НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

кафедра автоматика та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ О. І. РОЛІК
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 «Інформаційні системи та технології»
(код і назва спеціальності)

на тему: Роботизований комплекс точкового оброблення гербіцидами сільськогосподарських угідь

Виконав : студент __6_ курсу, групи ____ІА–82мпв_____
(шифр групи)

Блінніков Богдан Вадимович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н., доц. Писаренко А.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки
(повна назва)
Кафедра автоматики та управління в технічних системах
(повна назва)
Ступінь вищої освіти –другий (магістерський)
(код, назва)
Спеціальність 126 «Інформаційні системи та технології»
(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О. І. РОЛІК
(підпис) (ініціали, прізвище)

“___”_____2019_р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Бліннікову Богдану Вадимовичу

(прізвище, ім’я, по батькові)

1.**Тема дисертації** Роботизований комплекс точкового оброблення гербіцидами сільськогосподарських угідь_____
Науковий керівник дисертації_к.т.н., доц. Писаренко Андрій Володимирович
затверджені наказом по університету від “_29_” _жовтня_2019_р.№_____

2.Строк подання студентом дисертації “_21_ ” грудня 2019_р.

3. Об`єкт дослідження: оброблення сільськогосподарських угідь гербіцидами

4.Зміст пояснювальної записки**:** а) огляд предметної області та аналогів; б) методи розпізнавання об’єктів; в) розроблення функціональної схеми підсистеми робота; г) розроблення структурної схеми роботи системи; д) опис алгоритму роботи підсистеми робота; е) моделювання підсистеми розпізнавання рослин; ж) розроблення стартап-проекту.

5. Перелік графічного (ілюстративного) матеріалу: Схема електрична структурна; Схема електрична функціональна підсистеми робота; Схема алгоритму роботи підсистеми робота.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання "_29_" __жовтня__2019_р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Огляд і аналіз існуючих рішень	02.10.2018	
2	Алгоритми реалізації під-регулятора	15.10.2018	
3	Досліди по реалізації	29.10.2018	
4	Реалізація	06.11.2018	
5	Стартап-проект	20.11.2018	
6	Оформлення текстової та графічної документації	2.12.2018	
7	Представлення до захисту	4.12.2018	

Студент _____ Блінніков Б.В.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Писаренко А.В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему «Роботизований комплекс точкового оброблення гербіцидами сільськогосподарських угідь». 113 с., 43 рис., 24 таб., 8 додатків, 36 джерел.

В роботі отримано нове вирішення актуальної науково-практичної задачі підвищення ефективності сільськогосподарського виробництва шляхом розроблення роботизованого комплексу точкового оброблення гербіцидами сільськогосподарських угідь, розроблено математичну модель системи розпізнавання рослин, спроектовано логічно-фізичну систему робота.

Значну увагу в роботі приділено моделюванню системи розпізнавання паростків рослин за допомогою програмного комплексу Matlab/Simulink.. Результати цього моделювання підтвердили правильність розробленої системи розпізнавання.

Ключові слова: моделювання, розпізнавання об’єктів, робот

SUMMARY

Master's dissertation of educational-qualifying level of "Master" on " Robotic complex of point cultivation with herbicides of agricultural lands". 113 pgs., 43 fig., 24 tables, 8 applications, 36 sources.

The paper presents a new solution to the actual scientific and practical problem of increasing the efficiency of agricultural production by developing a robotic complex of point cultivation with herbicides, developed a mathematical model of the system of plant recognition, designed a logical and physical system of work.

Much attention is paid to the modeling of plant sprout recognition system using Matlab/Simulink software. The results of this simulation confirmed the correctness of the developed recognition system.

