання правила Хебба з одним вектором-прототипом:  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . Тоді маємо  $\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ . Функція Ляпунова з великим посиленням (рис. 14.10) має вигляд  $V(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T \mathbf{W} \mathbf{y} = -\frac{1}{2}\mathbf{y}^T \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{y}$ . Матриця Гессе для функції  $V(\mathbf{y})$  має вигляд  $\nabla^2 V(\mathbf{y}) = -\mathbf{W} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$ , її власні числа  $\lambda_1 = -S = -2$ ;  $\lambda_2 = 0$ , а відповідні власні вектори  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$  і  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ . Перший власний вектор, який відповідає власному числу « $S$ », зображує простір  $X = \{\mathbf{y}: y_1 = y_2\}$ . Другий власний вектор, який відповідає власному числу « $0$ », є ортогональним доповненням до першого власного вектора і зображує простір  $X^\perp = \{\mathbf{y}: y_1 = -y_2\}$ .

Поверхня ФЛ має область нульового викривлення, що відповідає  $X^\perp$ . Початкові умови лівого або правого краю цієї області прямують відповідно до точки  $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$  або  $\mathbf{y} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ , змодельованих системою точок векторів-прототипів.

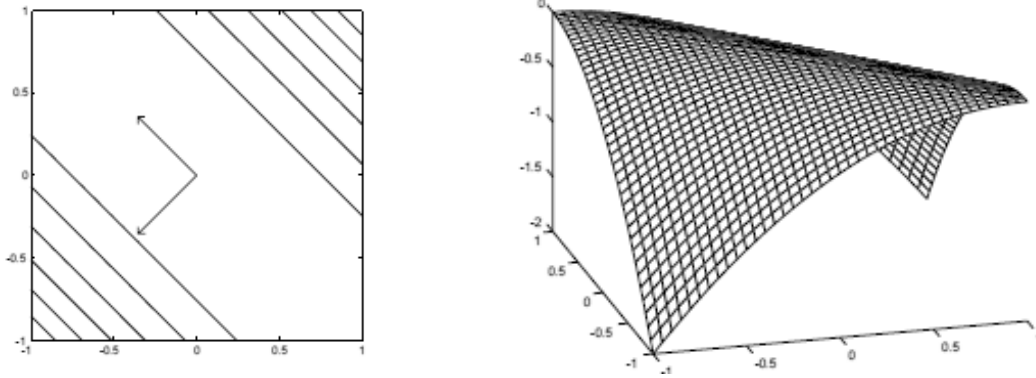


Рис. 14.10. Функція Ляпунова

**Поверхня Ляпунова.** У багатьох дослідженнях мережі Хопфілда діагональні елементи вагової матриці прирівнюють до нуля. Проаналізуємо ефективність такої операції для поверхні ФЛ.

Для мережі, що функціонує як асоціативна пам'ять, усі елементи, розміщені на діагоналі вагової матриці, дорівнюють значенню  $Q$  (кількість векторів-прототипів), оскільки елементи кожного вектора  $\mathbf{p}_q$  дорівнюють « $\pm 1$ ». Таким чином, можна обнулити діагональ, віднявши  $Q$  разів одиничну матрицю:  $\mathbf{W}' = \mathbf{W} - Q\mathbf{I}$ . Розглянемо, як ці зміни впливають на вигляд ФЛ. Помноживши нову вагову матрицю на один із векторів-прототипів, отримаємо  $\mathbf{W}'\mathbf{p}_q = [\mathbf{W} - Q\mathbf{I}]\mathbf{p}_q = S\mathbf{p}_q - Q\mathbf{p}_q = (S - Q)\mathbf{p}_q$ , тому  $(S - Q)$  є власним числом матриці  $\mathbf{W}'$  та відповідає власному простору  $X$  (лінійна оболонка векторів-прототипів). Помноживши нову вагову матрицю на вектор  $\mathbf{y} \in X^\perp$ , отримаємо:  $\mathbf{W}'\mathbf{y} = [\mathbf{W} - Q\mathbf{I}]\mathbf{y} = \mathbf{0} - Q\mathbf{y} = -Q\mathbf{y}$ , звідки  $-Q$  є власним числом матриці  $\mathbf{W}'$  та відповідає власному простору  $X^\perp$  (ортогональне доповнення простору  $X$ ). Таким чином, власні вектори матриці  $\mathbf{W}'$  збігаються із власними векторами матриці  $\mathbf{W}$ , але власні числа тепер дорівнюють  $(S - Q)$  і  $-Q$  замість « $S$ » і « $0$ ».

Підводячи підсумок, зазначимо, що власні числа матриці Гессе для модифікованої ФЛ (для якої справджується  $\nabla^2 V(\mathbf{y}) = -\mathbf{W}'$ ) дорівнюють числам  $-(S - Q)$  і  $Q$ . Це свідчить про те, що енергетична поверхня модифікованої ФЛ буде мати від'ємне викривлення у просторі  $X$  і додатне викривлення у просторі  $X^\perp$ , тоді як для звичайної ФЛ від'ємне викривлення спостерігається у просторі  $X$ , а нульове – у просторі  $X^\perp$ . З порівняння рис. 14.10 і 14.9 видно ефект впливу обнулення діагональних елементів вагової матриці на ФЛ. Отже, маємо:

1. Якщо початкова умова мережі Хопфілда перебуває за межами прямої  $y_1 = -y_2$ , то вихід мережі буде прямувати до одного з кутів гіперкуба  $\{\mathbf{y}: -1 < y_i < 1\}$ :  $\mathbf{y} = [1 \ 1]^T$  й  $\mathbf{y} = [-1 \ -1]^T$ .

2. Якщо початкова умова потрапляє безпосередньо на пряму  $y_1 = -y_2$  і використовується вагова матриця  $\mathbf{W}$ , то на виході мережі одержимо константу; якщо використовується вагова матриця  $\mathbf{W}'$ , то вихід мережі буде прагнути до сідлової точки (див. рис. 14.5, в).

Жодний із цих результатів не є бажаним, тому що вихід мережі не збігається в точці мінімуму ФЛ. Єдиним випадком, коли мережа прямує до сідлової точки, є той, коли початкова точка потрапляє безпосередньо на пряму  $y_1 = -y_2$ , що на практиці ніколи не трапляється.

### Приклади розв'язання задач\*

**Приклад 14.2.** Перевірити стійкість початку координат для системи, яка описується рівняннями  $\frac{dy_1}{dt} = -y_1 + (y_2)^2$ ;  $\frac{dy_2}{dt} = -y_2(y_1 + 1)$ .

*Розв'язання.* Основна задача полягає у визначенні ФЛ  $V(\mathbf{y})$ , яка є додатно визначеною та має від'ємну напіввизначену або від'ємно визначену похідну. Визначимо ФЛ як  $V(\mathbf{y}) = (y_1)^2 + (y_2)^2$ , тоді її похідна має вигляд

$$\begin{aligned} \frac{d}{dt}V(\mathbf{y}) &= (\nabla V)^T \left( \frac{d\mathbf{y}}{dt} \right) = \frac{\partial V}{\partial y_1} \left( \frac{dy_1}{dt} \right) + \frac{\partial V}{\partial y_2} \left( \frac{dy_2}{dt} \right) = 2y_1(-y_1 + (y_2)^2) + \\ &+ 2y_2(-y_2(y_1 + 1)) = -2(y_1)^2 - 2(y_2)^2. \end{aligned}$$

Отже, похідна  $\frac{dV(\mathbf{y})}{dt}$  є від'ємно визначеною, тому початок координат асимптотично стійкий.

**Приклад 14.3.** Перевірити стійкість початку координат для системи, яка описується рівняннями  $\frac{dy_1}{dt} = -(y_1)^5$ ;  $\frac{dy_2}{dt} = -5(y_2)^7$ .

*Розв'язання.* Визначимо ФЛ як  $V(\mathbf{y}) = (y_1)^2 + (y_2)^2$ , тоді маємо  $\frac{d}{dt}V(\mathbf{y}) = 2y_1(-(y_1)^5) + 2y_2(-5(y_2)^7) = -2(y_1)^6 - 10(y_2)^8$ . Таким чином, похідна  $\frac{dV(\mathbf{y})}{dt}$  є від'ємно визначеною, тому початок координат асимптотично стійкий.

**Приклад 14.4.** Розглянемо механічну систему, зображену на рис. 14.11, яка складається з пружини та поршня. Позначимо  $y_1 = x$ ;  $y_2 = \frac{dx}{dt}$ . Рівняння руху механічної системи мають вигляд  $\frac{dy_1}{dt} = y_2$ ;

---

\* Задачі взято з посібника [14].

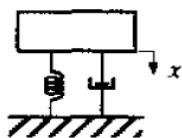


Рис. 14.11.  
Механічна система

$\frac{dy_2}{dt} = -(y_1)^3 - y_2$  – нелінійна пружність. Нехай ФЛ має вигляд  $V(\mathbf{y}) = 0,25(y_1)^4 + 0,5(y_2)^2$ . Використовуючи наслідок інваріантної теореми Ла–Салля, отримати інформацію про точки рівноваги та область притягання.

*Розв’язання.* Обчислимо похідну ФЛ:

$$\frac{d}{dt}V(\mathbf{y}) = \frac{\partial V}{\partial y_1} \left( \frac{\partial y_1}{\partial t} \right) + \frac{\partial V}{\partial y_2} \left( \frac{\partial y_2}{\partial t} \right) = (y_1)^3 y_2 + y_2 (-(y_1)^3 - y_2) = -(y_1)^2.$$

Таким чином, похідна  $\frac{dV(\mathbf{y})}{dt}$  є від’ємно визначеною і не змінює знака у просторі  $\mathbb{R}^2$ . Визначимо множину  $G = \Omega_\eta = \{\mathbf{y} : V(\mathbf{y}) \leq \eta\}$  за  $\eta = 1$ . Контурний графік функції  $V(\mathbf{y})$  та множину  $\Omega_1$  зображено на рис. 14.12.

Визначимо множину  $Z = \{\mathbf{y} : \frac{dV(\mathbf{y})}{dt} = 0, \mathbf{y} \in \bar{G}\} = \{\mathbf{y} : y_2 = 0, \mathbf{y} \in \bar{G}\} = \{\mathbf{y} : y_2 = 0, -\sqrt{2} \leq y_1 \leq \sqrt{2}\}$ . Інваріантна множина  $L = \{\mathbf{y} : y_1 = 0, y_2 = 0\}$ , тому початок координат  $\mathbf{y} = \mathbf{0}$  є аттрактором і  $\Omega_1$  – областю притягання. Збільшивши  $\eta$ , можна показати, що  $\mathbb{R}^2$  є областю притягання для початку координат. Рух механічної системи з початкової позиції  $\mathbf{2}$  та з початковою швидкістю  $\mathbf{2}$  ( $\mathbf{y}(0) = [2 \ 2]^T$ ) показано на рис. 14.13.

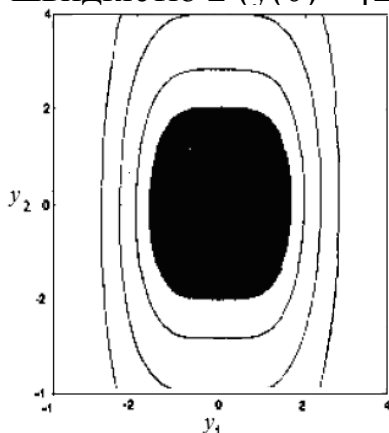


Рис. 14.12. Контурний графік функції  $V(\mathbf{y})$  та множина  $\Omega_1$

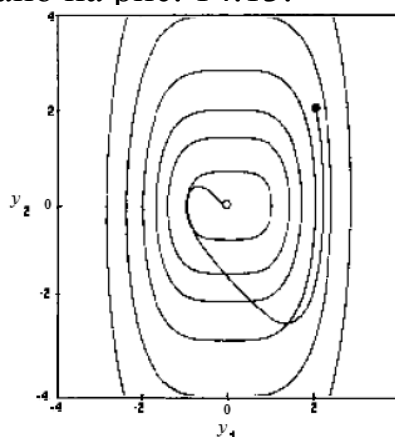


Рис. 14.13. Траєкторія механічної системи для початкової точки:  $\mathbf{y}(0) = [2 \ 2]^T$

**Приклад 14.5.** Розглянемо нелінійну систему вигляду

$$\frac{dy_1}{dt} = y_1((y_1)^2 + (y_2)^2 - 4) - y_2; \quad \frac{dy_2}{dt} = y_1 + y_2((y_1)^2 + (y_2)^2 - 4).$$

Ця система має дві інваріантні множини: початок координат  $\{\mathbf{y} : \mathbf{y} = \mathbf{0}\}$  і коло  $\{\mathbf{y} : (y_1)^2 + (y_2)^2 = 4\}$ . Використовуючи інваріантну теорему Ла–Салля для ФЛ  $V(\mathbf{y}) = (y_1)^2 + (y_2)^2$ , визначити область притягання початку координат.

*Розв'язання.* Обчислимо похідну ФЛ:

$$\begin{aligned} \frac{d}{dt}V(\mathbf{y}) &= \frac{\partial V}{\partial y_1} \left( \frac{\partial y_1}{\partial t} \right) + \frac{\partial V}{\partial y_2} \left( \frac{\partial y_2}{\partial t} \right) = 2y_1[y_1((y_1)^2 + (y_2)^2 - 4) - y_2] + \\ &+ 2y_2[y_1 + y_2((y_1)^2 + (y_2)^2 - 4)] = 2((y_1)^2 + (y_2)^2)((y_1)^2 + (y_2)^2 - 4). \end{aligned}$$

Отже,  $\frac{dV(\mathbf{y})}{dt} = 0$  у точці  $\mathbf{y} = \mathbf{0}$  і на колі  $(y_1)^2 + (y_2)^2 = 4$ .

Оскільки рух відбувається зовні кола радіуса 2 до його середини, то знак похідної  $\frac{dV(\mathbf{y})}{dt}$  на  $\mathbb{R}^2$  змінюється з «+» на «-», тому  $\frac{dV(\mathbf{y})}{dt}$  –

від'ємна напіввизначена в середині кола  $(y_1)^2 + (y_2)^2 = 4$ . Виберемо область  $G$  у середині кола так, щоб контур кола не входив до  $G$ . Наприклад, нехай  $G = \Omega_1 = \{\mathbf{y} : V(\mathbf{y}) \leq 1\}$ . Існують лише дві точки, в яких  $\frac{dV(\mathbf{y})}{dt} = 0$ , і тільки одна з них розміщена в середині  $\Omega_1$  – це точка

$\mathbf{y} = \mathbf{0}$ . Тому  $Z = \{\mathbf{y} : y_1 = 0, y_2 = 0\}$  і  $L^0 = L = Z$ . Початок координат є аттрактором, а  $\Omega_1$  – його область притягання. Ці ж аргументи можна використати, щоб показати, що область притягання для початку координат містить точки в середині кола  $(y_1)^2 + (y_2)^2 = 4$ .

На рис. 14.14 зображено дві траєкторії руху механічної системи: одна починається в середині кола  $(y_1)^2 + (y_2)^2 = 4$ , а інша – зовні нього. Коло – це інваріантна множина, яка не є аттрактором, отже, єдиний аттрактор цієї системи – початок координат.

**Приклад 14.6.** Нехай задано вектори-прототипи образів:  $\mathbf{p}_1 = [1 \ 1 \ -1 \ -1]^T$ ;  $\mathbf{p}_2 = [1 \ -1 \ 1 \ -1]^T$ .

**I.** Використовуючи правило Хебба, спроектувати мережу Хопфілда (визначити значення вагової матриці) розпізнавання образів.

**II.** Визначити матрицю Гессе для ФЛ з великим посиленням, її власні числа та власні вектори.

**III.** Визначити точки стійкого стану рівноваги для заданої мережі.

*Розв'язання. I.* Використовуючи контрольоване правило Хебба, визначимо вагову матрицю для заданих векторів входу. Одержимо:

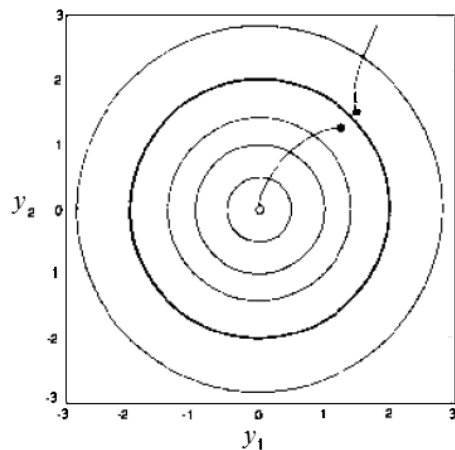


Рис. 14.14. Траєкторії до прикл. 14.5

$$\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T + \mathbf{p}_2(\mathbf{p}_2)^T = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 \\ -2 & 0 & 0 & 2 \end{bmatrix}.$$

**II.** Матриця Гессе для ФЛ з великим посиленням є протилежною до вагової матриці:  $\nabla^2 V(\mathbf{y}) = -\mathbf{W} = \begin{bmatrix} -2 & 0 & 0 & -2 \\ 0 & -2 & 2 & 0 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & -2 \end{bmatrix}$ .

Вектори-прототипи є ортогональними, оскільки  $(\mathbf{p}_1)^T \mathbf{p}_2 = 0$ . Тому власні числа мають вигляд  $\lambda_1 = -S = -4$  і  $\lambda_2 = 0$ . Власний простір для  $\lambda_1 = -4$  має вигляд  $X = \text{span}\{\mathbf{p}_1, \mathbf{p}_2\}$  і містить вектори, які можна записати у вигляді лінійної комбінації векторів-прототипів  $\mathbf{p}_1, \mathbf{p}_2$ . Власний простір для  $\lambda_2 = 0$  є ортогональним доповненням до  $X$ :  $X^\perp =$

$$= \text{span}\left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \right\}, \text{ і містить вектори, які є ортогональними до будь-якого}$$

вектора з  $X$ , тобто  $\forall \mathbf{p}_q \in X$  та  $\forall \mathbf{y} \in X^\perp$ :  $(\mathbf{p}_q)^T \mathbf{y} = 0, q = 1, 2, \dots$ .

**III.** Точки стійкого стану рівноваги мають вигляд  $\mathbf{p}_1, \mathbf{p}_2, -\mathbf{p}_1$  і  $-\mathbf{p}_2$ . Точки рівноваги можуть бути іншими, якщо кути гіперкуба розміщені у просторі  $X = \text{span}\{\mathbf{p}_1, \mathbf{p}_2\}$ . Всього маємо  $2^4 = 16$  кутів гіперкуба: чотири у просторі  $X$  і чотири у просторі  $X^\perp$ , решта кутів розподілені між просторами  $X$  і  $X^\perp$ .

**Приклад 14.7.** Розглянемо мережу Хопфілда з великим посиленням, яка має таку вагову матрицю та вектор зсуву:  $\mathbf{W} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$ ;

$\mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ . Побудувати графік контурних ліній ФЛ для заданої мережі.

Для початкової умови вигляду  $\mathbf{y}(0) = [1 \ 1]^T$  визначити точки, в яких мережа збігається.

*Розв'язання.* Функція Ляпунова з великим посилення має вигляд  $V(\mathbf{y}) = -\frac{1}{2} \mathbf{y}^T \mathbf{W} \mathbf{y} - \mathbf{b}^T \mathbf{y}$ , її матриця Гессе  $\nabla^2 V(\mathbf{y}) = -\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ . Визначи-

мо власні числа  $|\nabla^2 V(\mathbf{y}) - \lambda \mathbf{I}| = \begin{vmatrix} 1-\lambda & 1 \\ 1 & 1-\lambda \end{vmatrix} = -\lambda^2 - 2\lambda + 1 - 1 = \lambda(\lambda - 2)$ ,

звідки  $\lambda_1 = 0, \lambda_2 = 2$ . Тепер визначимо власні вектори:



– для  $\lambda_1 = 0$  маємо  $[\nabla^2 V(\mathbf{y}) - \lambda_1 \mathbf{I}]\mathbf{z}_1 = 0$ , звідки  $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{z}_1 = 0$ ,  
й  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ ;

– для  $\lambda_2 = 2$  маємо  $[\nabla^2 V(\mathbf{y}) - \lambda_2 \mathbf{I}]\mathbf{z}_2 = 0$ , звідки  $\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{z}_2 = 0$ ,  
й  $\mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ .

Отже, елемент  $-\frac{1}{2}\mathbf{y}^T \mathbf{W} \mathbf{y}$  має нульове викривлення у напрямку вектора  $\mathbf{z}_1$  і додатне – у напрямку  $\mathbf{z}_2$ . Спочатку побудуємо графік контурних ліній ФЛ без врахування лінійного елемента (рис. 14.15, а). Оскільки лінійний елемент формує від’ємне викривлення в напрямку  $\mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ , то ФЛ теж буде мати від’ємне викривлення в цьому ж напрямку (рис. 14.15, б). Незалежно від початкових умов усі траєкторії будуть збігатися до точки  $[1 \ -1]^T$ . Функція Ляпунова має лише один мінімум у точці  $[1 \ -1]^T$  (рис. 14.15, б). Зазначимо, що вихід мережі обмежений гіперкубом  $\{\mathbf{y}: -1 < y_1 < 1\}$ .

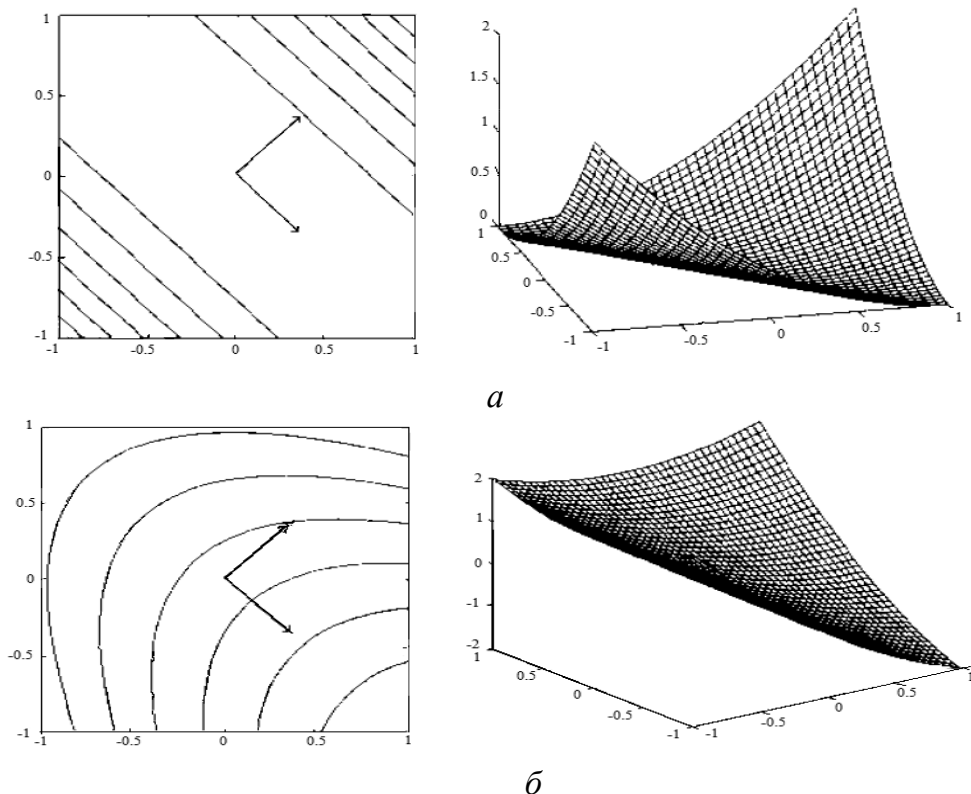


Рис. 14.15. Контурні лінії ФЛ без лінійного елемента (а) та з лінійним елементом (б)

**Приклад 14.8.** Нехай задано вектори-прототипи образів:  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;

$$\mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

**I.** Спроектувати мережу Хопфілда для розпізнавання заданих образів.

**II.** Визначити матрицю Гессе для ФЛ з великим посиленням, її власні числа та власні вектори.

**III.** Визначити точки стійкої рівноваги мережі Хопфілда з великим посиленням.

**IV.** Як добре задана мережа виконує розпізнавання образів?

*Розв'язання.* **I.** Для визначення вагової матриці використаємо правило Хебба вигляду  $\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T + \mathbf{p}_2(\mathbf{p}_2)^T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ .  
Виберемо значення зсуву  $\mathbf{b} = [0 \ 0]^T$ .

**II.** Матриця Гессе для ФЛ з великим посиленням  $\nabla^2 V(\mathbf{y}) = -\mathbf{W} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$ . Маємо власні числа  $\lambda_1 = \lambda_2 = -S = -2$  та власні вектори  $\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ;  $\mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ , або будь-які їхні лінійні комбінації. Вхідний простір  $\mathcal{R}^2$  є власним простором для власного числа  $\lambda = -2$ .

**III.** Відомо, що якщо власні числа матриці Гессе збігаються, то контурні лінії відповідної функції є колоподібними. Оскільки власні числа від'ємні, то функція має один максимум. Існують чотири точки мінімуму в чотирьох кутах гіперкуба  $\{\mathbf{y}: -1 < y_i < 1\}$  та одна сідлова точка (всього маємо дев'ять стійких точок рівноваги). Функцію Ляпунова з великим посиленням зображено на рис. 14.16.

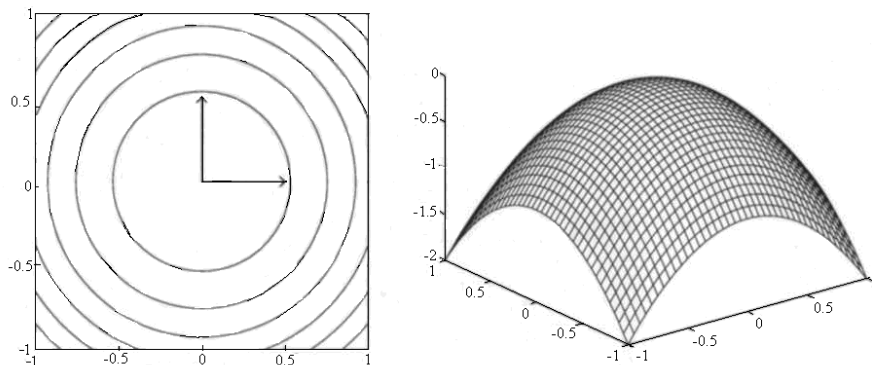


Рис. 14.16. Функція Ляпунова з великим посиленням

Використавши теорему інваріантності Ла–Салля, можна показати, що точка максимуму в початку координат має область притягання, що склада-

ється лише з початку координат. Отже, вона не є точкою стійкої рівноваги. Областями притягання сідлових точок є лінії. Наприклад, сідлова точка  $[-1 \ 0]^T$  має область притягання вздовж від'ємної осі  $y_1$ . Чотири кути гіперкуба є єдиними атракторами, які мають двовимірні області притягання: областю притягання кожного кута є відповідний квадрат гіперкуба. Функцію Ляпунова з невеликим посиленням з коефіцієнтом посилення  $\gamma = 1,4$  і збіжністю до сідлової точки та до точки мінімуму зображено на рис. 14.17.

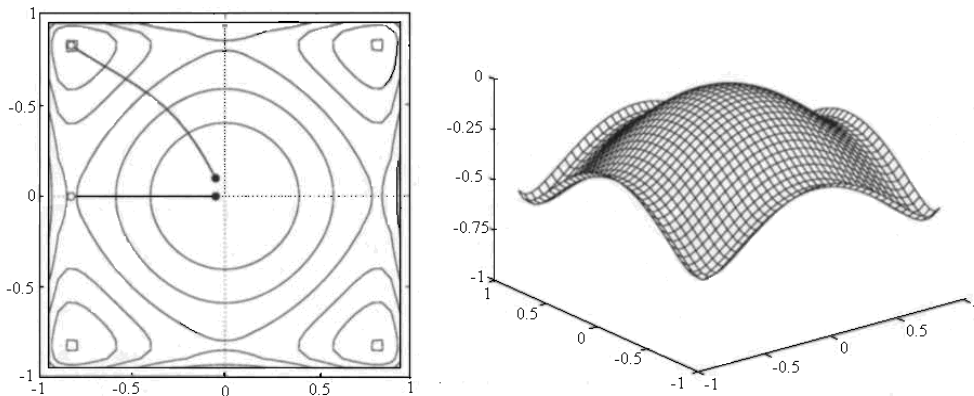


Рис. 14.17. Функція Ляпунова з невеликим посиленням

**IV.** Мережа виконує не дуже якісну роботу з розпізнавання образів. Вона збігається до кута, найближчого до вектора входу, навіть якщо необхідно зберегти два вектори-прототипи образів. Це відбувається, оскільки кількість векторів-прототипів, які мережа може зберегти за правилом Хебба, становить лише 15 % від кількості нейронів (маємо лише два нейрони).

**Приклад 14.9.** Нехай мережа Хопфілда має ФЛ з великим посиленням вигляду  $V(\mathbf{y}) = -\frac{1}{2}(7(y_1)^2 + 12y_1y_2 - 2(y_2)^2)$ .

**I.** Визначити вагову матрицю мережі.

**II.** Визначити градієнт та матрицю Гессе для ФЛ.

**III.** Побудувати контурний графік ФЛ. Побудувати траєкторію, яку проходить алгоритм найшвидшого спуску, що відповідає функції  $V(\mathbf{y})$  з початковими умовами  $[0,25 \ 0,25]^T$ .

**Розв'язання. I.** Функція  $V(\mathbf{y})$  є квадратичною і має вигляд  $V(\mathbf{y}) = -\frac{1}{2}(7(y_1)^2 + 12y_1y_2 - 2(y_2)^2) = -\frac{1}{2}\mathbf{y}^T \begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix} \mathbf{y}$ , тому вагова матриця  $\mathbf{W} = \begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix}$ .

**II.** Оскільки  $V(\mathbf{y})$  – квадратична функція, то її градієнт

$$\nabla V(\mathbf{y}) = -\begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix} \mathbf{y}, \text{ а матриця Гессе } \nabla^2 V(\mathbf{y}) = -\begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix}.$$

$$\text{III. Визначимо власні числа: } |\nabla^2 V(\mathbf{y}) - \lambda \mathbf{I}| = \begin{vmatrix} -7-\lambda & -6 \\ -6 & 2-\lambda \end{vmatrix} = (\lambda+10)(\lambda-5),$$

звідки  $\lambda_1 = -10$ ;  $\lambda_2 = 5$ . Тепер визначимо власні вектори:

$$\text{— для } \lambda_1 = -10 \text{ маємо } (\nabla^2 V(\mathbf{y}) - \lambda_1 \mathbf{I}) \mathbf{z}_1 = 0, \text{ звідки } \begin{pmatrix} 3 & -6 \\ -6 & 12 \end{pmatrix} \mathbf{z}_1 = 0;$$

$$\mathbf{z}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix};$$

$$\text{— для } \lambda_2 = 5 \text{ маємо } (\nabla^2 V(\mathbf{y}) - \lambda_2 \mathbf{I}) \mathbf{z}_2 = 0, \text{ звідки } \begin{bmatrix} -12 & -6 \\ -6 & -3 \end{bmatrix} \mathbf{z}_2 = 0;$$

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

Випадок  $\lambda_1 < 0 < \lambda_2$  відповідає сідловій точці. Маємо від'ємне викривлення уздовж вектора  $\mathbf{z}_1$  і додатне — уздовж  $\mathbf{z}_2$ . Графік контурних ліній ФЛ з великим посиленням і траєкторію найшвидшого спуску зображено на рис. 14.18. Траєкторія відповідає від'ємному градієнту та перпендикулярна до контурних ліній. Коли траєкторія досягає вершини гіперкуба, вона спрямована вниз до точки мінімуму.

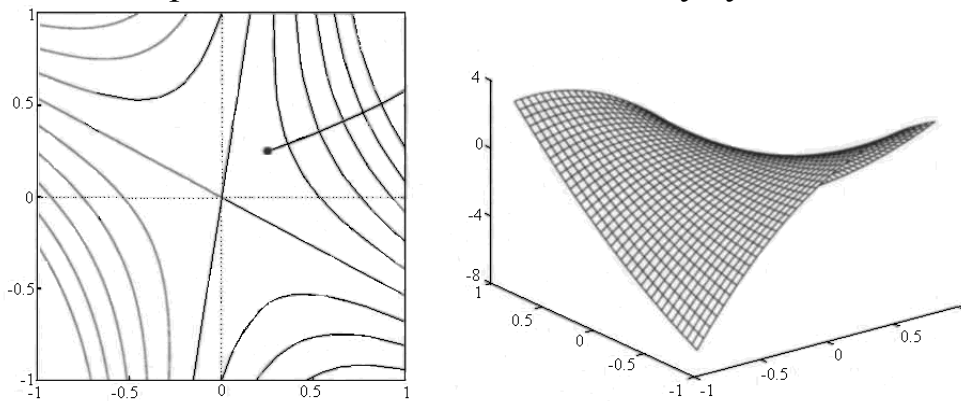


Рис. 14.18. Функція Ляпунова з великим посиленням і траєкторія найшвидшого спуску

Функція Ляпунова з великим посиленням є лише апроксимацією, оскільки вона має невизначене посилення. Для порівняння на рис. 14.19 зображено ФЛ та траєкторію функціонування мережі Хопфілда для посилення  $\gamma = 0,5$ .

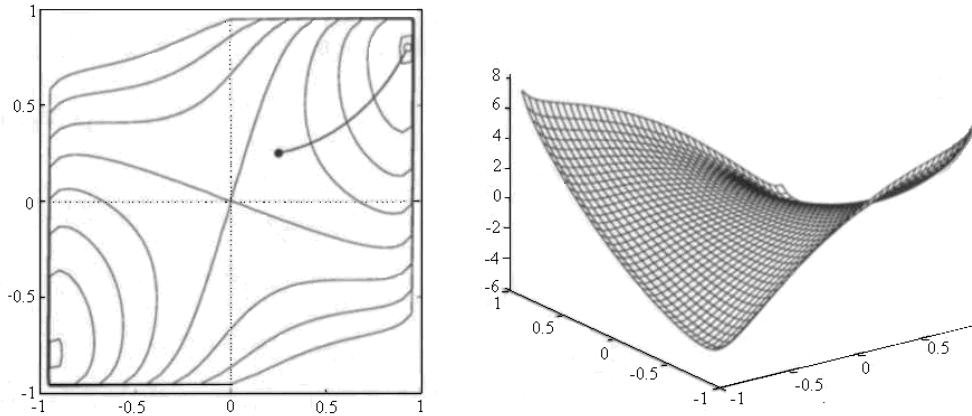


Рис. 14.19. Функція Ляпунова та траєкторія мережі Хопфілда для  $\gamma = 0,5$

**Приклад 14.10.** Мережу Хопфілда використовують для додатків з адресною пам'яттю. Прикладом такого використання є аналого-цифрове перетворення, яке полягає в тому, що спочатку береться аналоговий сигнал  $y$ , який конвертується в серію бітів (нулів та одиниць). Наприклад, 2-бітний аналого-цифровий перетворювач намагається апроксимувати цей сигнал таким чином:  $y \cong \sum_{i=1}^2 a_i 2^{(i-1)} = a_1 + a_2 2$ , де  $a_1$  та  $a_2$  набувають значення «0» і «1». На основі мережі Хопфілда виконується перетворення такого вигляду:

$$J(\mathbf{a}) = \frac{1}{2} \left( y - \sum_{i=1}^2 a_i 2^{(i-1)} \right)^2 - \frac{1}{2} \left( \sum_{i=1}^2 2^{2(i-1)} a_i (a_i - 1) \right),$$

де перший елемент зображує похибку аналого-цифрового перетворення, а другий змушує  $a_1$  та  $a_2$  набувати значення «0» і «1».

Показати, що це перетворення:

**I.** Можна записати у вигляді ФЛ.

**II.** Встановити, чи воно визначає вагову матрицю та вектор зсуву мережі Хопфілда.

*Розв'язання. I.* Розпишемо елементи функції перетворення:

$$\begin{aligned} \left( y - \sum_{i=1}^2 a_i 2^{(i-1)} \right)^2 &= y^2 - 2y \sum_{i=1}^2 a_i 2^{(i-1)} + \sum_{j=1}^2 \sum_{i=1}^2 a_i a_j 2^{(j-1)+(i-1)}; \\ \left( \sum_{i=1}^2 2^{2(i-1)} a_i (a_i - 1) \right) &= \sum_{i=1}^2 (a_i)^2 2^{2(i-1)} - \sum_{i=1}^2 (a_i) 2^{2(i-1)}. \end{aligned}$$

Підставивши ці елементи у функцію перетворення, одержимо:

$$J(\mathbf{a}) = \frac{1}{2} \left( y^2 + \sum_{\substack{j=1 \\ i \neq j}}^2 \sum_{i=1}^2 a_i a_j 2^{(j-1)+(i-1)} + \sum_{i=1}^2 a_i (2^{2(i-1)} - 2^i y) \right).$$

Оскільки перший елемент не є функцією від  $a$ , то він не впливає на поведінку функції і його можна не враховувати.

**II.** Покажемо, що задане перетворення має вигляд ФЛ з великим посиленням:  $V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T \mathbf{W} \mathbf{a} - \mathbf{b}^T \mathbf{a}$ . Це буває у випадку, якщо

$$\mathbf{W} = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \text{ і } \mathbf{b} = \begin{bmatrix} y - \frac{1}{2} \\ 2y - 2 \end{bmatrix}. \text{ Така мережа Хопфілда відрізняється від}$$

адресної пам'яті: входом у мережу Хопфілда є скалярна величина  $y$ , яку використовують для обчислення вектора зсуву, а входом до адресної пам'яті  $y$  у вигляді мережі є вектори-прототипи образів. Зазначимо, що функція активації цієї мережі має обмежувати виходи в межах  $0 < a < 1$ .

Прикладом такої функції може бути  $f(u) = \frac{1}{(1 - e^{-\gamma u})}$ .

### **Контрольні запитання**

1. Опишіть концепцію теорії стійкості (точка рівноваги, стійкість за Ляпуновим, асимптотична стійкість, теорема Ляпунова про стійкість; інваріантна теорема Ла–Салля та її наслідок) – див. дод. Д.

2. Опишіть схему моделі мережі Хопфілда дискретного часу. Які її особливості?

3. Опишіть схему моделі мережі Хопфілда неперервного часу. Які її особливості? Опишіть ФЛ для цієї мережі.

4. Визначіть поняття «атрактор».

5. Опишіть модель мережі Хопфілда, яку використовують для реалізації асоціативної пам'яті.

### **Задачі для самостійного розв'язання**

**Задача 14.1.** Використовуючи теорему стійкості Ляпунова, перевірити стійкість початку координат для систем вигляду:

**I.**  $\frac{dy_1}{dt} = -(y_1)^2 + (y_2)$ ;  $\frac{dy_2}{dt} = -y_1 - y_2$ .

**II.**  $\frac{dy_1}{dt} = -y_1 + (y_2)^2$ ;  $\frac{dy_2}{dt} = -y_2(y_1 + 1)$ .

**Задача 14.2.** Розглянути нелінійну систему:

$$\frac{dy_1}{dt} = y_2 - 2y_1((y_1)^2 + (y_2)^2); \quad \frac{dy_2}{dt} = -y_1 - 2y_2((y_1)^2 + (y_2)^2).$$

**I.** Дослідити на стійкість початок координат, використовуючи інваріантну теорему Ла–Салля та ФЛ вигляду  $V(y) = \alpha(y_1)^2 + \beta(y_2)^2$ .