Keywords: modeling, object recognition, robot

Оглавление

ВСТУП 7
1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ 8
1.1 Система аналізу ґрунту Adigo Field Flux Robot 11
1.2 Система спостереження Ladybird 12
1.3 Аналізатор Rosphere 12
1.4 Vitirover 13
1.5 Огляд маніпуляторів 14
1.6 Технічні вимоги 15
2 МЕТОДИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ 16
2.1 Машинне навчання 18
2.1.1 Завдання типу T 19
2.1.2 Міра ефективності типу P 24
2.1.3 Знання типу E 25
2.1.4 Лінійна регресія 28
2.1.5 Ємність, перенавчання та недонавчання 32
2.1.6 Керовані алгоритми навчання 40
2.1.7 Некеровані алгоритми навчання 48
2.1.8 Побудова алгоритму машинного навчання 55
2.2 Глибоке навчання 57
2.2.1 Deep Feedforward Networks 58
2.2.2 Дизайн архітектури 61
2.2.3 Алгоритми зворотного поширення та інші диференціації 68
2.2.4 Обчислювальні графи 69
3 РОЗРОБЛЕННЯ СТРУКТУРИ СИСТЕМИ 72
3.1 Розроблення структурної схеми системи 72
3.2 Розроблення функціональної схеми підсистеми робота 73
3.3 Розроблення алгоритму роботи підсистеми робота 74
3.3.1 Алгоритм запуску 74
3.3.2 Алгоритм основного циклу 75
3.3.3 Алгоритм завершення роботи 77
4 МОДЕЛЮВАННЯ ПІДСИСТЕМИ РОЗПІЗНАВАННЯ РОСЛИН 78

4.1 Розроблення моделі підсистеми у MATLAB/Simulink 78
4.2 Експериментальні дослідження роботи моделі 85
4.2.1 Розпізнавання паростків Calla Palustris (Образки болотяні) 86
4.2.2 Розпізнавання паростків картоплі 92
4.2.3 Розпізнавання паростків буряку 94
4.3 Висновки до розділу 96
5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ 97
5.1 Опис ідеї проекту (товару, послуги, технології) 97
5.2 Технологічний аудит ідеї проекту 98
5.3 Аналіз ринкових можливостей запуску стартап-проекту 99
5.4 Розроблення ринкової стратегії проекту 104
5.5 Розроблення маркетингової програми стартап-проекту 106
5.6 Висновки 109
ВИСНОВКИ 110
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 111

ВСТУП

Сільськогосподарське виробництво – важлива складова економіки кожної країни.

На сьогоднішній день в Україні існує не так багато технічних рішень, для забезпечення поставлених задач. Фермери в основному працюють самостійно або наймають робочих для догляду за рослинами. Але це не ефективно, так як це займає багато часу для аналізу та догляду за угіддями. До того ж догляд включає в себе роботу з деякими хімікатами, що негативно впливає на самих фермерів.

Саме тому ця тема є актуальною на сьогоднішній день. Роботизований комплекс точкового оброблення гербіцидами сільськогосподарських угідь буде корисний для підвищення ефективності сільськогосподарського виробництва: врожайності, зменшення кількості людської робочої сили, тощо.

Метою магістерської дисертації є підвищення ефективності сільськогосподарського виробництва шляхом розроблення роботизованого комплексу точкового оброблення гербіцидами сільськогосподарських угідь.

Для досягнення поставленої мети були вирішені наступні задачі:

Огляд існуючих рішень в області робототехніки у сільськогосподарському виробництві.

- Огляд методів розпізнавання об’єктів.
- Розроблення структурної схеми роботизованого комплексу
- Розроблення функціональної схеми роботизованого комплексу
- Розроблення алгоритму роботи підсистеми робота
- Моделювання підсистеми розпізнавання рослин

Об’єкт досліджень: оброблення сільськогосподарських угідь гербіцидами

Предмет досліджень: роботизація процесу оброблення сільськогосподарських угідь гербіцидами.