**II.** Перевірити отриманий результат у системі MatLab, написавши *m*-файл для моделювання траєкторій цієї системи за декількох початкових умов. Побудувати графіки траєкторій.

**Задача 14.3.** Розглянути нелінійну систему вигляду  $\frac{dy}{dt} = \sin y$ . Визначити інваріантні множини, ФЛ, аттрактори та області притягання.

**Задача 14.4.** Розглянути нелінійну систему вигляду  $\frac{dy_1}{dt} = (y_2)$ ;  
 $\frac{dy_2}{dt} = -y_1 - (y_2)^3$ .

**I.** Визначити точки рівноваги. Дослідити на стійкість точки рівноваги, використавши наслідок інваріантної теореми Ла–Салля та ФЛ вигляду  $V(y) = (y_1)^2 + (y_2)^2$ .

**II.** Перевірити результати, отримані в п. I, у системі MatLab, написавши *m*-файл для моделювання траєкторій цієї системи за декількох початкових умов.

**Задача 14.5.** Розглянути нелінійну систему вигляду  $\frac{dy}{dt} = (1-y)(1+y) = 1 - y^2$ . Визначити точки рівноваги та ФЛ (на основі похідної).

**Задача 14.6.** Розглянути нелінійну систему вигляду  $\frac{dy_1}{dt} = y_2 - y_1((y_1)^4 + 2(y_2)^2 - 10)$ ;  $\frac{dy_2}{dt} = -(y_1)^3 - 3(y_2)^5((y_1)^4 + 2(y_1)^2 - 10)$ .

**I.** За допомогою системи MatLab визначити інваріантні множини.

**II.** Дослідити на стійкість інваріантні множини, отримані в п. I, використавши наслідок інваріантної теореми Ла–Салля та ФЛ  $V(y) = ((y_1)^4 + 2(y_2)^2 - 10)^4$ .

**Задача 14.7.** Нехай задано вектори-прототипи образів:  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ . Мережа Хопфілда для розпізнавання заданих образів має вагову матрицю та зсув:  $\mathbf{W} = \begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix}$ ;  $\mathbf{b} = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$ .

**I.** Побудувати графік контурних ліній ФЛ з великим посиленням.

**II.** Написати *m*-файл системи MatLab, який моделює мережу Хопфілда, та отримати виходи цієї мережі для декількох початкових умов.

**Задача 14.8.** Нехай задано мережу Хопфілда з ФЛ з великим посиленням  $V(y) = -\frac{1}{2}((y_1)^2 + 2y_1y_2 + 4(y_2)^2 + 6y_1 + 10y_2)$ .

- I.** Визначити вагову матрицю та вектор зсуву для цієї мережі.
- II.** Визначити градієнт і матрицю Гессе для функції  $V(y)$ .
- III.** Побудувати графік контурних ліній для функції  $V(y)$ .
- IV.** Визначити стаціонарні точки для  $V(y)$ . Використавши теорему інваріантності Ля–Салля, надати якомога більше інформації про зсув та області притягання будь-яких точок стійкої рівноваги.

**Задача 14.9.** У прикл. 14.10 демонструвалося, як мережа Хопфілда може бути використана в аналого-цифровому перетворенні.

**I.** Побудувати графік контурних ліній ФЛ з великим посиленням для 2-бітного аналого-цифрового перетворювача у вигляді мережі, використавши значення входу  $y = 0,5$ . Визначити точки мінімуму.

**II.** Виконати п. I для значення входу  $y = 2,5$ .

**III.** Використавши результати, отримані в п. I і II, пояснити принцип роботи мережі. Чи буде мережа адекватно виконувати аналого-цифрове перетворення?

**Задача 14.10.** Нехай задано вектори-прототипи образів:  $\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$ ;

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}.$$

**I.** Спроекувати мережу Хопфілда для розпізнавання заданих образів (визначити вагову матрицю та вектор зсуву, використавши правило Хебба).

**II.** Визначити матрицю Гессе для ФЛ з великим підсиленням, її власні вектори та власні числа.

**III.** Визначити точки стійкої рівноваги для цієї мережі Хопфілда з великим посиленням.



## Додатки

### Додаток А

#### Приклади функцій активації пакету прикладних програм Neural Network Toolbox системи MatLab

Розглянемо приклади функцій активації.

1. HARDLIM – ФА з різким порогом. Синтаксис:  $y = \text{hardlim}(u)$ . Функцію активації *hardlim* та її похідну *dhardlim* визначають таким чином:

$$\text{hardlim}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 1, & \text{якщо } u \geq 0; \end{cases} \quad d\text{hardlim}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 0, & \text{якщо } u \geq 0; \end{cases}$$

Така послідовність команд будує графік ФА *hardlim* (рис. А.1):

$$u = -3:0.1:3; y = \text{hardlim}(u); \text{plot}(u, y).$$

Приклад. Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  ФА з різким порогом для шару з трьох нейронів й обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$y = \text{hardlim}(\mathbf{u})$	$\mathbf{d} = d\text{hardlim}(\mathbf{u}, \mathbf{y})$
1	0
1	0
0	0

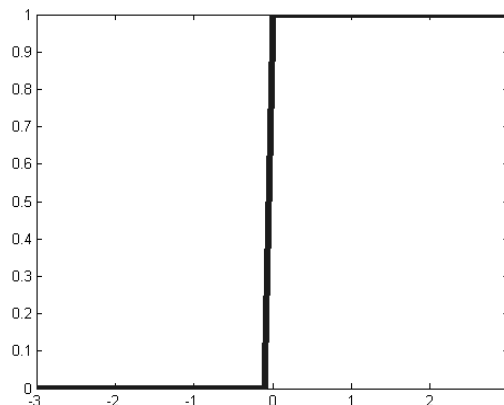


Рис. А.1. Графік ФА *hardlim*

2. HARDLIMS – симетрична ФА з різким порогом. Синтаксис:  $y = \text{hardlims}(u)$ . Функцію активації *hardlims* та її похідну *dhardlims* визначають таким чином:

$$\text{hardlims}(u) = \begin{cases} -1, & \text{якщо } u < 0; \\ 1, & \text{якщо } u \geq 0; \end{cases} \quad d\text{hardlims}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 0, & \text{якщо } u \geq 0. \end{cases}$$

Така послідовність команд буде графік ФА *hardlims* (рис. А.2):

$$u = -3:0.1:3; y = \text{hardlims}(u); \text{plot}(u,y).$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  симетричної ФА з різким порогом для шару із трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ .

Одержимо:

$\mathbf{y} = \text{hardlims}(\mathbf{u})$	$\mathbf{d} = d\text{hardlims}(\mathbf{u}, \mathbf{y})$
1	0
1	0
~1	0

3. PURELIN – лінійна ФА. Синтаксис:  $y = \text{purelin}(u)$ . Функцію активації *purelin* та її похідну *dpurelin* визначають таким чином:  $\text{purelin}(u) = u$ ;  $dpurelin(u) = 1$ . Така послідовність команд буде графік ФА *purelin* (рис. А.3):

$$u = -3:0.1:3; y = \text{purelin}(u); \text{plot}(u,y).$$

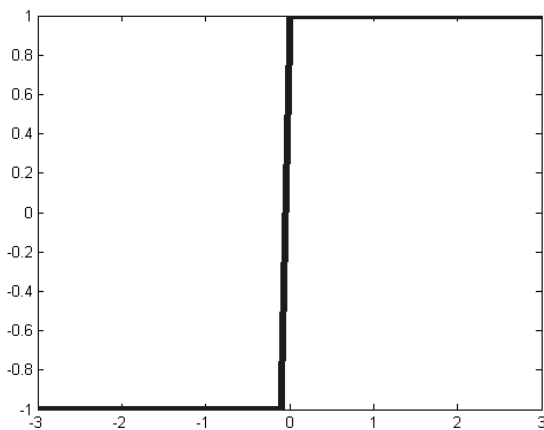


Рис. А.2. Графік ФА *hardlims*

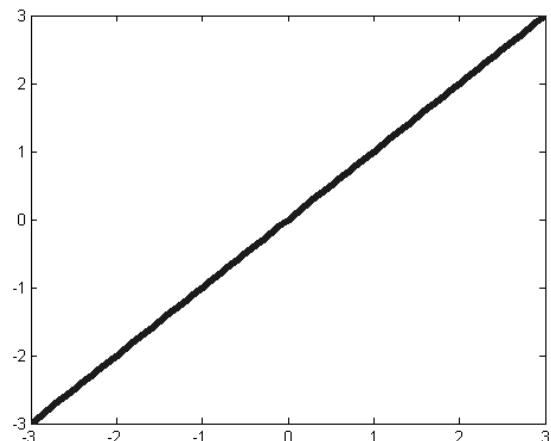


Рис. А.3. Графік ФА *purelin*

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  лінійної ФА для шару з трьох нейронів, обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$\mathbf{y} = \text{purelin}(\mathbf{u})$	$\mathbf{d} = dpurelin(\mathbf{u}, \mathbf{y})$
0,10	1
0,80	1
-0,70	1

4. POSLIN – додатна лінійна ФА Синтаксис:  $y = \text{poslin}(u)$ . Функцію активації *poslin* та її похідну *dposlin* визначають таким чином:

$$\text{poslin}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ u, & \text{якщо } 0 \leq u; \end{cases} \quad d\text{poslin}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 1, & \text{якщо } 0 \leq u. \end{cases}$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  додатної лінійної ФА для шару з трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$y = \text{poslin}(\mathbf{u})$	$\mathbf{d} = \text{dposlin}(\mathbf{u}, \mathbf{y})$
0,1000	1
0,8000	1
0	0

5. SATLIN – лінійна ФА з порогом. Синтакс:  $y = \text{satlin}(u)$ . Функцію активації *satlin* та її похідну *dsatlin* визначають таким чином:

$$\text{satlin}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ u, & \text{якщо } 0 \leq u \leq 1; \\ 1, & \text{якщо } u > 1; \end{cases} \quad \text{dsatlin}(u) = \begin{cases} 0, & \text{якщо } u < 0; \\ 1, & \text{якщо } 0 \leq u \leq 1; \\ 0, & \text{якщо } u > 1. \end{cases}$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  лінійної ФА з порогом для шару з трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$y = \text{satlin}(\mathbf{u})$	$\mathbf{d} = \text{dsatlin}(\mathbf{y}, \mathbf{u})$
0,10	1
0,80	1
0	0

6. SATLINS – симетрична лінійна ФА з порогом. Синтаксис:  $y = \text{satlins}(u)$ . Функцію активації *satlins* та її похідну *dsatlins* визначають таким чином:

$$\text{satlins}(u) = \begin{cases} -1, & \text{якщо } u < -1; \\ u, & \text{якщо } -1 \leq u \leq 1; \\ 1, & \text{якщо } u > 1; \end{cases} \quad \text{dsatlins}(u) = \begin{cases} 0, & \text{якщо } u < -1; \\ 1, & \text{якщо } -1 \leq u \leq 1; \\ 0, & \text{якщо } u > 1. \end{cases}$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  симетричної лінійної ФА з порогом для шару із трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$y = \text{satlins}(\mathbf{u})$	$\mathbf{d} = \text{dsatlins}(\mathbf{u}, \mathbf{y})$
0,1000	1
0,8000	1
0,7000	1

7. RADBAS – радіальна базисна ФА. Синтаксис:  $y = \text{radbas}(u)$ . Така послідовність команд будує графік ФА *radbas* (рис. А.4):

$$u = -3:0.1:3; y = \text{radbas}(u); \text{plot}(u, y).$$

Функцію активації *radbas* та її похідну *dradbas* визначають таким чином:  $radbas(u) = e^{-2u}$ ;  $dradbas(u) = -2ue^{-2u}$ .

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ 0,7]^T$  радіальної базисної ФА для шару із трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$\mathbf{y} = radbas(\mathbf{u})$	$\mathbf{d} = dradbas(\mathbf{u}, \mathbf{y})$
0,9900	-0,1980
0,5273	-0,8437
0,6126	0,8577

8. TRIBAS – трикутна ФА. Синтаксис:  $y = tribas(u)$ . Така послідовність команд будує графік ФА *tribas* (рис. А.5):

$u = -2:0.1:2; y = tribas(u); plot(u,y).$

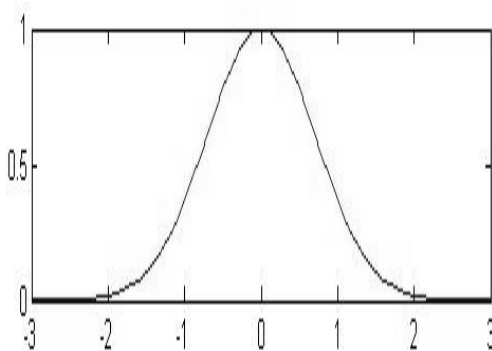


Рис. А.4. Графік ФА *radbas*

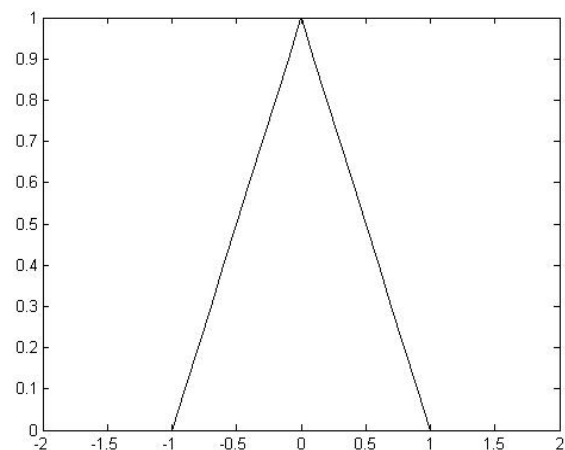


Рис. А.5. Графік ФА *tribas*

Функцію активації *tribas* та її похідну *dtribas* визначають таким чином:

$$tribas(u) = \begin{cases} 0, & \text{якщо } u < -1; \\ 1 - \text{abs}(u), & \text{якщо } -1 \leq u \leq 1; \\ 0, & \text{якщо } u > 1; \end{cases} \quad dtribas(u) = \begin{cases} 0, & \text{якщо } u < -1; \\ 1, & \text{якщо } -1 \leq u < 0; \\ -1, & \text{якщо } 0 < u < 1; \\ 0, & \text{якщо } u > 1. \end{cases}$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  трикутної ФА для шару із трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$\mathbf{y} = tribas(\mathbf{u})$	$\mathbf{d} = dtribas(\mathbf{u}, \mathbf{y})$
0,90	-1
0,20	-1
0,30	-1

9. COMPET – конкуруюча ФА. Синтаксис:  $y = \text{compet}(u)$ . Функція *compet* не має похідної. Вона перетворює вектор входу шару нейронів таким чином, щоб нейрон із найбільшим входом мав вихід зі значенням «1», а всі інші нейрони – зі значенням «0».

Приклад. Задамо вектор входу  $\mathbf{u} = [0,5 \ 1 \ -0,5 \ 0,5]^T$  конкуруючої ФА. Розрахуємо вектор виходу  $\mathbf{y}$ , зобразимо входи та виходи у вигляді стовпців діаграм (рис. А.6):

```
y = compet(u); subplot(2,1,1), bar(u), ylabel('u')
subplot(2,1,2), bar(y), ylabel('y').
```

10. LOGSIG – логістична сигмоїдна ФА. Синтаксис:  $y = \text{logsig}(u)$ . Побудуємо графік ФА *logsig* (рис. А.7):

```
u = -5:0.1:5; y = logsig(u); plot(u,y).
```

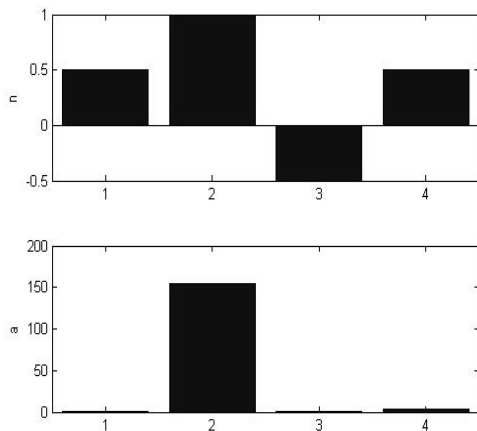


Рис. А.6. Графік ФА *compet*

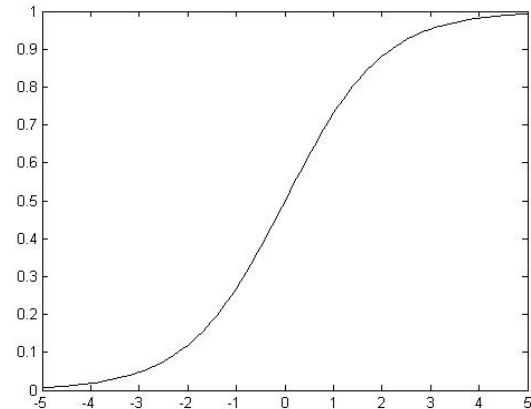


Рис. А.7. Графік ФА *logsig*

Функцію активації *logsig* та її похідну *dlogsig* визначають таким чином:

$$\text{logsig}(u) = \frac{1}{1 + e^{-u}}; \quad d\text{logsig}(u) = \frac{e^{-u}}{(1 + e^{-u})^2}.$$

Приклад. Задамо вектор входу  $\mathbf{u} = [0,1 \ 0,8 \ -0,7]^T$  логістичної ФА для шару із трьох нейронів і обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$\mathbf{y} = \text{logsig}(\mathbf{u})$	$\mathbf{d} = d\text{logsig}(\mathbf{u}, \mathbf{y})$
0,5250	0,2494
0,6900	0,2139
0,3318	0,2217

11. TANSIG – гіперболічна тангенціальна сигмоїдна ФА. Синтаксис:  $y = \text{tansig}(u)$ . Така послідовність команд будує графік ФА *tansig* (рис. А.8):

```
u = -3:0.1:3; y = tansig(u); plot(u,y).
```

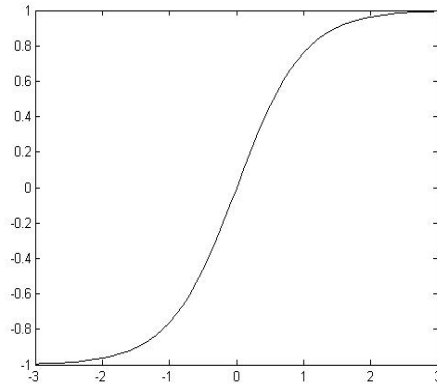


Рис. А.8. Графік ФА *tansig*

Функцію активації *tansig* та її похідну *dtansig* визначають таким чином:

$$tansig(u) = \frac{2}{1 + e^{-2u}} - 1; \quad dtansig(u) = \tilde{1} - tansig^2(u).$$

*Приклад.* Задамо вектор входу  $\mathbf{u} = [0,1 \quad 0,8 \quad -0,7]^T$  гіперболічної тангенціальної ФА для шару із трьох нейронів та обчислимо вектор виходу  $\mathbf{y}$  та похідну  $\mathbf{d}$ . Одержимо:

$\mathbf{y} = tansig(\mathbf{u})$	$\mathbf{d} = dtansig(\mathbf{u}, \mathbf{y})$
0,0997	0,9901
0,6640	0,5591
0,6044	0,6347

**Приклади реалізацій у середовищі MatLab****1. Процедура навчання нейрона типу персептрон.**

Визначити параметри нейрона, який виконує бінарну класифікацію відповідно до заданої функції  $y = \neg x_1 \wedge (x_2 \rightarrow x_3)$

```
% function perc
clc;close all;clear all;

for i=1:8
X(:,i)=bitget(i-1,3:-1:1)';
T(i)=~X(1,i) && (~X(2,i)||X(3,i));
end;
A=X'; A(:,4)=T;
disp (' y=~x1^(x2->x3) ');
disp ('  x1  x2  x3  y');
disp (A);

-rab=size(X')
for i=1:rab(1)
    inputs(1,i)=1;
inputs(2,i)=X(1,i); inputs(3,i)=X(2,i);
inputs(4,i)=X(3,i);
end;
% Initializatia
w=[0 0 0 0]';
targets=T;
% Aktivatia
s = 1; flag = [0 0 0 0 0 0 0 0];
k = 1; k1 = 1;
while(s > 0)
    for i = 1:rab(1)
        n = w'*inputs(:,i);
        a = hardlim(n);
        e = targets(i) - a;
        if(e ~= 0)
            dw = e*inputs(:, i);
            w = w + dw;
            flag(i) = 1;
        end
    end
    s = sum(flag);
    k1 = k1 + 1;
    k = k1;
end
```

```

else
    flag(i) = 0;
end;
training(k,:)= [k k1 w(1) w(2) w(3) w(4) inputs(1, i) a e];
k = k + 1;
end;
s = sum(flag);
k1 = k1 + 1;
end
w

```

Результат виконання програми для функції  $y = \neg x_1 \wedge (x_2 \rightarrow x_3)$ :

x1	x2	x3	y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

inputs =

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

w =

0
-3
-1
2

## 2. Алгоритм золотого перетину.

$\tau=0.618$

$c\_k = a\_k + (1-\tau) * (b\_k - a\_k)$  %k=1 – початкові значення, k1=k+1

$F\_c = F(c\_k);$

$d\_k = b\_k - (1-\tau) * (b\_k - a\_k); F\_d = F(d\_k);$



```

while (b_k1-a_k1>tol)
IF F_c<F_d THEN
  a_k1=a_k, b_k1=d_k, d_k1=c_k
  c_k1=a_k1+(1-τ)*(b_k1-a_k1),
  F_d=F(c_k); F_c=F(c_k1);
ELSE
  a_k1=c_k, b_k1=b_k, c_k1=d_k
  d_k1=b_k1+(1-τ)*(b_k1-a_k1),
  F_c=F_d; F_d=F(d_k1);
END;
END

```

Тут  $tol$  – встановлена користувачем точність.

### **3. Приклад алгоритму навчання мережі, яка враховує нечутливість «мертвих» нейронів.**

*Алгоритм.* На початку навчання усім нейронам конкурентного шару присвоюється однаковий коефіцієнт активності:  $c_0 = \frac{1}{N}$ , де  $N$  – кількість нейронів конкуруючого шару (дорівнює кількості кластерів).

У процесі навчання значення цього коефіцієнта збільшується для активних нейронів і зменшується для неактивних за формулою  $\Delta c = \alpha(y_i - c)$ , де  $\alpha$  – коефіцієнт швидкості навчання;  $y$  – вектор виходу конкуруючого шару:  $y_i = \begin{cases} 1, & \text{якщо } i = i^*, \\ 0, & \text{якщо } i \neq i^*, \end{cases}$  де  $i^*$  – номер нейрона-

переможця. Для всіх нейронів, окрім нейрона-переможця, прирости  $\Delta c < 0$ . Оскільки коефіцієнти активності пов'язані зі зсувами співвідношенням  $c = \exp(1 - \log(b))$ , то зсув нейрона-переможця зменшується, а зсуви інших нейронів збільшуються. Приріст вектора зсуву обчислюють за формулою  $\Delta b = \exp(1 - \log_2(c)) - b$ .

Якщо нейрон не виграє, то його зсув під час навчання стає достатньо великим, а нейрон – конкурентоспроможним. У цьому випадку його ваговий вектор починає наближатися до деякої групи векторів входу. Як тільки нейрон починає перемагати, його зсув зменшується. Таким чином, розв'язується задача активізації «мертвих» нейронів. Зсув вирівнює значення параметрів активності нейронів, вони притягують приблизно однакову кількість векторів входу (якщо один із кластерів притягує більшу кількість векторів, ніж інший, то область із великою кількістю векторів входу притягує додаткову кількість нейронів і буде поділена на менші за розміром кластери).

Лістинг програми мовою MatLab:

```
function competitive
clc; close all; clear all;
% [P(:,1) P(:,2)]=textread('met.txt','%f %f');
P1=[1 2 3 4 6 4 3 2 3 4 5 6 7 8 9 0]
P2=[2 4 1 3 5 7 3 5 2 5 7 8 8 6 4 4]
P=[P1; P2]'
%%Parameters Kohonen learning rate
KLR=0.01;
%Conscience learning rate
CLR=0.001;
numOfNeurons=7;
epochs=1000;
%%Initialization
min_max=mean([min(P,[],1)-1; max(P,[],1)+1],2);
W=[ones(numOfNeurons,1).*min_max(1),ones(numOfNeurons,1).*mi
n_max(2)];
c=ones(numOfNeurons,1)/numOfNeurons;
b=exp(1-log(c));
%%Train
Q=size(P,1);
for k=1:epochs
    for n=1:Q
        i=fix(rand*Q)+1;
        p=P(i,:)' ;
        z=ndist(W,p)+b;
        a=compet(z);
        db=learnb(b,a,CLR);
        b=b+db;
        dw=learnW(W,p,a,KLR);
        W=W+dw;
    end;
end
%%Results
drawPlot(P,W,'kS');
for k=1:size(P,1)
    simulate(P(k,:),W)
end;
```

```

function dw=learnW(w,p,a,lr)
[S,R]=size(w);
Q=size(p,2);
pt=p';
dw=zeros(S,R);
for q=1:Q
    i=find(a(:,q));
    dw(i,:)=dw(i,:)+lr*(pt(q+zeros(length(i),1),:)-w(i,:));
end
function db=learnb(b,a,lr)
[s,q]=size(a);
if q~=1, a=(1/q)*sum(a,2);end
%b->conscience
c=exp(1-log(b));
%update conscience
c=(1-lr)*c+lr*a;
%conscience->db
db=exp(1-log(c))-b;
function a=compet(n)
[S,Q]=size(n);
[maxn,indn]=max(n,[],1);
a=sparse(indn,1:Q,ones(1,Q),S,Q);
function z=ndist(w,p)
[S,R]=size(w);
[R2,Q]=size(p);
z=zeros(S,Q);
if (Q<S)
    p=p';
    copies=zeros(1,S);
    for q=1:Q
        z(:,q)=sum((w-p(q+copies,:)).^2,2);
    end
else
    w=w';
    copies=zeros(1,Q);
    for i=1:S
        z(i,:)=sum((w(:,i+copies)-p).^2,1);
    end
end
end

```

```

z=-z.^0.5;
function drawPlot(P,W,color)
hold on;
minV=min(P,[],1)-0.2
maxV=max(P,[],1)+0.2
axis([minV(1) maxV(1) minV(2) maxV(2)])
for k=1:size(P,1)
    plot(P(k,1),P(k,2),'*g');
end
for k=1:size(W,1)
    plot(W(k,1),W(k,2),color);
    plot(W(k,1),W(k,2),'yo', 'markersize',25)
    xos1=[W(k,1)-0.05, W(k,1)];
    yos1=[W(k,2)+0.05, W(k,2)];
    [x1,y1]=ds2nfu(xos1,yos1);
    class=sprintf('Class N%d',k);
    text(xos1(1)+0.05,yos1(1)+0.1,class,'FontSize',8)
end
function simulate(p,W)
a=ndist(W,p');
class=find(compet(a));
s=sprintf('This is class #0%d', class);
disp(s)
function varargout=ds2nfu(varargin)
if length(varargin{1})==1 && ishandle(varargin{1})...
    && strcmp(get(varargin{1},'type'),'axes')
    hAx=varargin{1};
    varargin=varargin(2:end);
else
    hAx=gca;
end;
if length(varargin)==1
    pos=varargin{1};
else
    [x,y]=deal(varargin{:});
end
axun=get(hAx,'Units');
set(hAx,'Units','normalized');
axpos=get(hAx,'Position');

```

```

axlim=axis(hAx);
axwidth=diff(axlim(1:2));
axheight=diff(axlim(3:4));
if exist('x','var')
    varargout{1}=(x-axlim(1))*axpos(3)/axwidth+axpos(1);
    varargout{2}=(y-axlim(3))*axpos(4)/axheight+axpos(2);
else
    pos(1)=(pos(1)-axlim(1))/axwidth*axpos(3)+axpos(1);
    pos(2)=(pos(2)-axlim(3))/axheight*axpos(4)+axpos(2);
    pos(3)=pos(3)*axpos(3)/axwidth;
    pos(4)=pos(4)*axpos(4)/axheight;
    varargout{1}=pos;
end
set(hAx,'Units',axun)

```

Результат виконання програми:

P =

1	2	- 1kl
3	4	- 7kl
6	4	- 3kl
3	2	- 6kl
3	4	- 5kl
5	6	- 2kl
7	8	- 6kl
9	0	- 7kl
2	4	- 3kl
1	3	- 6kl
5	7	- 2kl
3	5	- 4kl
2	5	- 4kl
7	8	- 5kl
8	6	- 5kl
4	4	- 1kl

## **Деякі положення векторної алгебри у контексті нейронних мереж**

Розглянемо основні поняття векторного простору: спочатку загальне визначення поняття «лінійний векторний простір» та основні властивості векторів, далі вектори-стовпці, які мають вигляд послідовності дійсних чисел із  $n$  елементів і позначаються як  $\mathbf{x} = [x_1 \dots x_n]^T$ . Вони утворюють простір  $\mathcal{R}^n$  (евклідов векторний простір розмірністю  $n$ ).

### **1. Лінійний векторний простір.**

Нехай  $F$  – деяка числова множина. *Лінійний векторний простір*  $X$  – це набір векторів, який задовольняє такі умови:

1. На  $X$  можна визначити операцію додавання векторів, тобто  $\forall \mathbf{x}, \mathbf{y} \in X: (\mathbf{x} + \mathbf{y}) \in X$ .

2. Умова комутативності:  $\forall \mathbf{x}, \mathbf{y} \in X: \mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$ .

3. Умова асоціативності:  $\forall \mathbf{x}, \mathbf{y} \in X: (\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$ .

4. Існує єдиний нульовий вектор  $\mathbf{0} \in X$  такий, що  $\forall \mathbf{x} \in X: \mathbf{x} + \mathbf{0} = \mathbf{x}$ .

5. За  $\forall \mathbf{x} \in X$  існує єдиний вектор  $(-\mathbf{x}) \in X$  такий, що:  $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$ .

6. На  $X$  можна визначити операцію множення, тобто  $\forall a \in F$  і  $\forall \mathbf{x} \in X: a\mathbf{x} \in X$ .

7.  $\forall \mathbf{x} \in X: 1\mathbf{x} = \mathbf{x}$  (де 1 – скаляр).

8.  $\forall a, b \in F$  і  $\forall \mathbf{x} \in X: a(b\mathbf{x}) = (ab)\mathbf{x}$ .

9.  $\forall a, b \in F$  і  $\forall \mathbf{x} \in X: (a + b)\mathbf{x} = a\mathbf{x} + b\mathbf{x}$ .

10.  $\forall a, b \in F$  і  $\forall \mathbf{x}, \mathbf{y} \in X: a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$ .

Перевіримо виконання цих умов для деяких векторних множин. Спочатку розглянемо стандартний двовимірний евклідов простір  $\mathcal{R}^2$  (рис. В.1, а), який є лінійним векторним простором, оскільки для нього виконуються усі 10 умов, далі – деякі підмножини простору  $\mathcal{R}^2$  й перевіримо, чи утворюють вони лінійний векторний простір.

Підмножину  $X_1$  простору  $\mathcal{R}^2$ , яка має вигляд квадрата із центром у початку координат, зображено на рис. В.1, б. Зазначимо, що  $X_1$  не є лінійним векторним простором, оскільки, як показано на рис. В.1, б, існують вектори  $\mathbf{x}, \mathbf{y} \in X_1$ , для яких  $(\mathbf{x} + \mathbf{y}) \notin X_1$ . Із цього прикладу можна зробити висновок, що обмежені множини не можуть утворювати лінійного векторного простору.

Підмножину  $X_2$  простору  $\mathcal{R}^2$ , яка має вигляд прямої, що проходить через початок координат, зображено на рис. В.1, в. Можна показати, що для  $X_2$  виконуються всі десять умов означення лінійного векторного

простору. Таким чином, будь-яка пряма в  $\mathbb{R}^2$ , що проходить через початок координат, є лінійним векторним простором. Зауважимо, що будь-які інші прямі простору  $\mathbb{R}^2$  не утворюють лінійного векторного простору, оскільки для них не виконується, наприклад, четверта умова.

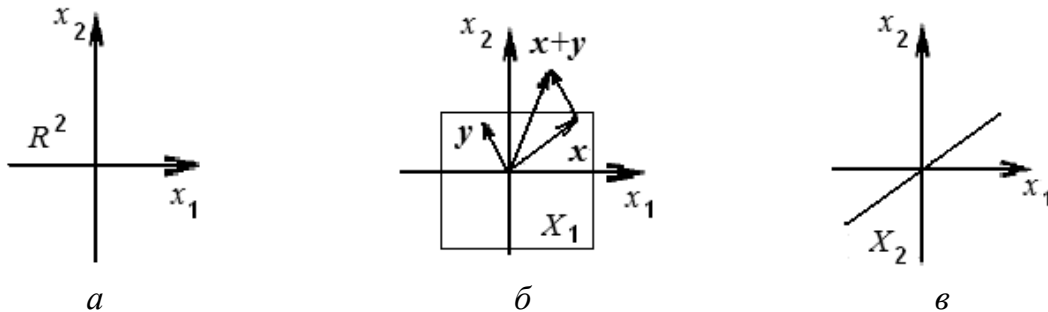


Рис. В.1. Приклади векторних просторів: *a* – двовимірний евклідов простір  $\mathbb{R}^2$ ; *б* – підмножина  $X_1$  простору  $\mathbb{R}^2$ ; *в* – підмножина  $X_2$  простору  $\mathbb{R}^2$

Крім евклідових просторів, існують інші множини, які утворюють лінійний векторний простір. Наприклад,  $P^2$  – множина багаточленів, степенів яких не перевищує двох. До цієї множини, приміром, належать такі елементи:  $(2 + t + 4t^2)$ ,  $(1 + 5t)$ . Множина  $P^2$  є лінійним векторним простором, оскільки для неї виконуються усі 10 умов відповідного означення.

Нехай  $C([0, 1])$  – множина функцій, неперервних на відрізку  $[0; 1]$ . До цієї множини, наприклад, належать функції  $\sin t$ ,  $e^{-2t}$ . Можна показати, що множина  $C([0, 1])$  є лінійним векторним простором.

## 2. Лінійна незалежність векторів.

Вектори  $\mathbf{x}_1, \dots, \mathbf{x}_n$  називають лінійно залежними, якщо існує набір скалярних значень  $a_1, \dots, a_n$ , серед яких принаймні одне ненульове і виконується рівність  $a_1\mathbf{x}_1 + \dots + a_n\mathbf{x}_n = 0$ . Якщо  $a_1\mathbf{x}_1 + \dots + a_n\mathbf{x}_n = 0$  лише коли всі  $a_i = 0$ , то вектори  $\mathbf{x}_1, \dots, \mathbf{x}_n$  називають лінійно незалежними. Зазначимо, що жоден вектор із множини лінійно незалежних векторів не можна записати у вигляді лінійної комбінації інших векторів цієї множини.

Розглянемо приклад лінійно незалежних векторів  $\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$ ;

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}.$$

Нехай  $a_1\mathbf{p}_1 + a_2\mathbf{p}_2 = 0$ , тоді  $\begin{bmatrix} a_1 + a_2 \\ -a_1 + a_2 \\ -a_1 + (-a_2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ , що виконується тільки за  $a_1 = a_2 = 0$ .

Розглянемо три елементи лінійного векторного простору  $P^2$  багаточленів, степінь яких не перевищує двох:  $\mathbf{x}_1 = 1 + t + t^2$ ;  $\mathbf{x}_2 = 2 + 2t + t^2$ ;  $\mathbf{x}_3 = 1 + t$ . Для значень  $a_1 = 1$ ;  $a_2 = -1$ ;  $a_3 = 1$  одержимо рівність  $a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + a_3\mathbf{x}_3 = 0$ . Отже, вектори  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$  є лінійно залежними.

### 3. Розмірність лінійного векторного простору.

Нехай  $F$  – числовий простір,  $F^n$  – відповідний арифметичний векторний простір. Тоді систему векторів  $\mathbf{u}_1, \dots, \mathbf{u}_k$  називають системою твірних простору  $F^n$ , якщо кожний вектор  $\mathbf{x} \in F^n$  лінійно виражається через  $\mathbf{u}_1, \dots, \mathbf{u}_k$ :  $\mathbf{x} = a_1\mathbf{u}_1 + \dots + a_k\mathbf{u}_k$ , де  $a_1, \dots, a_k$  – деякі скаляри.

Систему векторів  $\mathbf{u}_1, \dots, \mathbf{u}_k$  називають *базисом простору*  $X \subset F^n$ , якщо вона лінійно незалежна та є системою твірних простору  $F^n$ .

*Розмірністю векторного простору*  $X$  називають кількість векторів у його базисі.

Розглянемо лінійний векторний простір  $P^2$  багаточленів, степінь яких не перевищує двох. Можливий базис цього простору  $\mathbf{u}_1 = 1$ ;  $\mathbf{u}_2 = t$ ;  $\mathbf{u}_3 = t^2$ . Очевидно, що будь-який багаточлен, степінь якого не перевищує двох, можна подати у вигляді лінійної комбінації цих трьох векторів. Наприклад, багаточлен  $\mathbf{x} = 1 + 2t$  виражається через цей базис таким чином:  $\mathbf{x} = \mathbf{u}_1 + 2\mathbf{u}_2$ . Більше того, будь-які три лінійно незалежні вектори з  $P^2$  формують базис цього простору, тому, крім розглянутого, базисом простору  $P^2$  є також система багаточленів:  $\bar{\mathbf{u}}_1 = 1$ ;  $\bar{\mathbf{u}}_2 = 1 + t$ ;  $\bar{\mathbf{u}}_3 = 1 + t + t^2$ , а багаточлен  $\mathbf{x}$  виражається через нього як  $\mathbf{x} = -\bar{\mathbf{u}}_1 + 2\bar{\mathbf{u}}_2$ .

### 4. Скалярний добуток.

Нехай  $X$  – лінійний векторний простір,  $F$  – числова множина. Бінарна функція  $(\cdot, \cdot)$  є скалярним добутком, якщо:

1.  $\forall \mathbf{x} \in X: (\mathbf{x}, \mathbf{x}) \geq 0$  (рівність виконується тоді й тільки тоді, коли  $\mathbf{x} = \mathbf{0}$ ).

2.  $\forall \mathbf{x}, \mathbf{y} \in X: (\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})$ .

3.  $\forall a, b \in F$  і  $\forall \mathbf{x}, \mathbf{y}_1, \mathbf{y}_2 \in X: (\mathbf{x}, a \cdot \mathbf{y}_1 + b \cdot \mathbf{y}_2) = a(\mathbf{x}, \mathbf{y}_1) + b(\mathbf{x}, \mathbf{y}_2)$ .

Стандартний скалярний добуток у  $\mathbb{R}^n$  можна визначити таким чином:  $(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = x_1 y_1 + \dots + x_n y_n$ , але існують також інші формули його визначення.