Матеріали магістерської дисертації були оприлюднені на VII міжнародній науково-практичній конференції з інформаційних систем та технологій Winter InfoCom Advanced Solutions 2018.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ

Якщо подумати про глобальне зростання населення і необхідності його годувати, то фермери повинні стати більш ефективними у прибиранні та виробництві всіх сільськогосподарських культур. У майбутньому роботи будуть використовуватися для виконання більшості завдань – від посіву і підгодівлі до застосування хімікатів.

Сезонні працівники на фермах – пережиток минулого, тепер роботи будуть виконувати їх функції – збирати урожай, боротися з комахами і бур'янами.

При розпилюванні пестицидів польові працівники піддаються впливу шкідливих хімічних речовин. Система автономного розпилення може знизити використання пестицидів до 80 відсотків, оскільки вона працює дуже вибірково.

На фермі також використовуються дрони для завчасної ідентифікації грибкових захворювань, що дозволяє здійснювати більш раннє і успішне лікування. Оснащений камерою дрон з функцією GPS буде отримувати зображення сільськогосподарських культур з високою роздільною здатністю, надаючи фермерам вид з висоти пташиного польоту, що дозволяє розглянути, де культури здорові, а де потребують догляду.

Сільськогосподарські дрони також можуть здійснювати цільове розпорошення, особливо у випадку зі спеціальними культурами, обприскувати які за допомогою пілотованих літальних апаратів або занадто складно, або занадто небезпечно. Дослідники з Університету Каліфорнії в Девісі експериментують з сільськогосподарськими дронами для обприскування винограду в долині Напа.

Сучасні ферми вже використовують трактори з автоматичним рульовим керуванням, а молочні ферми встановлюють машини, які можуть доїти корів. Однак визначення окремих фруктів або овочів є набагато більш складним завданням.

Роботизовані доїльні машини беруть на себе певний обсяг праці фермерів (наприклад, годування і доїння корів без участі людини), зберігаючи час і трудові ресурси. Корови самі визначають, коли їх можна доїти, а кожна з них отримує індивідуальне обслуговування завдяки хомута з передавачами, які показують

кількість молока. Передавач може відстежувати кількість споживаної коровою трави і навіть то, скільки кроків вона зробила.

Немає двох однакових продуктів – кожен має унікальну форму, розмір і колір. Освітлення, мінливі протягом дня і ночі, сприяє тому, що кожен фрукт або овоч виглядають в різних умовах по-різному. Багато зелені овочі виглядають так само, як листові кущі або лози, на яких вони ростуть.

Прикладом цієї проблеми є прибирання зеленої квасолі, оскільки її необхідно збирати молодий, до того, так насіння всередині стручка сформують горбки. Чим частіше ви її збираєте, тим більше вона буде плодоносити, тому квасоля необхідно зривати кожні 2-3 дні. Якщо ви залишите її дозрівати, лоза перестане плодоносити і усохне

Для того щоб зрозуміти організацію в рамках віртуального безладу сільськогосподарської середовища, дослідники працюють над інтелектуальними системами зондування. Мультиспектральні камери, які аналізують довжину хвиль світла, що відбивається від об'єктів, можуть бути використані для знаходження закономірності, яка дозволить роботу зрозуміти, що він бачить, наприклад, перець, незалежно від того, як овоч росте.

Робот потім зможе вчитися на своїх помилках і вдосконалюватися під час роботи. Алгоритм буде бачити прості форми, і, якщо овоч частково покритий листям, не стане використовувати алгоритм повної форми.

Після того як робот ідентифікує урожай, він повинен буде зібрати його. Таким чином, з'являється необхідність в схоплюють інструменті, який зможе захоплювати продукцію в потрібному місці і зривати її з застосуванням правильної сили і твердості. Дослідники вивчають рух руки людини і за допомогою іншого набору алгоритмів намагаються повторити його.

Робот для догляду за салатом здатний прополоти грядки від бур'янів навколо основи рослини. Він може також розрідити грядки, в той час як для виконання такої процедури вручну знадобитися близько 20 робітників.