### 5. Норма.

Нормованим простором називають лінійний векторний простір, в якому визначена функція  $\|\mathbf{x}\|$ , яку називають нормою  $\mathbf{x}$ . В основу поняття «норма» покладене поняття «довжина вектора».

Нехай  $X$  – лінійний векторний простір,  $F$  – числова множина.



Скалярну функцію  $\|\cdot\|$  називають нормою, якщо вона задовольняє такі властивості:

1.  $\forall \mathbf{x} \in X: \|\mathbf{x}\| \geq 0$ .
2.  $\forall \mathbf{x} \in X: \|\mathbf{x}\| = 0$ , тоді й тільки тоді, коли  $\mathbf{x} = \mathbf{0}$ .
3.  $\forall a \in F$  і  $\forall \mathbf{x} \in X: \|a \cdot \mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$ .
4.  $\forall \mathbf{x}, \mathbf{y} \in X: \|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ .

Існує багато функцій, які задовольняють зазначені умови. Одна з норм основана на скалярному добутку:  $\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})}$ . В евклідовому просторі  $\mathbb{R}^n$  таку норму можна обчислити за формулою  $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} = \sqrt{x_1^2 + \dots + x_n^2}$  (стандартна норма Евкліда). У нейронних мережах норму використовують для нормалізації вхідних векторів, тобто для отримання вхідних векторів з нормою «1». Використовуючи норму та скалярний добуток для лінійного векторного простору, розмірність якого більша від двох або дорівнює цьому значенню, можна визначити кут  $\theta$  між двома векторами  $\mathbf{x}$  і  $\mathbf{y}$ :  $\cos \theta = \frac{(\mathbf{x}, \mathbf{y})}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$ .

## 6. Ортогональність векторів.

Нехай  $X$  – лінійний векторний простір. Два вектори  $\mathbf{x}, \mathbf{y} \in X$  називають ортогональними, якщо  $(\mathbf{x}, \mathbf{y}) = 0$ . Розглянемо поняття «ортогональний простір».

Вектор  $\mathbf{x} \in X$  називають *ортогональним до підпростору  $X_1$  простору  $X$* , якщо  $\mathbf{x}$  є ортогональним до кожного вектора з  $X_1$  (позначають  $\mathbf{x} \perp X_1$ ).

Нехай  $X_1$  і  $X_2$  – підпростори простору  $X$ . Підпростір  $X_1$  є ортогональним підпростору  $X_2$ , якщо кожний вектор з  $X_1$  є ортогональним кожному вектору з  $X_2$  (позначають  $X_1 \perp X_2$ ). Два ортогональні підпростори простору  $\mathbb{R}^3$  зображено на рис. В.2.

## 7. Ортогоналізація Грамма–Шмідта.

Існує зв'язок між ортогональністю та незалежністю векторів.

Іноді виникає потреба перетворити множину незалежних векторів у множину ортогональних векторів, що утворює той самий векторний простір. Стандартну процедуру, яка виконує зазначене перетворення, називають *ортогоналізацією Грамма–Шмідта*.

Нехай маємо  $m$  незалежних векторів  $\mathbf{y}_1, \dots, \mathbf{y}_m$ , з яких потрібно одержати  $n$  ортогональних векторів  $\mathbf{v}_1, \dots, \mathbf{v}_n$ . Перший ортогональний вектор збігається з першим незалежним вектором:  $\mathbf{v}_1 = \mathbf{y}_1$ . Щоб одержати другий ортогональний вектор  $\mathbf{v}_2$ , відніmemo від  $\mathbf{y}_2$  вектор, спрямований у напрямку  $\mathbf{v}_1$ , тобто  $\mathbf{v}_2 = \mathbf{y}_2 - a\mathbf{v}_1$ , де  $a$  обирають таким чином,

щоб вектор  $\mathbf{v}_2$  був ортогональним до  $\mathbf{v}_1$ :  $(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1, \mathbf{y}_2 - a\mathbf{v}_1) = (\mathbf{v}_1, \mathbf{y}_2) - a(\mathbf{v}_1, \mathbf{v}_1) = 0$ , звідки  $a = \frac{(\mathbf{v}_1, \mathbf{y}_2)}{(\mathbf{v}_1, \mathbf{v}_1)}$ . Зазначимо, що  $a\mathbf{v}_1 = \frac{(\mathbf{v}_1, \mathbf{y}_2)}{(\mathbf{v}_1, \mathbf{v}_1)} \mathbf{v}_1$  є проекцією вектора  $\mathbf{y}_2$  на вектор  $\mathbf{v}_1$ . Якщо продовжити цей процес, то на  $k$ -му кроці одержимо  $\mathbf{v}_k = \mathbf{y}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{v}_i, \mathbf{y}_k)}{(\mathbf{v}_i, \mathbf{v}_i)} \mathbf{v}_i$ .

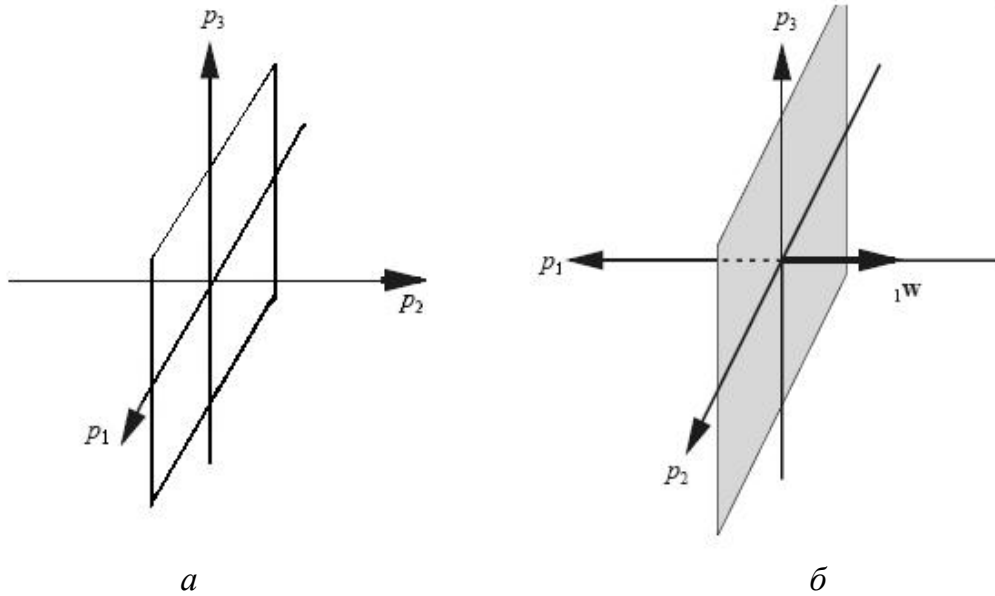


Рис. В.2. Приклади ортогональних векторів до підпросторів простору  $\mathbb{R}^3$  (осі  $p_1$  і  $p_3$  утворюють площину в просторі  $\mathbb{R}^3$ , яка є ортогональною до осі  $p_2$ ):  $a$  – вектор  $\mathbf{p}_2$  ортогональний до підпростору  $p_1 p_3$  простору  $\mathbb{R}^3$ ;  $b$  – ваговий вектор  $\mathbf{1w}$  ортогональний до підпростору  $p_2 p_3$  простору  $\mathbb{R}^3$

**Приклад В.1.** Розглянемо такі незалежні вектори в  $\mathbb{R}^2$ :

$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \mathbf{y}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}.$$

Визначимо перший ортогональний вектор:  $\mathbf{v}_1 = \mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . За методом ортогоналізації Грама – Шміда другий ортогональний вектор має

$$\text{вигляд (рис. В.3): } \mathbf{v}_2 = \mathbf{y}_2 - \frac{(\mathbf{v}_1, \mathbf{y}_2)}{(\mathbf{v}_1, \mathbf{v}_1)} \mathbf{v}_1 = \begin{bmatrix} -1 \\ 2 \end{bmatrix} - \frac{\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 2 \end{bmatrix}}{\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1,5 \\ 1,5 \end{bmatrix}.$$

Можна нормалізувати отримані вектори  $\mathbf{v}_1$  і  $\mathbf{v}_2$ , поділивши кожний вектор на його норму.

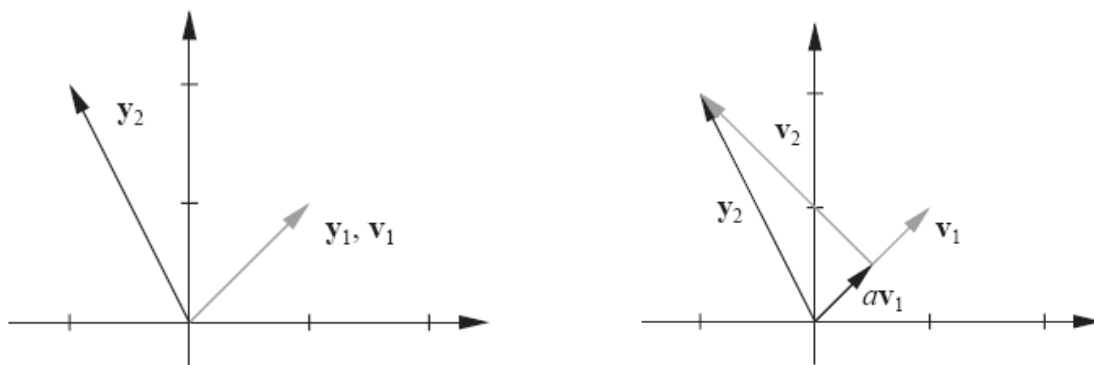


Рис. В.3. Графічний вигляд ортогоналізації Грамма–Шмідта до прикл. В.1

### 8. Розкладання векторів.

Покажемо, що будь-який вектор зі скінченновимірною лінійною векторного простору можна зобразити у вигляді стовпця чисел, тобто еквівалентно до векторів із простору  $\mathbb{R}^n$ . Нехай лінійний векторний простір  $X$  має базисну множину  $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ . Тоді кожний вектор  $\mathbf{x} \in X$  має єдиний векторний розклад:

$$\mathbf{x} = \sum_{i=1}^n a_i \mathbf{v}_i = a_1 \mathbf{v}_1 + \dots + a_n \mathbf{v}_n. \quad (\text{В.1})$$

Отже, кожному вектору із простору  $X$  можна поставити у відповідність вектор-стовпець чисел:  $\mathbf{a} = [a_1 \dots a_n]^T$ , де  $\mathbf{a}$  – зображення вектора  $\mathbf{x}$  у просторі  $\mathbb{R}^n$ . Зазвичай, щоб інтерпретувати значення  $\mathbf{a}$ , потрібно знати базисну множину. Зміна базисної множини зумовлює зміну вектора  $\mathbf{a}$ .

Опишемо цей процес більш детально. Спочатку розглянемо випадок, коли вектори базисної множини є ортогональними, тобто  $(\mathbf{v}_i, \mathbf{v}_j) = 0$ , якщо  $i \neq j$ . Застосуємо операцію «скалярний добуток на вектор  $\mathbf{v}_j$ »

до обох частин рівняння (В.1):  $(\mathbf{v}_j, \mathbf{x}) = \left( \mathbf{v}_j, \sum_{i=1}^n a_i \mathbf{v}_i \right) = \sum_{i=1}^n a_i (\mathbf{v}_j, \mathbf{v}_i) = a_j (\mathbf{v}_j, \mathbf{v}_j)$ . Отже, коефіцієнти розкладу вектора  $\mathbf{x}$  обчислюють за формулою

$a_j = \frac{(\mathbf{v}_j, \mathbf{x})}{(\mathbf{v}_j, \mathbf{v}_j)}$ ,  $j = 1, 2, \dots, n$ . Якщо вектори базисної множини не є ортогональними, то процес обчислення коефіцієнтів розкладання вектора за базисом є більш складним. Цей випадок розглянуто далі.

### 9. Обернені базисні вектори.

Нехай потрібно отримати розклад вектора за базисом, і вектори базисної множини не є ортогональними. Тоді розглядають обернені базисні вектори за допомогою рівнянь

$$\begin{aligned}(\mathbf{b}_i, \mathbf{v}_j) &= 0, \text{ якщо } i \neq j; \\(\mathbf{b}_i, \mathbf{v}_j) &= 1, \text{ якщо } i = j,\end{aligned}\tag{B.2}$$

де  $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  – базисні вектори;  $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$  – обернені базисні вектори.

Якщо вектори мають вигляд стовпців чисел, які визначаються з розкладу вектора за базисом, то, використавши стандартний скалярний добуток  $(\mathbf{b}_i, \mathbf{v}_j) = \mathbf{b}_i^T \mathbf{v}_j$ , рівняння (B.2) можна записати у матричній формі:  $\mathbf{B}^T \mathbf{V} = \mathbf{I}$ , де  $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n]$ ;  $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_n]$ ;  $\mathbf{I}$  – одинична матриця. Отже, матрицю  $\mathbf{B}$  можна визначити з рівняння  $\mathbf{B}^T = \mathbf{V}^{-1}$ , а обернені базисні вектори є стовпцями матриці  $\mathbf{B}$ .

Розглянемо розклад (B.1) вектора  $\mathbf{x}$  за базисом. Застосувавши операцію «скалярний добуток на вектор  $\mathbf{b}_1$ » до обох частин рівності (B.1), одержимо:  $(\mathbf{b}_1, \mathbf{x}) = a_1(\mathbf{b}_1, \mathbf{v}_1) + \dots + a_n(\mathbf{b}_1, \mathbf{v}_n)$ . За означенням обернених базисних векторів маємо  $(\mathbf{b}_1, \mathbf{v}_2) = \dots = (\mathbf{b}_1, \mathbf{v}_n) = 0$ ,  $(\mathbf{b}_1, \mathbf{v}_1) = 1$ . Отже, перший коефіцієнт розкладу  $a_1 = (\mathbf{b}_1, \mathbf{x})$ . Аналогічно у загальному випадку маємо:  $a_j = (\mathbf{b}_j, \mathbf{x})$ ,  $j = 1, 2, \dots, n$ .

Проілюструємо процес визначення обернених базисних векторів та коефіцієнтів розкладу вектора за базисом на прикладі.

**Приклад В.2.** Розглянемо два базисних вектори:  $\mathbf{v}_1^s = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ;  $\mathbf{v}_2^s = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ . Визначимо розклад вектора  $\mathbf{x}^s = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$  за базисом  $\{\mathbf{v}_1, \mathbf{v}_2\}$ . Зауважимо, що всі розглянуті вектори визначені у просторі  $\mathbb{R}^2$  зі стандартним базисом  $\{\mathbf{s}_1, \mathbf{s}_2\}$ , зображеним на рис. В.4, а.

Визначимо обернені базисні вектори:  $\mathbf{B}^T = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 \\ 0,5 & -0,5 \end{bmatrix}$ , звідки  $\mathbf{b}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ;  $\mathbf{b}_2 = \begin{bmatrix} 0,5 \\ -0,5 \end{bmatrix}$ . Тепер можна визначити коефіцієнти розкладу вектора  $\mathbf{x}$  за базисом  $\{\mathbf{v}_1, \mathbf{v}_2\}$ :  $a_1^v = \mathbf{b}_1^T \mathbf{x}^s = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 2 \end{bmatrix} = 2$ ;  $a_2^v = \mathbf{b}_2^T \mathbf{x}^s = \begin{bmatrix} 0,5 & -0,5 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 2 \end{bmatrix} = -1,5$ , або у матричній формі:  $\mathbf{x} = \mathbf{B}^T \mathbf{x}^s = \mathbf{V}^{-1} \mathbf{x}^s = \begin{bmatrix} 0 & 1 \\ 0,5 & -0,5 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ -1,5 \end{bmatrix}$ . Отже, маємо такі розклади

(рис. В.4 б):  $\mathbf{x}^s = (-1)\mathbf{s}_1 + 2\mathbf{s}_2$  – за базисом  $\{\mathbf{s}_1, \mathbf{s}_2\}$ ;  $\mathbf{x}^v = 2\mathbf{v}_1 - 1,5\mathbf{v}_2$  – за базисом  $\{\mathbf{v}_1, \mathbf{v}_2\}$ , або  $\mathbf{x}^s = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ ;  $\mathbf{x}^v = \begin{bmatrix} 2 \\ -1,5 \end{bmatrix}$ . Рівняння  $\mathbf{x}^v = \mathbf{V}^{-1}\mathbf{x}^s$  описує зв'язок між двома формами розкладу вектора  $\mathbf{x}$ :  $\mathbf{x}^v$  і  $\mathbf{x}^s$ .

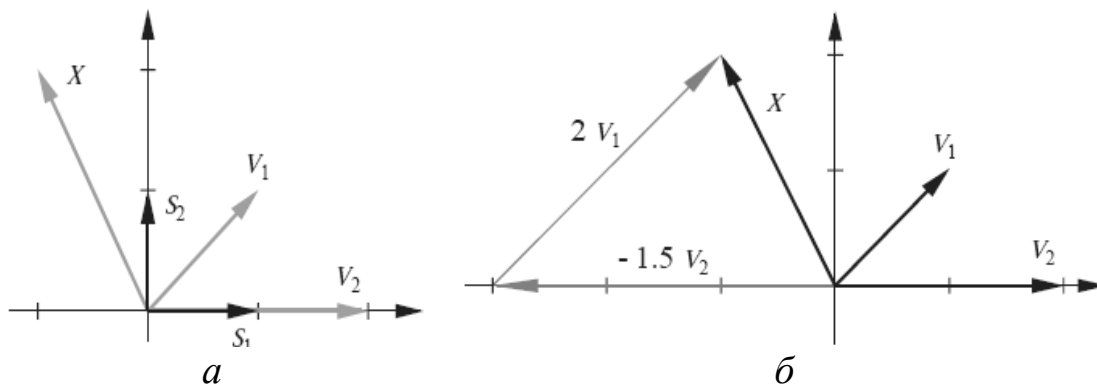


Рис. В.4. Графічна ілюстрація прикл. В.2:  
 а – базисні вектори; б – вектори розкладання

## Деякі положення функціонального аналізу в контексті нейронних мереж

Розкладання в ряд Тейлора – це засіб аналізу функціонала, що описує функціонування мережі, та визначення умов, яким повинні відповідати оптимальні точки цього функціоналу. Розглянемо приклади функціоналів, які описують функціонування мережі, та формулу розкладу в ряд Тейлора для визначення оптимальних точок цих функціоналів.

**1. Ряд Тейлора.** Нехай ефективність функціонування мережі – це функція  $F(x)$ , яку потрібно мінімізувати ( $x$  – скалярний параметр налаштування). Припустимо, що ефективність описується деякою аналітичною функцією, для якої існують всі її похідні. Тоді цю функцію в околі деякої точки  $x^*$  можна зобразити у вигляді *ряду Тейлора*:

$$F(x) = F(x^*) + \frac{d}{dx} F(x) \Big|_{x=x^*} (x - x^*) + \frac{1}{2} \frac{d^2}{dx^2} F(x) \Big|_{x=x^*} (x - x^*)^2 + \dots + \frac{1}{n!} \frac{d^n}{dx^n} F(x) \Big|_{x=x^*} (x - x^*)^n + \dots \quad (\text{Г.1})$$

**Приклад Г.1.** Нехай задано функцію  $F(x) = \cos(x)$ . Необхідно визначити апроксимацію  $F(x)$  в околі точки  $x^* = 0$ .