Робот для догляду за виноградом котиться через виноградники, обрізаючи лозу, в той час як інші роботи, які зараз знаходяться на стадії розробки, будуть віддалено перевіряти культури на показники росту, вологи і ознак захворювань.

У Франції з'явився новий працівник виноградника з чотирма колесами, двома руками і шістьма камерами, який обрізає 600 виноградних лоз в день і ніколи не йде на лікарняний. Wall-Ye V.I.N., який є дітищем бургундського винахідника Крістофа Міллота, є одним з роботів, розроблених для виконання робіт на виноградниках.

Він виконує такі завдання, як обрізка і пасинкування (видалення непродуктивних молодих пагонів), а також накопичує важливі дані про стан і вітальності грунту, плодів і лози.

Vision Robotics, компанія з Сан-Дієго, працює над парою роботів, які будуть переміщатися по фруктовим садам і збирати апельсини, яблука або інші фрукти з дерев. Через кілька років ці машини зможуть виконувати трудомістку рутинну роботу збирання плодів, для виконання якої в даний час наймаються тисячі трудових мігрантів кожен сезон.

Два роботи стануть працювати як одна команда. Перший буде сканувати дерево і створювати 3D-карту розташування та розміру кожного апельсина, обчислюючи найкращий порядок, в якому можна зірвати фрукт. Другий – це щось на зразок металевого восьминога, здатного м'яко стосуватися плодів. Перший робот зможе сканувати і відправляти інформацію другого, комбайну, який буде зривати фрукти, а запланована послідовність рухів не дасть восьми довгим рукам наштовхнутися один на одного.

Для того щоб годувати мільярди людей в усьому світі, фермери повинні використовувати роботів. Зростання чисельності населення тільки в Америці просто приголомшує, оскільки воно виросло на 22,5% в період між 1990-м (250 млн. Чоловік) і 2010 роком (310 млн.), А Бюро перепису населення очікує, що до 2050-го цифри збільшяться до більш ніж 420 мільйонів.

Якщо подумати про дозвіл проблеми глобального зростання населення і необхідності годувати цю зростаючу популяцію, то фермери повинні стати більш ефективними у прибиранні та виробництві всіх сільськогосподарських культур. У

майбутньому роботи будуть використовуватися для виконання більшості завдань – від посіву до підгодівлі, а також застосування хімікатів. Обробляти все вручну більше не буде потрібно.

1.1 Система аналізу ґрунту Adigo Field Flux Robot

Азотні добрива виділяють N2O, який негативно впливає на екологію і може пошкодити рослини: пожовтіння листя, руйнування мембрани або уповільнення зростання. В першу чергу, щоб запобігти негативному впливу закису азоту на рослини потрібно визначити кількість N2O на поле. В середньому такий тест займає 27 годин, але компанія Adigo розробила робота, який може це зробити за годину. Зовні апарат нагадує коромисло, він опускає алюмінієві блоки на землю і проводить аналіз ґрунту.

Рисунок 1.1 – Adigo Field Flux Robot

1.2 Система спостереження Ladybird

Робот Ladybird або "Сонечко" був спроектований і побудований спеціально для овочевої промисловості. Його використовують для спостереження за фермою і складання технологічних карт. На ньому встановлено цілий ряд датчиків і сонячних панелей, які дозволяють роботові стежити за ростом рослин і появою шкідників цілодобово. Тести показали, що робот може працювати три дні без підзарядки. У "Божої корівки" також є механічна рука, яка дозволяє видаляти з поля бур'яни.

Рисунок 1.2 – Ladybird

1.3 Аналізатор Rosphere

Дослідники з Мадридського університету створили сферичного робота для збору інформації про стан грунту і посівів. Принцип пересування робота нагадує зорб

або прогулянкову кулю – всередині Rosphere знаходиться маятниковий механізм, здатний рухатися в двох незалежних напрямках по команді електронної системи управління. Конструкція дозволяє роботу не тільки котитися по прямій, але і здійснювати повороти. Робот-колобок оснащений GPS-трекером і цілим рядом датчиків, завдяки яким він збирає інформацію про здоров'я посівів, склад грунту, її температури і вологості. Потім він передає цю інформацію на комп'ютер фермера за допомогою Wi-fi.