Ряд Тейлора для  $F(x)$  в околі точки  $x^* = 0$  визначають за формулою

$$\begin{aligned} \cos(x) &= \cos(0) - \sin(0)(x - 0) - \frac{1}{2} \cos(0)(x - 0)^2 + \\ &+ \frac{1}{6} \sin(0)(x - 0)^3 + \dots = 1 - \frac{1}{2} x^2 + \frac{1}{24} x^4 - \dots \end{aligned}$$

Виконаємо наближення (апроксимацію) функції  $F(x)$ : нульового порядку  $F_0(x) = 1$ ; першого порядку  $F_1(x) = F_0(x)$ ; другого порядку  $F_2(x) = 1 - \frac{1}{2} x^2$ ; третього порядку  $F_3(x) = F_2(x) = 1 - \frac{1}{2} x^2$ ; четвертого порядку  $F_4(x) = 1 - \frac{1}{2} x^2 + \frac{1}{24} x^4$ .

Функцію  $F(x)$  і всі три її наближення, які є точними, якщо точка  $x$  міститься дуже близько до точки  $x^* = 0$ , зображено на рис. Г.1, а. З віддаленням точки  $x$  від точки  $x^*$  лише наближення вищого порядку будуть точними. Зазначимо, що наближення другого порядку є точним на більш широкому інтервалі, ніж наближення нульового порядку, а наближення четвертого порядку є точним на ще більш широкому інтервалі, ніж наближення другого порядку, і т. д.

**Приклад Г.2.** Нехай задано функцію  $F(x) = e^{-x}$ . Необхідно визначити апроксимацію  $F(x)$  в околі точки  $x^* = 0$ .

Ряд Тейлора для  $F(x)$  в околі точки  $x^* = 0$  визначається за формулою

$$\begin{aligned} e^{-x} &= e^{-0} - e^{-0}(x-0) + \frac{1}{2}e^{-0}(x-0)^2 - \frac{1}{6}e^{-0}(x-0)^3 + \dots = \\ &= 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \dots \end{aligned}$$

Апроксимації функції  $F(x)$  мають такий вигляд (рис. Г.1, б): нульового порядку  $F_0(x) = 1$ ; першого порядку  $F_1(x) = 1 - x$ ; другого порядку  $F_2(x) = 1 - x + \frac{1}{2}x^2$ .

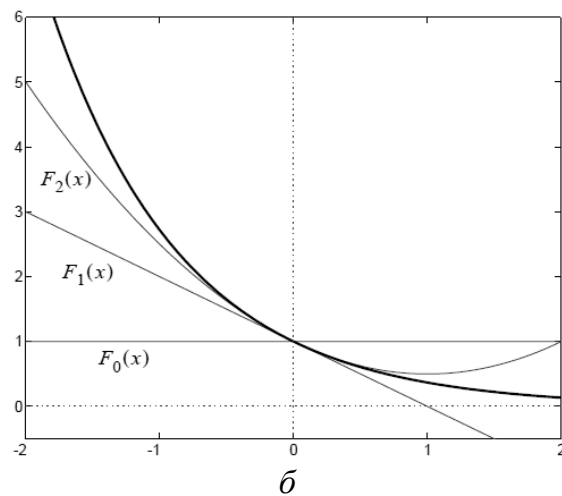
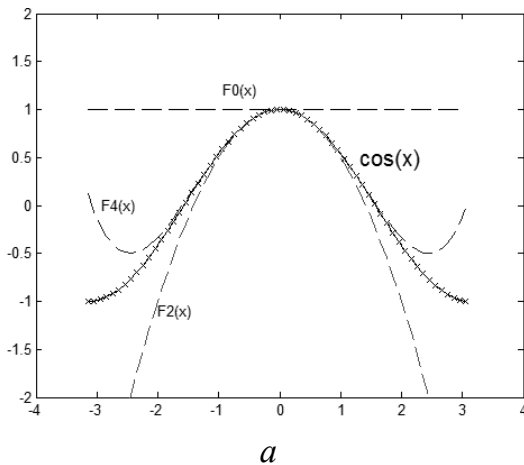


Рис. Г.1. Графіки функцій та їх апроксимацій за допомогою ряду Тейлора

**Векторний випадок.** Зазвичай показник ефективності функціонування НМ не є скалярною функцією, оскільки він має залежати від усіх параметрів мережі (матриці ваги та зсуву). Тому необхідно визначити ряд Тейлора для функції від багатьох змінних.

Ряд Тейлора для функції  $F(\mathbf{x}) = F(x_1, \dots, x_n)$  в околі точки  $\mathbf{x}^* = [x_1^* \dots x_n^*]^T$  має такий вигляд:

$$\begin{aligned} F(\mathbf{x}) &= F(\mathbf{x}^*) + \frac{\partial}{\partial x_1} F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*) + \frac{\partial}{\partial x_2} F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (x_2 - x_2^*) + \\ &+ \dots + \frac{\partial}{\partial x_n} F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (x_n - x_n^*) + \frac{1}{2} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)^2 + \\ &+ \frac{1}{2} \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*) (x_2 - x_2^*) + \dots \end{aligned}$$

Цей вираз можна записати у матричному вигляді:

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots$$

Тут  $\nabla F(\mathbf{x}) = \left[ \frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \dots \quad \frac{\partial}{\partial x_n} F(\mathbf{x}) \right]^T$  – градієнт функції  $F(\mathbf{x})$ ,  $\nabla^2 F(\mathbf{x})$  –

матриця Гессе, яка визначається таким чином:  $\nabla^2 F(\mathbf{x}) =$

$$= \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(\mathbf{x}) \\ \dots & \dots & \dots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}.$$

## 2. Похідні за напрямком.

Визначимо похідну функції в довільному напрямку. Нехай  $\mathbf{p}$  – вектор напрямку, за яким необхідно визначити похідну функції. Тоді похідну першого порядку функції  $F(\mathbf{x})$  у напрямку вектора  $\mathbf{p}$  можна обчислити за формулою

$$\frac{d}{d\mathbf{x}} F_{\mathbf{p}}(\mathbf{x}) = \frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}. \quad (\text{Г.2})$$

Похідну другого порядку функції  $F(\mathbf{x})$  у напрямку вектора  $\mathbf{p}$  можна знайти за формулою

$$\frac{d^2}{d\mathbf{x}^2} F_{\mathbf{p}}(\mathbf{x}) = \frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2}. \quad (\text{Г.3})$$

**Приклад Г.3.** Розглянемо функцію (рис. Г.2) вигляду  $F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2$ . Визначити похідну першого порядку функції  $F(\mathbf{x})$  у точці  $\mathbf{x}^* = [0,5 \ 0]^T$  у напрямку вектора  $\mathbf{p} = [1 \ 1]^T$ . Градієнт функції  $F(\mathbf{x})$  у точці  $\mathbf{x}^*$  має вигляд

$$\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} = \left[ \frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \right] \Big|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 2x_1 + 2x_2 \\ 2x_1 + 4x_2 \end{bmatrix} \Big|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Похідну першого порядку функції  $F(\mathbf{x})$  у напрямку вектора  $\mathbf{p}$  обчи-

слюють за формулою  $\frac{d}{d\mathbf{x}} F_{\mathbf{p}}(\mathbf{x}) = \frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|} = \frac{[1 \ -1] \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}}{\left\| \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\|} = \frac{0}{\sqrt{2}} = 0$ . Отже,



функція  $F(\mathbf{x})$  у точці  $\mathbf{x}^*$  має нульовий нахил у напрямку  $\mathbf{p}$  (функція має нульовий нахил у будь-якому напрямку, який є ортогональним до її градієнта). Максимальний нахил буває, коли напрямки векторів  $\mathbf{p}$  і градієнта  $\nabla F(\mathbf{x})$  однакові.

### 3. Точки мінімуму функції.

Існують різні алгоритми навчання, які відрізняються тим, що у процесі введення даних і функціонування мережі параметри мережі (вагові коефіцієнти та зсув) налаштовуються таким чином, щоб оптимізувати показник ефективності функціонування мережі. Оптимальною точкою називатимемо мінімум показника ефективності функціонування НМ (наведені далі означення можна легко змінити для задачі максимізації).

Нехай  $F: X \rightarrow \mathbb{R}$ , тоді точку  $x^*$  називають *точкою сильного локального мінімуму функції*  $F(x)$ , якщо існує такий окіл точки  $x^*$ , що для всіх точок  $x$  із цього околу виконується нерівність  $F(x^*) \leq F(x)$ . Точку сильного локального мінімуму іноді називають точкою локального мінімуму.

Точку  $x^*$  називають *точкою глобального мінімуму функції*  $F(x)$ , якщо  $\forall x \in X: F(x^*) < F(x)$ .

Точку  $x^*$  називають *точкою слабого локального мінімуму функції*  $F(x)$ , якщо вона не є точкою сильного локального мінімуму, та існує число  $\delta > 0$  таке, що  $\forall x \in X: 0 < \|x - x^*\| < \delta$ , тоді маємо  $F(x^*) < F(x)$ .

**Приклад Г.4.** Розглянемо функцію  $F(x) = 3x^4 - 7x^2 - 0,5x + 6$  (рис. Г.3), яка має два сильних локальних мінімуми в точках «-1,1» і «1,1», причому точка «1,1» є глобальним мінімумом. Функція  $F(x)$  не має слабого мінімуму.

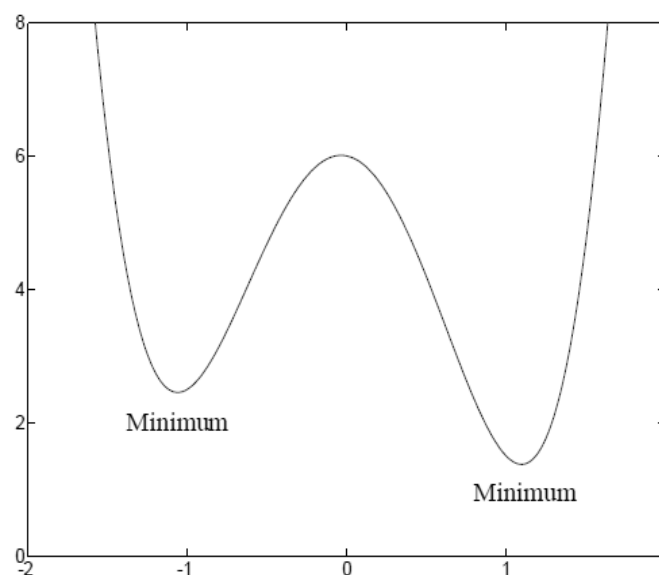
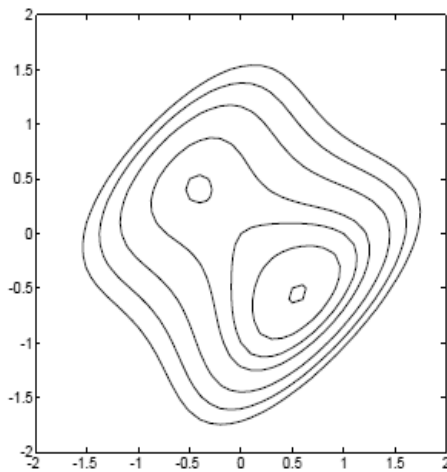
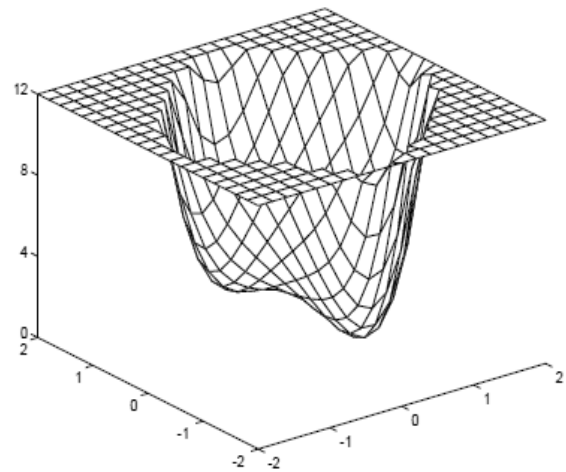


Рис. Г.3. Графік функції (до прикл. Г.4)

**Приклад Г.5.** Розглянемо функцію  $F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$ . На рис. Г.4, а зображено контурні лінії – криві, яким відповідає незмінне значення функції  $F(\mathbf{x})$ , а на рис. Г.4, б – 3D частина цієї функції, яка задовольняє умову  $F(\mathbf{x}) \leq 12$ . Функція  $F(\mathbf{x})$  має дві точки сильного локального мінімуму:  $(0,42; 0,42)$  і  $(0,55; 0,55)$ , причому точка  $(0,55; 0,55)$  є її глобальним мінімумом. Точка  $(0,13; 0,13)$  є сідловою: по лінії  $x_1 = x_2$  вона є локальним максимумом, а по ортогональній до неї лінії – локальним мінімумом.



а



б

Рис. Г.4. Графік функції (до прикл. Г.5)

**Приклад Г.6.** Розглянемо функцію  $F(\mathbf{x}) = (x_1^2 - 1,5x_1x_2 + 2x_2^2)x_1^2$ . Контурні лінії та 3D частина цієї функції зображено на рис. Г.5. Будь-яка точка на прямій  $x_1 = 0$  є точкою слабого мінімуму функції  $F(x)$ .

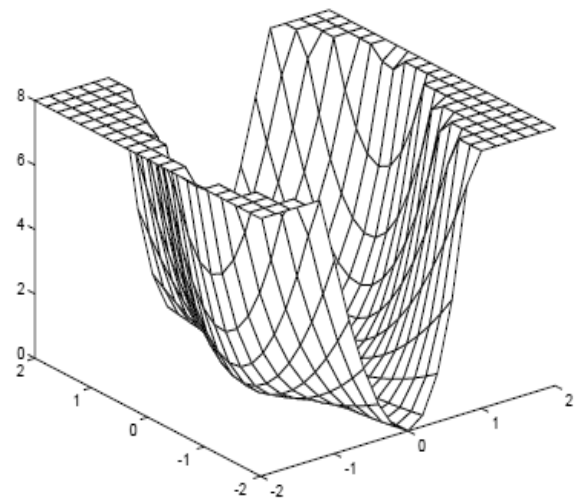
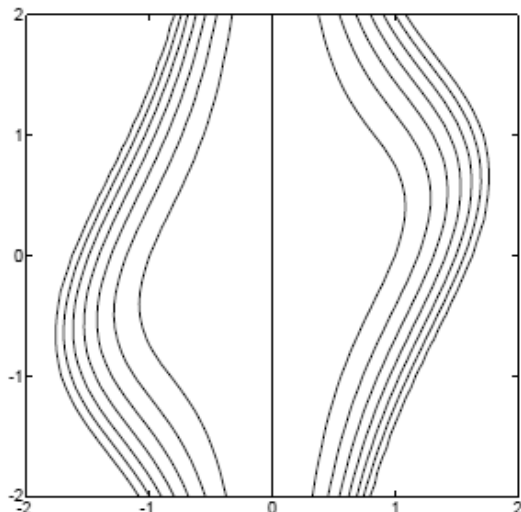


Рис. Г.5. Графік функції та точки її слабого мінімуму (до прикл. Г.6)

#### 4. Необхідні умови оптимізації.

Використавши ряд Тейлора, визначимо умови, які мають задовольняти точки мінімуму:

$$\begin{aligned} F(\mathbf{x}) &= F(\mathbf{x}^* + \Delta\mathbf{x}) = \\ &= F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} + \dots, \end{aligned} \quad (\Gamma.3)$$

де  $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$ .

*Умова першого порядку.* Якщо значення  $\|\Delta\mathbf{x}\|$  дуже мале, то функцію  $F(\mathbf{x})$  можна апроксимувати за формулою  $F(\mathbf{x}^* + \Delta\mathbf{x}) \cong F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x}$ . Якщо точка  $\mathbf{x}^*$  є точкою мінімуму, то функція  $F(\mathbf{x})$  повинна збільшуватися (або принаймні не зменшуватися) за  $\Delta\mathbf{x} \neq \mathbf{0}$ . Для цього необхідно, щоб другий доданок у рівнянні (Г.4) не був від'ємним, тобто:

$$\nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} \geq 0. \quad (\Gamma.5)$$

Проте, якщо цей вираз є додатним, тобто

$$\nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} > 0, \quad (\Gamma.6)$$

то  $F(\mathbf{x}^* - \Delta\mathbf{x}) \cong F(\mathbf{x}^*) - \nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} < F(\mathbf{x}^*)$ . Але ця умова не виконується, оскільки точка  $\mathbf{x}^*$  є мінімумом. Тому, оскільки рівняння (Г.5) виконується, а (Г.6) – ні, то єдиною альтернативою є умова  $\nabla F(\mathbf{x})^T \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} = 0$ , а оскільки  $\Delta\mathbf{x} \neq \mathbf{0}$ , то маємо

$$\nabla F(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0}. \quad (\Gamma.7)$$

Отже, градієнт функції повинен дорівнювати нулю в точці її мінімуму. Це є *першою необхідною умовою локального мінімуму*. Точки, які задовольняють рівняння (Г.7), називають *стаціонарними*. Приклади стаціонарних точок для функцій однієї змінної показано на рис. Г.6. Зазначимо, що стаціонарна точка не обов'язково є точкою екстремуму (рис. Г.6, б).

*Умови другого порядку.* Припустимо, що точка  $\mathbf{x}^*$  – стаціонарна. Оскільки градієнт функції  $F(\mathbf{x})$  дорівнює нулю в усіх стаціонарних точках, то розклад функції  $F(\mathbf{x})$  у ряд Тейлора в околі точки  $\mathbf{x}^*$  має вигляд

$$F(\mathbf{x}^* + \Delta\mathbf{x}) = F(\mathbf{x}^*) + \frac{1}{2} \Delta\mathbf{x}^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} + \dots \quad (\Gamma.8)$$

Для точок з околу точки  $\mathbf{x}^*$  норма  $\|\Delta\mathbf{x}\|$  набуває невеликих значень, тому функцію  $F(\mathbf{x})$  можна апроксимувати першими двома доданками рівняння (Г.8). Сильний мінімум у точці  $\mathbf{x}^*$  існуватиме, якщо  $\Delta\mathbf{x}^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{x}^*} \Delta\mathbf{x} > 0$ .

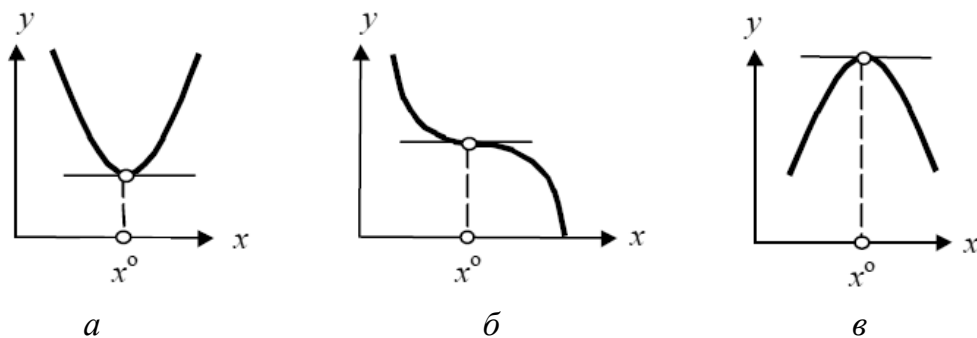


Рис. Г.6. Приклади стаціонарних точок функцій однієї змінної

Отже,  $\Delta \mathbf{x} \neq \mathbf{0}$ , якщо матриця Гессе є *додатно визначеною*. Нагадаємо, що матрицю  $\mathbf{A}$  називають *додатно визначеною*, якщо для будь-якого вектора  $\mathbf{z} \neq \mathbf{0}$ :  $\mathbf{z}^T \mathbf{A} \mathbf{z} > 0$ , і *додатно напіввизначеною*, якщо для будь-якого вектора  $\mathbf{z}$ :  $\mathbf{z}^T \mathbf{A} \mathbf{z} \geq 0$ . Зазначимо, що якщо всі власні значення матриці  $\mathbf{A}$  додатні, то матриця є *додатно визначеною*, а якщо невід'ємні, то *додатно напіввизначеною*. Таким чином, для точки  $\mathbf{x}^*$  – сильного мінімуму функції  $F(\mathbf{x})$  – виконуються:

1) *необхідні умови*:

$\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0}$  і матриця Гессе  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$  *додатно напіввизначена*;

2) *достатні умови*:

$\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0}$  і матриця Гессе  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$  *додатно визначена*.