Рисунок 1.3 – Rosphere

1.4 Vitirover

Vitirover – це робот, який призначений для зрізання трави і бур'янів між виноградними лозами. Завдяки датчикам і GPS-трекера маленький французький робот вміє розрізняти виноград від інших рослин і може рухатися по полю без сторонньої допомоги. Попередньо налаштувати робота можна за допомогою мобільного додатку. В цілому, один робот може обробити 1 га за 150 годин роботи, працюючи навіть вночі завдяки встановленій сонячної панелі.

Рисунок 1.4 – Vitirover

1.5 Огляд маніпуляторів

Маніпулятор – механізм для управління просторовим положенням знарядь, об'єктів праці і конструкційних вузлів і елементів. Це значення закріпилося за словом з середини XX століття, завдяки застосуванню складних механізмів для маніпулювання небезпечними об'єктами в атомній промисловості. Використовується для переміщення різних вантажів, набув широкого поширення в сучасному суспільстві.

Основу маніпуляторів становлять просторові механізми з багатьма ступенями свободи. Маніпулятори виконують роботи в середовищах, недоступних або небезпечних для людини (підводні глибини, вакуум, радіоактивна середовище та інші агресивні середовища), допоміжні роботи в промисловому виробництві. Маніпулятори використовуються в медичній техніці (наприклад, в протезуванні).

Маніпулятори вивчає теорія маніпуляторів, яка є розділом теорії машин і механізмів. У вузькому сенсі маніпулятором називається механічна рука.

Маніпулятори діляться на керовані людиною і автоматичні маніпулятори (роботи-маніпулятори як різновид роботів). Розвиток маніпуляторів призвело до створення промислових роботів. Проектування механізмів-маніпуляторів вимагає рішення таких задач, як створення маневреності, стійкості в роботі, вибір правильного співвідношення корисних і холостих ходів. Іноді потрібно проектування таких систем, в яких оператор відчуває зусилля, що створюється на робочому органі.

У даному роботі роль маніпуляторів будуть виконувати два пристрої, що рухаються у 3-вимірному просторі автоматично, що розбризкують гербіциди на потрібні рослини.

1.6 Технічні вимоги

Розроблюваний роботизований комплекс має такі технічні характеристики:

- Полікристалічні сонячні панелі 380 W
- Бак на 30л
- Відеокамера з високим розширенням, не менш ніж 720p

Для АРМ оператора:

- Процесор: тактова частота 1.8 GHz або вище
- Оперативна пам’ять: 2 Гб або більше
- Місце на жорсткому диску: 1 Гб або більше

2 МЕТОДИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

Розпізнавання об'єктів – це техніка комп'ютерного зору для ідентифікації об'єктів у зображеннях або відео. Розпізнавання об'єктів є ключовим результатом глибокого навчання та алгоритмів машинного навчання. Коли ми дивимся на фотографію або відео, ми можемо легко помітити людей, об'єкти, сцени та візуальні подробиці. Мета полягає в тому, щоб навчити комп'ютер робити те, що є природним для людей: отримати розуміння того, що містить зображення.

Розпізнавання об'єктів є ключовою технологією автомобілів без водія, що дозволяє їм розпізнати знак зупинки або відрізнити пішохода від ліхтаря. Це також корисно в різних додатках, таких як виявлення хвороб у біовізуалізації, промислової інспекції та роботизованого зору.

Виявлення об'єктів і розпізнавання об'єктів є схожими методами для ідентифікації об'єктів, але вони розрізняються за їх виконанням. Виявлення об'єктів – це процес знаходження екземплярів об'єктів у зображеннях. У разі глибокого навчання виявлення об'єктів є підмножиною розпізнавання об'єкта, де об'єкт не тільки ідентифікується, а й розташований у зображенні. Це дозволяє ідентифікувати декілька об'єктів і розташовуватися в межах одного зображення.

Можна використовувати різні підходи для розпізнавання об'єктів. Останнім часом методики машинного навчання та глибокого навчання стали популярними підходами до проблем розпізнавання об'єктів. Обидва методи вчаться ідентифікувати об'єкти в зображеннях, але вони відрізняються їх виконанням.

Методи глибокого навчання стали популярними методами для розпізнавання об'єктів. Глибокі моделі навчання, такі як згорткові нейронні мережі, використовуються для автоматичного вивчення невід'ємних функцій об'єкта для того, щоб ідентифікувати цей об'єкт. Наприклад, мережа може навчитися виявляти відмінності між кішками і собаками, аналізуючи тисячі навчальних образів і вивчаючи особливості, які роблять кішок і собак різними.

Існує два підходи до виконання розпізнавання об'єктів за допомогою глибокого навчання:

- Підготовка моделі з нуля: щоб навчити глибоку мережу з нуля, ви збираєте дуже великий позначений набір даних і розробляєте дизайн архітектури мережі, яка вивчає можливості і будуєте модель. Результати можуть бути вражаючими, але такий підхід вимагає великої кількості навчальних даних, і вам потрібно налаштувати шари і ваги у мережі.
- Використання попередньої підготовленої моделі навчання: Більшість програм для навчання використовують підхід до трансферного навчання, процес, який передбачає точне налаштування попередньо підготовленої моделі. Ви починаєте з існуючої мережі, наприклад, AlexNet або GoogleNet, і вводите нові дані, що містять раніше невідомі класи. Цей метод є менш трудомістким і може забезпечити швидкий результат, тому що модель вже навчена на тисячах або мільйонах зображень.

Глибоке навчання пропонує високий рівень точності, але вимагає великої кількості даних, щоб зробити точні прогнози.

Методи машинного навчання також популярні для розпізнавання об'єкту і пропонують різні підходи, ніж глибоке навчання. Поширеними прикладами методів машинного навчання є:

- Видобуток функцій HOG з моделлю машинного навчання SVM
- Сумка-з-слів моделей з функціями, такими як Surf і MSER
- Алгоритм Віола-Джонса, який може бути використаний для розпізнавання різноманітних об'єктів, включаючи обличчя та верхніх тіл .

Щоб виконати розпізнавання об'єктів за допомогою стандартного підходу до машинного навчання, потрібно мати колекцію зображень (або відео) і вибрати відповідні функції кожного зображення. Наприклад, алгоритм витягнення функції може витягувати функції краю або кута, які можна використовувати для розрізнення класів даних. Ці функції додаються до моделі машинного навчання, яка відокремлюватиме ці функції в їх різних категоріях, а потім використовує цю інформацію під час аналізу та класифікації нових об'єктів Можна використовувати різні алгоритми машинного навчання та методи видобування, які пропонують безліч комбінацій для створення точної моделі розпізнавання об'єктів. Використання машинного навчання для розпізнавання об'єктів пропонує гнучкість для вибору

найкращого поєднання функцій та класифікаторів для навчання. Він може досягти точних результатів з мінімальними даними.

Визначення найкращого підходу до розпізнавання об'єктів залежить від програми та проблеми, яку потрібно вирішити. У багатьох випадках Машинне навчання може бути ефективною технікою, особливо якщо ви знаєте, які риси або характеристики зображення краще використовувати для диференціації класів об'єктів.

При виборі між машинним навчанням і глибоким навчанням потрібно приділити увагу, чи є потужний GPU і багато позначених зображень для навчання. Якщо відповідь на будь-яке з цих питань немає, підхід машинного навчання може бути кращим вибором. Методи глибокого навчання, як правило, працюють краще з більшою кількістю зображень, і GPU допомагає скоротити час, необхідний для навчання моделі.