**Приклад Г.7.** Розглянемо функцію  $F(\mathbf{x}) = x_1^4 + x_2^2$ . Необхідно визначити стаціонарні точки цієї функції.

Прирівняємо градієнт функції  $F(\mathbf{x})$  до нуля:  $\nabla F(\mathbf{x}) = \begin{bmatrix} 4x_1^3 \\ 2x_2 \end{bmatrix} = \mathbf{0}$ .

Отже, єдиною стаціонарною точкою функції  $F(\mathbf{x})$  є точка  $\mathbf{x}^* = [0 \ 0]^T$ . Перевіримо виконання умови другого порядку. Для цього визначимо

матрицю Гессе в точці  $\mathbf{x}^* = [0 \ 0]^T$ :  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix} \Big|_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$ .

Оскільки матриця Гессе є *додатно напіввизначеною*, то виконується необхідна умова сильного мінімуму для точки  $\mathbf{x}^* = \mathbf{0}$ . Але, незважаючи на те, що достатня умова сильного мінімуму для точки  $\mathbf{x}^* = \mathbf{0}$  не виконується, вона все одно є точкою сильного мінімуму.

**Приклад Г.8.** Розглянемо функцію  $F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$ . Необхідно визначити стаціонарні точки функції  $F(\mathbf{x})$ .

Прирівняємо градієнт функції  $F(\mathbf{x})$  до нуля:  $\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} = \mathbf{0}$ ,

звідки  $\mathbf{x}^* = [-1 \ 0,5]^T$  – стаціонарна точка. Перевіримо виконання умови другого порядку. Для цього визначимо матрицю Гессе функції  $F(\mathbf{x})$

у точці  $\mathbf{x}^*$ :  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$ . Покажемо, що ця матриця є додатно

визначеною. Для цього визначимо власні числа матриці Гессе:

$$|\nabla^2 F(\mathbf{x}) - \lambda \mathbf{I}| = \begin{vmatrix} 2-\lambda & 2 \\ 2 & 4-\lambda \end{vmatrix} = 0; \quad \lambda^2 - 6\lambda + 4 = (\lambda - 0,76)(\lambda - 5,24) = 0,$$

звідки  $\lambda_1 = 0,76$ ;  $\lambda_2 = 5,24$  – додатні. Таким чином, точка  $\mathbf{x}^*$  – сильний локальний мінімум.

## Основи теорії стійкості нелінійних динамічних систем у контексті нейронних мереж

**1. Концепція теорії стійкості [14].** Визначимо основи концепції теорії стійкості. Розглянемо *рух кульки* за наявності тертя у гравітаційному полі, а саме такі випадки:

1. Кульку розміщено на дні ями в точці  $y^*$  (рис. Д.1, а). Якщо перемістити кульку, то вона почне здійснювати коливальні рухи вперед і назад, але через тертя повернеться назад і зупиниться на дні ями. Назвемо цю позицію точкою *асимптотичної стійкості*.

2. Кульку встановлено в центрі плоскої поверхні (рис. Д.1, б). Якщо кульку перемістити, то вона не буде рухатися. Положення в центрі поверхні не є асимптотично стійким, оскільки у разі переміщення кулька не повертається назад до центру. Стійкість у цьому випадку називають *стійкістю за Ляпуновим*.

3. Кульку встановлено на вершині пагорба, яка є точкою рівноваги (рис. Д.1, в). Якщо на кульку здійснити хоча б найменший вплив, то вона скотиться вниз. Отже, маємо *нестійку точку рівноваги*.

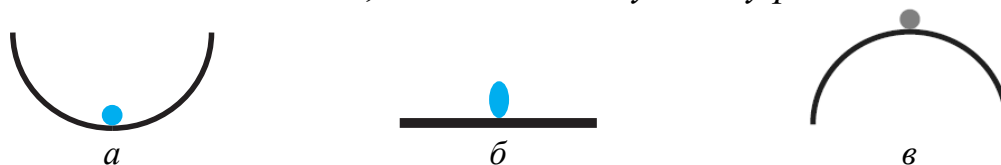


Рис. Д.1. Можливі види точок рівноваги

Розглянемо рис. Д.2: у випадку А кульку, яка котиться за наявності тертя з нульовою швидкістю, помістили у деяку западину, при цьому вона залишиться в цій западині та зупиниться на її дні (у точці стійкості); у випадку Б кульку помістили в точці  $p$ , при цьому важко визначити, в якій точці стійкості зупиниться кулька (кулька не зупиниться поблизу точки  $p$ ).

Дамо математичне визначення поняттю «стійкість».

*Точка рівноваги.* Говорять про стійкість нелінійної системи, описаної рівнянням

$$\frac{d}{dt}y(t) = g(y(t), p(t), t) \quad (\text{Д.1})$$

у точці рівноваги  $y^*$ , в якій  $\frac{dy}{dt} = 0$  (для простоти візьмемо точку  $y^* = 0$  – початок координат).

*Стійкість за Ляпуновим.* Початок координат є точкою стійкої рівноваги, якщо для будь-якого  $\varepsilon > 0$  існує число  $\delta(\varepsilon) > 0$  таке, що якщо  $\|y(0)\| < \delta$ , то результуючий рух (переміщення)  $y(t)$  задовольняє умову  $\|y(t)\| < \varepsilon$  для  $t > 0$ . Наведене визначення стверджує, що вихід системи не буде переміщуватися далеко від точки стійкості, якщо його початкове значення перебуває поблизу неї.

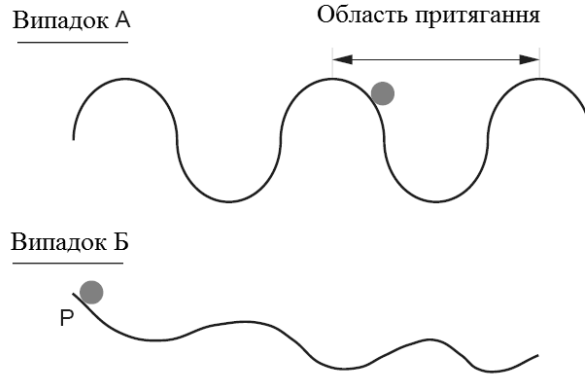


Рис. Д.2. Точки стійкості у западинах

Нехай необхідно, щоб вихід системи перемістився не більше, ніж на величину  $\varepsilon$  від початку координат. Якщо початок координат є стійким, то можна визначити відстань  $\delta$ , яка залежить від  $\varepsilon$ . Якщо відстань від виходу системи до початку координат не перевищує  $\delta$ , то у початковий момент часу  $t = 0$  вихід завжди буде перебувати в околі початку координат.

*Асимптотична стійкість.* Початок координат є асимптотично стійкою точкою рівноваги, якщо існує число  $\delta > 0$  таке, що якщо  $\|y(0)\| < \delta$ , то результуючий рух задовольняє умову  $\|y(t)\| \rightarrow 0$  за  $t \rightarrow \infty$ . Це більш точне визначення стійкості, яке стверджує: якщо вихід системи від самого початку не відходить від точки стійкості більше, ніж на  $\delta$ , то вихід є збіжним у точці стійкості.

Для аналізу стійкості використаємо такі визначення.

*Додатно визначена функція.* Скалярну функцію  $V(y)$  називають додатно визначеною, якщо  $V(0) = 0$  та  $V(y) > 0$  для  $y \neq 0$ .

*Додатно напіввизначена функція.* Скалярну функцію  $V(y)$  називають додатно напіввизначеною, якщо  $V(y) \geq 0$  для всіх  $y$ .

Наведені визначення можна модифікувати для від'ємно визначеної та від'ємно напіввизначеної функцій.

**2. Теорема Ляпунова про стійкість.** Одним із важливих підходів до дослідження стійкості нелінійних систем є теорія, запропонована Олександром Ляпуновим. Основна його робота вперше була опублікована 1892 р. і лише тепер привернула увагу науковців. Розглянемо

автономну систему, яка на пряму не залежить від часу:

$$\frac{dy}{dt} = \mathbf{g}(\mathbf{y}). \quad (\text{Д.2})$$

Можна вважати, що  $V(\mathbf{y})$  є функцією енергії. Теорема Ляпунова про стійкість стверджує, що якщо енергія системи постійно зменшується (тобто є від'ємно визначеною), то система перейде у стан, який відповідає мінімальній енергії. Метою Ляпунова було узагальнити концепцію енергії, застосувавши теорему до систем, в яких енергія не має значення або її важко визначити.

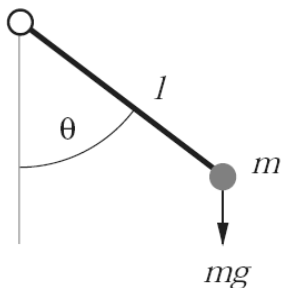


Рис. Д.3. Маятник

**Приклад Д.1.** Застосуємо теорему Ляпунова про стійкість до простої механічної системи маятника (рис. Д.3), яка ілюструє важливі концепції, що будуть застосовані до проектування мережі Хопфілда. Використовуючи другий закон Ньютона ( $F = ma$ ), функціонування маятника можна описати рівнянням

$$ml \frac{d^2}{dt^2}(\theta) = -c \frac{d\theta}{dt} - mg \sin(\theta),$$

або

$$ml \frac{d^2\theta}{dt^2} + c \frac{d\theta}{dt} + mg \sin \theta = 0, \quad (\text{Д.3})$$

де  $\theta$  – кут відхилення маятника;  $m$  – маса маятника;  $l$  – довжина плеча маятника;  $c$  – коефіцієнт тертя;  $g$  – прискорення вільного падіння.

Розглянемо елементи рівняння (Д.3):

1)  $\frac{d\theta}{dt}$  – сила тертя, яка пропорційна швидкості руху маятника (цей елемент зображує розсіювання енергії системою);

2)  $mg \sin \theta$  – сила гравітації, яка пропорційна синусу кута відхилення маятника (вона дорівнює нулю, коли маятник розміщений вертикально вниз, та набуває максимального значення, коли маятник розміщено горизонтально).

Якщо коефіцієнт тертя  $c \neq 0$ , то маятник зупиниться в нижньому вертикальному положенні. Розв'язок має вигляд  $\theta = 2\pi\eta$ , де  $\eta = 0, \pm 1, \pm 2, \dots$ . Тому за відповідних початкових умов маятник повинен перебувати в положенні  $\theta = 0$  або робити обертові рухи  $\theta = 2\pi$ .

Для аналізу стійкості цієї системи перетворимо рівняння маятника (Д.3) на рівняння з диференціалами першого порядку. Визначимо такі змінні стану:  $y_1 = \theta$ ;  $y_2 = \frac{d\theta}{dt}$ . Перепишемо рівняння маятника,



використавши ці змінні стану, отримаємо:

$$\frac{dy_1}{dt} = y_2; \quad (Д.4)$$

$$\frac{dy_2}{dt} = -\frac{c}{ml}y_2 - \frac{g}{l}\sin y_1. \quad (Д.5)$$

Дослідимо стійкість системи маятника на початку координат, який відповідає куту маятника  $\theta = 0$  і нульовій швидкості. Перевіримо, чи міститься початок координат у точці рівноваги, підставивши  $\mathbf{y} = \mathbf{0}$ , отримаємо:

$\frac{dy_1}{dt} = y_2 = 0$ ;  $\frac{dy_2}{dt} = -\frac{c}{ml}y_2 - \frac{g}{l}\sin y_1 = -\frac{g}{l}\sin 0 - 0 = 0$ . Оскільки  $\frac{d\mathbf{y}}{dt} = \mathbf{0}$ , то початок координат є точкою рівноваги. Визначимо ФЛ  $V(\mathbf{y})$ , використавши енергію системи маятника. Для визначення загальної енергії маятника додамо кінетичну та потенціальну енергію:

$$V(\mathbf{y}) = \frac{1}{2}ml^2(y_2)^2 + mgl(1 - \cos y_1), \quad (Д.6)$$

де  $y_1$  – кут відхилення маятника;  $y_2$  – кутова швидкість руху маятника. Для перевірки стійкості системи визначимо похідну ФЛ  $V(\mathbf{y})$  за часом:

$$\frac{d}{dt}V(\mathbf{y}) = [\nabla V(\mathbf{y})]^T \mathbf{g}(\mathbf{y}) = \frac{dV}{dy_1} \left( \frac{dy_1}{dt} \right) + \frac{dV}{dy_2} \left( \frac{dy_2}{dt} \right), \quad \text{звідки, враховуючи}$$

$$\text{вираз (Д.6), маємо: } \frac{d}{dt}V(\mathbf{y}) = (mgl \sin y_1)y_2 + (ml^2 y_2) \left( -\frac{g}{l} \sin y_1 - \frac{c}{ml} y_2 \right) =$$

$= -cl(y_2)^2 \leq 0$ . Оскільки похідна від'ємно напіввизначена, то початок координат є (щонайменше) асимптотично стійким за Ляпуновим. Похідна дорівнює нулю на початку координат, але також дорівнює нулю для будь-яких значень  $y_1$  за  $y_2 = 0$ . Оскільки похідна  $\frac{d}{dt}V(\mathbf{y})$  є від'ємно напів-

визначеною, то за теоремою Ляпунова початок координат є точкою стійкості. Однак, виходячи з теореми Ляпунова, не можна стверджувати, що початок координат є асимптотично стійким. У цьому випадку відомо, що завдяки силі тертя маятник зупиниться у вертикальному положенні, тому початок координат є асимптотично стійким.

Теорема Ляпунова лише стверджує, що початок координат є стійким. Щоб довести, що початок координат є асимптотично стійким, необхідно її вдосконалити. Теорема Ла–Салля є доопрацюванням теореми Ляпунова.

**Приклад Д.2.** Продовжимо досліджувати систему маятника. Нехай задано такі параметри:  $g = 9,8$ ;  $m = 1$ ;  $l = 9,8$ ;  $c = 1,96$ . Тоді рівняння

маятника має вигляд  $y_2 = \frac{dy_1}{dt}$ ;  $\frac{dy_2}{dt} = -0,2y_2 - \sin y_1$ . Функція Ляпунова  $V(\mathbf{y})$  та її похідна

$$V(\mathbf{y}) = 9,8^2 \left[ \frac{1}{2}(y_2)^2 + (1 - \cos y_1) \right], \quad \frac{dV(\mathbf{y})}{dt} = -19,208(y_2)^2. \quad (\text{Д.7})$$

Зазначимо, що  $\frac{d}{dt}V(\mathbf{y}) = 0$  для будь-яких значень  $\mathbf{y}$  за  $y_2 = 0$ . На рис. Д.4 зображено 3D і контурні графіки поверхні енергії  $V(\mathbf{y})$  маятника: значення  $y_1$  змінюються від  $-10$  до  $10$  радіан, а  $y_2$  – від  $-2$  до  $2$  радіан за секунду. В цьому діапазоні розміщені три можливі мінімуми ФЛ  $V(\mathbf{y})$  (як енергетичної поверхні), яка змінюється в інтервалі від  $2\pi$  до  $-2\pi$ .

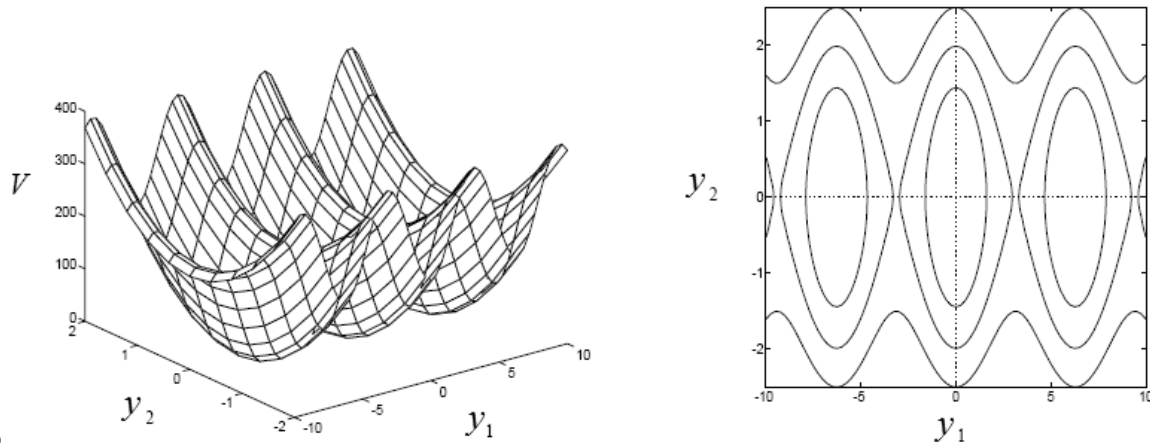


Рис. Д.4. Поверхня енергії  $V(\mathbf{y})$  маятника

Система маятника має багато точок мінімуму. Із графіка енергії (рис. Д.4) незрозуміло, яким чином маятник знаходить свій енергетичний мінімум. Тому один із можливих шляхів маятника відображено на графіку контурних ліній енергії (рис. Д.5). Відповідну траєкторію зображено лінією, яка розпочинається з позиції  $y_1(0) = 1,3$  радіана та має початкову швидкість  $y_2(0) = 1,3$  радіана за секунду. Траєкторія збігається в точці рівноваги  $\mathbf{y} = \mathbf{0}$ . Залежність двох змінних ( $y_1, y_2$ ) стану від часу  $t$  зображено на рис. Д.6. Зазначимо, що через додатну початкову швидкість маятник спочатку піднімається, досягши максимального відхилення на кут близько  $2$  радіан перед початком зворотної фази коливання (рух униз). Оскільки коливання загасають, то обидві змінні стану прямують до нуля (однак це не єдина можлива точка рівноваги), як і значення функції енергії маятника  $V(\mathbf{y})$ .

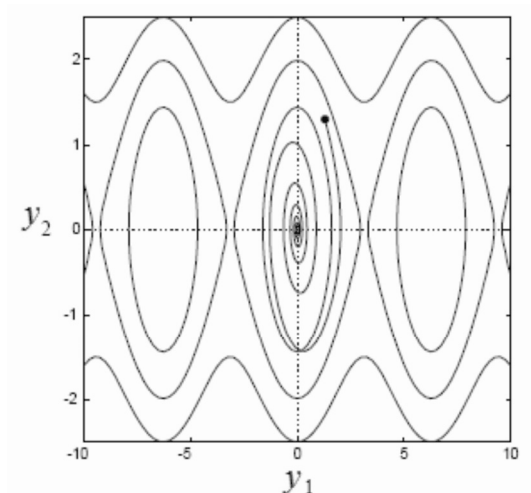


Рис. Д.5. Траєкторія маятника на фазовій площині за  $y_1(0) = 1,3$ ;  $y_2(0) = 1,3$

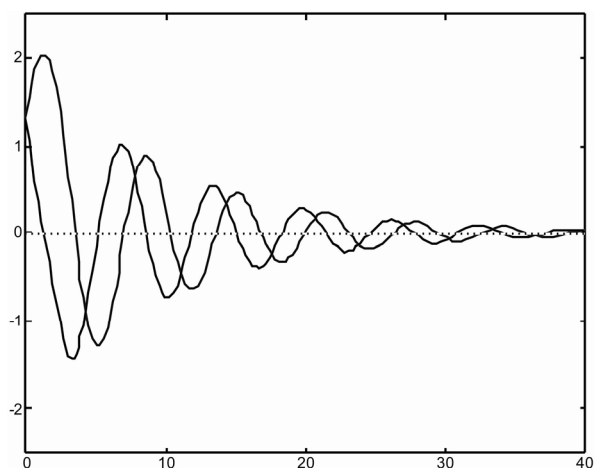


Рис. Д.6. Графіки залежності  $y_1$  та  $y_2$  від часу  $t$

Відповідно до рівняння (Д.7) енергія зростати не повинна. Це рівняння також показує, що похідна від енергії дорівнює нулю, коли кутова швидкість  $y_2 = 0$ . Зазначимо, що існують точки, в яких похідна функції енергії дорівнює нулю. Для визначення точок, в яких похідна ФЛ дорівнює нулю, застосовують інваріантну теорему Ла–Салля. Ця теорема також визначає, чи може система потрапити у ці точки. Точки, через які проходить траєкторія, називають *інваріантними множинами*. Якщо початок координат, в якому похідна ФЛ дорівнює нулю, належить траєкторії, то він є асимптотично стійким. Поведінка маятника залежить від початкових умов змінних стану  $y_1$  та  $y_2$ .