Інші більш основні підходи до розпізнавання об'єктів можуть бути достатніми залежно від програми.

- Відповідність шаблону – який використовує маленьке зображення або шаблон, щоб знайти відповідні регіони у більшому зображенні
- Сегментація зображення і аналіз BLOB – використовує прості властивості об'єкта, такі як розмір, колір або фігура

Як правило, якщо об'єкт може бути розпізнаний за допомогою простого підходу, як сегментація зображення, краще почати з використанням більш простого підходу. Це може забезпечити надійне рішення, яке не вимагає сотень або тисяч навчальних зображень або надмірно складного рішення.

2.1 Машинне навчання

Алгоритм машинного навчання – це алгоритм, який здатний навчатися з даних. У [1] дається визначення «Комп’ютерна програма вважається такою, що може навчатися на знаннях E щодо певного класу завдань T і показника ефективності P, якщо його ефективність при завданнях у T, виміряна P, вдосконалюється із знаннями

E». Можна уявити дуже широкий спектр знань E, завдань T та показників ефективності P. Далі не буде наводитись формальні визначення того, що може бути використане для кожної з цих сутностей. Натомість, буде подано інтуїтивно зрозумілі описи та екземпляри різних видів завдань, величин ефективності та знань, які можна використовувати для будови алгоритмів машинного навчання.

2.1.1 Завдання типу T

Машинне навчання дозволяє нам вирішувати завдання, які важко вирішити за допомогою статичних програм, написаних та розроблених людьми. З наукової та з філософської точки зору, машинне навчання цікаве тим, що розвиваючи наше розуміння машинного навчання тягне за собою розвиток нашого розуміння принципів, що закладені в основі інтелекту.

У цьому відносно формальному визначенні слова «завдання» процес навчання сам по собі не є завданням. Навчання – це засіб досягнення здатності виконувати завдання. Наприклад, якщо ми хочемо, щоб робот міг ходити, то завдання – це ходьба. Ми могли б запрограмувати робота, щоб він навчився ходити, або ми могли б спробувати безпосередньо написати програму, яка визначає, як ходити самостійно.

Завдання машинного навчання зазвичай описуються з точки зору того, як система машинного навчання повинна обробляти екземпляр. Екземпляр – це сукупність ознак, які були кількісно виміряні від якихось об'єктів чи подій, які система машинного навчання має обробити. Зазвичай екземпляр представляється як вектор $\boldsymbol{x} \in \mathbb{R}^n$, де кожен запис x_i вектора – інша ознака. Наприклад, ознаками зображення зазвичай є значення пікселів на зображенні.

Багато завдань можна вирішити за допомогою машинного навчання. До найбільш поширених загальних завдань машинного навчання належать:

- Класифікація: У цьому типі завдань комп'ютерну програму має вказати до якої з категорій k належать вхідні дані (далі «вхід»). Для вирішення цього завдання, навчання алгоритм зазвичай просять створити функцію $f : \mathbb{R}^n \to \{1,...,k\}$. Коли

$y = f(\boldsymbol{x})$, модель присвоює вхід, описаний вектором $\boldsymbol{x}$, категорії ідентифікований числовим кодом y. Є й інші варіанти класифікації завдання, наприклад, де f виводить розподіл ймовірності по класах. Прикладом завдання класифікації є розпізнавання об'єктів, де входом є зображення (зазвичай описується як набір значень яскравості пікселів), і вихід – числовий код, що ідентифікує об'єкт на зображенні. Сучасне розпізнавання об'єктів найкраще здійснюється за допомогою глибокого навчання [2]. Розпізнавання об'єктів – це та сама основна технологія, яка дозволяє комп'ютерам розпізнавати обличчя [3], які можна використовувати для автоматичного ідентифікування людей у наборі фотографій та дозволяють комп'ютерам взаємодіяти більш природньо з користувачами.