Приклад маятника показав неточності в теоремі Ляпунова: визначена ФЛ з від'ємно напіввизначеною похідною (не від'ємно визначеною), а початок координат є асимптотично стійким для системи маятника.

Розглянемо теорему, яка усуває ці неточності в теоремі Ляпунова через визначення областей простору, в яких похідна ФЛ дорівнює нулю, та виділення з них областей, які можуть належати траєкторії.

### 3. Інваріантна теорема Ла–Салля.

Перед тим, як розглянути інваріантну теорему Ла–Салля наведемо деякі визначення.

**Функція Ляпунова.** Нехай  $V: \mathbb{R}^n \rightarrow \mathbb{R}$  – неперервно диференційована функція,  $G \subset \mathbb{R}^n$ . Говорять, що  $V$  є *функцією Ляпунова на  $G$*  для системи  $\frac{dy}{dt} = g(y)$ , якщо

$$\frac{dV(y)}{dt} = (\nabla V(y))^T g(y) \quad (\text{Д.8})$$

не змінює знака на  $G$ . Це означення не вимагає додатної визначеності функції. Також немає чітких вимог до самої функції (окрім того, що вона має бути неперервно диференційованою). Є тільки одна вимога для похідної функції  $V$  – вона не може змінювати знак на множині  $G$ . Зазначимо, що похідна не змінюватиме знак, якщо вона від’ємно або додатно напіввизначена.

Щоб зрозуміти, як обрати множину  $G$ , розглянемо такі визначення:

1. *Множину  $Z$  визначають таким чином:*

$$Z = \left\{ \mathbf{y} : \frac{dV(\mathbf{y})}{dt} = 0, \mathbf{y} \in \overline{G} \right\}, \quad (\text{Д.9})$$

де  $\overline{G}$  (замикання  $G$ ) – це множина, яка включає внутрішні елементи і границю множини  $G$ . Згодом буде встановлено, в які точки множини  $Z$  може потрапити траєкторія функціонування системи.

2. *Інваріантна множина.* Множина точок із  $\mathbb{R}^n$  є інваріантною відносно  $\frac{d\mathbf{y}}{dt} = g(\mathbf{y})$ , якщо кожний розв’язок рівняння  $\frac{d\mathbf{y}}{dt} = g(\mathbf{y})$  починається у цій множині та ніколи не виходить за її межі. Потрапивши в інваріантну множину, система не може з неї вийти.

3. *Множина  $L$  визначена як найбільша інваріантна підмножина  $Z$ , яка включає всі можливі точки, до яких можуть прямувати розв’язки.* Функція Ляпунова не змінюється на множині  $L$ , бо її похідна дорівнює нулю, а траєкторія потрапить в  $L$ , оскільки це інваріантна множина. Якщо множина  $L$  має одну точку стійкості, то ця точка є асимптотично стійкою.

**Інваріантна теорема Ла–Салля** (узагальнення теореми Ляпунова про стійкість). Якщо  $V$  є ФЛ на множині  $G$  для  $\frac{d\mathbf{y}}{dt} = g(\mathbf{y})$ , то кожний розв’язок  $\mathbf{y}(t) \in G$  для всіх  $t > 0$  прямує до  $L^0 = L \cup \{\infty\}$  за  $t \rightarrow \infty$  ( $G$  – область притягання для множини  $L$ , якій належать всі точки стійкості).

Якщо всі траєкторії обмежені, то  $\mathbf{y}(t) \rightarrow L$  за  $t \rightarrow \infty$ . Якщо траєкторія залишається в  $G$ , то вона буде прагнути до  $L$  або до нескінченності. Якщо всі траєкторії обмежені, то вони будуть прямувати до  $L$ .

**Наслідок.** Нехай  $G$  – обмежена підмножина множини  $\Omega_\eta = \{\mathbf{y} : V(\mathbf{y}) < \eta\}$ ,  $\frac{dV(\mathbf{y})}{dt} \leq 0$  на множині  $G$  і множина  $L^0 = \overline{L \cap G}$  – підмножина  $G$ . Тоді  $L^0$  – атрактор (асимптотично стійка точка), і множина  $G$  розміщена в його області притягання.

Теорема Ла–Салля та її наслідок не лише можуть визначити, які точки є стійкими (формують  $L^0$ ), але також вказати область притягання  $G$ .

Розглянемо застосування теореми Ла–Салля на прикладі.

**Приклад Д3.** Застосуємо перший наслідок теореми до прикл. Д.1. Спочатку виберемо множину  $\Omega_\eta$ . Нехай  $\eta = 100$ , тоді  $\Omega_{100} = \{y: V(y) \leq 100\}$  є множиною точок (рис. Д.7), в яких енергія менше або дорівнює 100. Далі виберемо множину  $G$ , яка є підмножиною  $\Omega_{100}$ . Оскільки потрібно дослідити на стійкість початок координат, розглянемо підмножину  $G$  множини  $\Omega_{100}$ , яка містить  $y = 0$  (рис. Д.8). Оскільки з рівняння (Д.7) відомо, що похідна від ФЛ  $\frac{dV(y)}{dt}$  від'ємно напіввизначена, то вона буде меншою від нуля або дорівнювати йому на  $G$ . Тепер можна визначити множину атракторів. Визначимо множину  $L$ , яка є найбільшою інваріантною підмножиною множини  $Z$  (рис. Д.9) вигляду

$$Z = \left\{ y : \frac{dV(y)}{dt} = 0, y \in \overline{G} \right\} = \left\{ y : y_2 = 0, y \in \overline{G} \right\} = \left\{ y : y_2 = 0, -1,6 \leq y_1 \leq 1,6 \right\}.$$

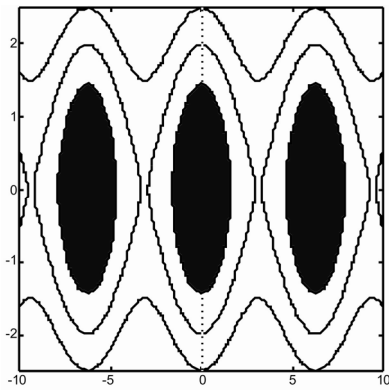


Рис. Д.7. Ілюстрація множини  $\Omega_{100}$

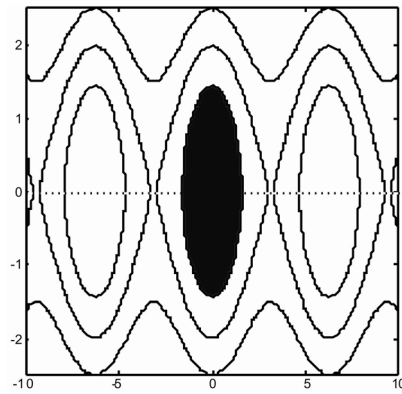


Рис. Д.8. Ілюстрація множини  $G$ , яка є підмножиною  $\Omega_{100}$

Із рівняння (Д.7) відомо, що рівність  $\frac{dV(y)}{dt} = 0$  виконується тільки тоді, коли швидкість маятника  $y_2 = 0$ . Тому множина  $Z$  складається з перетину осі  $y_2$  із множиною  $G$ . Для визначення множини  $L$  необхідно відповісти на запитання, чи залишиться швидкість маятника нульовою, якщо його запустити з точки, розміщеної між  $-1,6$  та  $1,6$  радіан з нульовою початковою швидкістю. Таке початкове положення збігається з точкою  $y = 0$ . Якщо запустити маятник з будь-якої іншої точки множини  $Z$ , то він почне падати, його швидкість вже не буде нульовою, а траєкторія вийде за межі  $Z$ . Тому множина  $L$  складається тільки з початку координат:  $L = \{y : y = 0\}$ .

Множина  $L^0$  є замиканням перетину множин  $L$  і  $G$ , яке у цьому випадку дорівнює множині  $L$ :  $L^0 = \overline{L \cap G} = L$ . Згідно з наслідком тео-

реми Ла–Салля  $L^0$  – аттрактор (асимптотично стійка точка) і  $G$  – множина притягання. Тому будь-яка траєкторія, яка розпочинається в  $G$ , буде прямувати до початку координат.

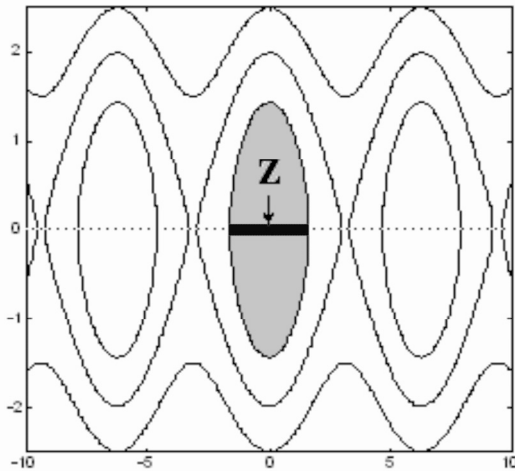


Рис. Д.9. Множина  $L = \{y: y = 0\}$

Зазначимо, що якщо  $V_1$  і  $V_2$  – ФЛ на  $G$ , похідні  $\frac{dV_1}{dt}$ ,  $\frac{dV_2}{dt}$  мають однаковий знак, то  $V = V_1 + V_2$  – також ФЛ і  $Z = Z_1 \cap Z_2$ . Якщо множина  $Z$  менша, ніж  $Z_1$  або  $Z_2$ , то  $V$  – краща ФЛ, ніж  $V_1$  або  $V_2$ . Отже, якщо є дві різні ФЛ, похідні яких мають однаковий знак, то їхня сума також є ФЛ, що має найменший аттрактор і найбільшу область притягання.

Основою теореми Ла–Салля є вибір ФЛ  $V$  і множини  $G$ . Необхідно, щоб область  $G$  була якомога більшою, оскільки вона визначає область притягання. Проте  $V$  потрібно вибрати так, щоб множина  $Z$ , яка містить аттрактор множини  $G$ , була якомога меншою.

Отже, розглянуто основні положення теорії стійкості динамічних систем, серед яких дві головні теореми:

1. Теорема Ляпунова про стійкість, яка є концепцією узагальненої енергії у вигляді ФЛ та стверджує, що якщо енергія системи постійно зменшується, то система зупиниться в мінімальній точці енергії.

2. Інваріантна теорема Ла–Салля є вдосконаленням теореми Ляпунова.

## Список літератури

1. Барцев С. И. Адаптивные сети обработки информации / С. И. Барцев, В. А. Охонин. – Красноярск : Ин-т физики СО АН СССР, 1986. – Препринт № 593. – 20 с.
2. Барцев С. И. Принцип двойственности в организации адаптивных сетей обработки информации / С. И. Барцев, С. Е. Гилев, В. А. Охонин // в кн.: Динамика химических и биологических систем. – Новосибирск : Наука, 1989. – С. 6–55.
3. Галушкин А. И. Синтез многослойных систем распознавания образов / А. И. Галушкин. – М. : «Энергия», 1974 – 368 с.
4. Горбань А. Н. Обучение нейронных сетей / А. Н. Горбань. – М. : СП ПараГраф, 1990. – 247 с.
5. Добровська Л. М. Штучний інтелект. Основи теорії нейронних мереж: метод. вказівки до практ. занять для студ. спец. «Інформаційні управляючі системи та технології» міжуніверситетськ. медико-інженер. ф-ту / Уклад. : Л. М. Добровська, І. А. Добровська. – Київ: НТУУ «КПІ», 2009. – 180 с.
6. Люгер Джордж Ф. Искусственный интеллект: стратегии и методы решения сложных проблем / Джордж Ф. Люгер; пер. с англ. – 4-е изд. – М. : Изд. дом «Вильямс», 2003. – 864 с.
7. Новотарський М. А. Штучні нейронні мережі: Обчислення / М. А. Новотарський, Б. Б. Нестеренко; Ін-т Математики НАН України. – Київ, 2004. – 407 с.
8. Миркес Е. М. Нейрокомпьютер. Проект стандарта / Е. М. Миркес. – Новосибирск : Наука, Сибирская изд. фирма РАН, 1999. – 337 с.
9. Осовский С. Нейронные сети для обработки информации / С. Осовский; пер. с польск. – М. : Финансы и статистика, 2004. – 343 с.
10. Пегат А. Нечеткое моделирование и управление / А. Пегат; пер. с англ. – М. : БИНОМ. Лаборатория знаний (Адаптивные и интеллектуальные системы), 2009. – 798 с.
11. Самойленко М. І. Математичне програмування: навч. посіб / М. І. Самойленко. – Харків : Основа, 2001. – 424 с.
12. Хайкин С. Нейронные сети: полный курс / С. Хайкин; пер. с англ. – 2-е изд. – М. : Изд. дом «Вильямс», 2006. – 1104 с.
13. Jacobs R. A. Increased rates of convergence through learning rate adaptation / R. A. Jacobs // In Proceedings of Neural Networks. – 1988 – P. 295–307.
14. Hagan Martin T. Neural Network Design / T. Martin Hagan, B. Howard Demuth, Mark Beale. – USA : Colorado University Bookstore, 2002. – 734 p.

15. Madan M. Gupta Statistic and Dynamic Neural Networks: From Fundamentals to Advanced Theory / M. Gupta Madan, Jin Liang, Homma Noriyasu. – USA : IEEE Press, 2003. – 722 p.
16. Rumelhart D. E. Learning Internal Representations by Error Propagation / D. E. Rumelhart, G. E. Hinton, R. J. Williams // In Parallel Distributed Processing. – Vol. 1. – Cambridge, MA, MIT Press. 1986. – P. 318–362.
17. Tollenaere T. SuperSAB: Fast adaptive back propagation with good scaling properties / T. Tollenaere // In Proceedings of Neural Networks. – 1990. – P. 561–573.
18. Wasserman P. D. Experiments in translating Chinese characters using backpropagation / P. D. Wasserman // Proceedings of the Thirty-Third IEEE Computer Society: International Conference – Washington : D. C. : Computer Society Press of the IEEE, 1988. – P. 399 – 402.
19. Werbos P. J. Beyond regression: New tools for prediction and analysis in the behavioral sciences / P. J. Werbos, Ph.D. thesis. – Harvard University, Cambridge, MA. – 1974.



## Предметний покажчик

<b>А</b>		<b>Г</b>	
Адаптивне фільтрування,	159	Графік сигмоїдної	
Аксон	13	функції активації	18
Алгоритм зворотного		– функції активації	
поширення	184	з різким порогом	17
– Левенберга–Марквардта		– лінійної функції активації	17
зі зворотним зв'язком	209		
–самоорганізації мап	297	<b>Д</b>	
– LMS	154	Дендрити	13
– відбору найбільш значимих			
базисних функцій	242	<b>З</b>	
– методу спряжених		Зведення лінійних перетворень	
градієнтів	128	до діагонального вигляду	72
– навчання персептрона	50	Зображення знань на основі НМ	28
Алгоритми навчання		<b>І</b>	
радіальних мереж		Інтегратор	321
на основі функцій Гаусса	234	<b>К</b>	
– оптимізації показника		Квазіінтегратор	322
ефективності функціонування		Класифікація НМ	27
мережі	117	Колатерали	13
Аналіз ефективності		Конкурентні нейронні мережі	279
навчання Хебба	92	<b>Л</b>	
Архітектура		Лінійна одношарова мережа	151
багатошарової мережі	21	Лінійний асоціатор	89
– нейрона типу персептрон	40	Лінійні перетворення НМ	63
– одношарової мережі	40	<b>М</b>	
Асоціативне контрольоване		Матриця Якобі	182
навчання Хебба	89	Матричне зображення лінійних	
– неконтрольоване навчання	250	перетворень	64
Асоціація	250	Мережа ART1	344
<b>Б</b>		– Гроссберга	318
Багатошаровий персептрон	177	– Хопфілда	371
Біологічний нейрон	12	– квантування векторів LVQ	298
<b>В</b>		– Хопфілда дискретного	
Включення апріорної інформації		часу	373
в структуру мережі	33	– Хопфілда неперервного	
– інваріантів у структуру		часу	376
мережі	34	«Мертві» нейрони	289
			393

Метод з використанням імпульсу	203
– із застосуванням змінного значення коефіцієнта швидкості навчання	204
– найшвидшого спуску	118
– Ньютона	123
– ортогоналізації найменших квадратів	240
– спряжених градієнтів	126
Методи визначення оптимальних точок	117
– підбору кількості базисних функцій	238
Модель нейрона з $R$ входами	19
– нейрона з одним входом	17
– нейрона Маккаллока–Піткса	26
– Хопфілда як асоціативна пам'ять	384
– штучного нейрона	16
Модифікації методу зворотного поширення	201

## Н

Нейрон з одним входом	16
– із декількома входами	18
Нейронна мережа Хеммінга	280
Нейронні мережі на основі радіальних базисних функцій	226
Неконтрольоване правило навчання Хебба	253
Нелінійна шунтуюча модель	321

## О

Одношарова мережа Оутстар	264
– мережа Інстар	258
Оптимізація характеристик функціонування мережі	111
Особливості лінійних мереж	164

## П

Персептрон	40
– Ф. Розенблатта	51
Перша модель штучної НМ	25
Постулат навчання Хебба	89

Правило Кохонена	263
– дельти	98
– навчання Хебба зі зменшенням ваги	256
– навчання Оутстар	264
– навчання нейрона Інстар	258
Принципи функціонування однеї нейронного персептрона	42
Прості асоціативні мережі	251
Процедура навчання одношарового персептрона	44
Псевдообернене правило	94

## Р

Робастість	10
------------	----

## С

Середньоквадратична похибка мережі АДАЛІН	153
Синапс	13
Сингулярне розкладання	229
Стратегії навчання	22
Схема персептрона	41

## Ш

Шар і мапа Кохонена	288
---------------------	-----

Навчальне видання

**Добровська Людмила Миколаївна**  
**Добровська Ірина Арбиківна**

## **Теорія та практика нейронних мереж**

**Навчальний посібник**

Редагування і коректура *Н. В. Мурашової*  
Комп'ютерне верстання *А. М. Мушницький*

Темплан 2015 р., поз. 1-2-019

Підп. до друку 28.10.2015. Формат 60×84<sup>1</sup>/<sub>16</sub>. Папір офс. Гарнітура Times.  
Спосіб друку – ризографія. Ум. друк. арк. 23,02. Обл.-вид. арк. 38,28. Наклад 55 пр. Зам. № 15-203.

---

Національний технічний університет України  
«Київський політехнічний інститут»  
Видавництво «Політехніка»  
Свідоцтво ДК № 1665 від 28.01.2004 р.  
03056, Київ, вул. Політехнічна, 14, корп. 15  
тел. (44) 406-81-78

[illegible]