- Класифікація з відсутніми вхідними даними: Класифікація стає складнішою, якщо не гарантується, що комп'ютерній програмі завжди буде надано кожне вимірювання у вхідному векторі. Щоб вирішити класифікаційну задачу, алгоритм навчання повинен визначати лише одне відображення функції з вектора входу до категоричного виходу. Коли деякі з входів можуть бути відсутніми, а не надання єдиної функції класифікації, алгоритм навчання повинен засвоїти набір функцій, аніж надавати єдину функцію класифікації. Кожна функція відповідає класифікованому $\boldsymbol{x}$ з іншими підмножинами його відсутніх входів. Така ситуація виникає часто в медичній діагностиці, тому що багато видів медичних тестів є дорогими або інвазивними. Одним із способів ефективного визначення такого великого набору функцій є навчання розподілу ймовірності за всіма відповідними змінними, а потім вирішити завдання класифікації шляхом маргіналізації відсутніх змінних. З n вхідних змінних, тепер можна отримати всі 2^n різних необхідних функцій класифікації для кожного можливого набору відсутніх входів, але потрібно лише вивчити одну функцію, що описує спільний розподіл ймовірностей. Багато інших завдань, описаних у цьому розділі, також можуть бути узагальнені для роботи з відсутніми входами; класифікація з відсутніми входами є лише один приклад того, що може зробити машинне навчання.

- Регресія: У цьому типі завдань комп’ютерна програма має передбачити числове значення, задане деяким входом. Для вирішення цього завдання алгоритм навчання має вивести функцію $f: \mathbb{R}^n \to \mathbb{R}$. Цей тип завдань схожий на класифікацію, за винятком того, що формат виходу відрізняється. Приклад завдання регресії – це передбачення очікуваної суми відшкодування, що вимагатиме застрахована особа (використовується для встановлення страхових внесків) або прогнозування майбутніх цін на цінні папери. Такі види прогнозів також використовуються для алгоритмічної торгівлі.

- Транскрипція: У цьому типі завдань система машинного навчання має спостерігати відносно неструктуроване подання якихось даних і переписати його в дискретний, текстовий вигляд. Наприклад, в розпізнаванні оптичних символів, в комп'ютерній програмі відображається фотографія, що містить зображення тексту і вимагається повернути цей текст у вигляді послідовності символів (наприклад, у форматі ASCII або Unicode). Google Street View використовує глибоке навчання таким чином обробляти адресні номери [4]. Інший приклад – розпізнавання мовлення, де комп'ютерній програмі надається звукова форма хвилі, після чого видає послідовність символів або ідентифікаційні коди слів, що описують слова, вимовлені в аудіозаписі. Глибоке навчання є найважливішим компонентом сучасних систем розпізнавання мовлення у великих компаніях, включаючи Microsoft, IBM та Google [5].

- Машинний переклад: у завданні машинного перекладу вхід вже складається з послідовності символів на якійсь мові, і комп'ютерна програма повинна перетворити це в послідовність символів іншою мовою. Це зазвичай застосовується до природних мов, таких як переклад з англійської на французьку. Нещодавно глибоке навчання почало мати важливий вплив на цей вид завдання [6].

- Структурований вихід: Завдання структурованого виходу включають будь-які завдання, де вихід – вектор (або інша структура даних, що містить кілька значень) з важливими відносинами між різними елементами. Це велика категорія, і включає описані вище завдання транскрипції та перекладу, але також багато інших завдань. Один із прикладів – синтаксичний аналіз – відображення природної мови

речення в дерево, яке описує його граматичну структуру та позначення вузлів дерев як дієслова, іменники чи прислівники тощо [7]. Ще один приклад – це піксельна сегментація зображень, коли комп'ютерна програма призначає кожен піксель на зображенні до певної категорії. Наприклад, глибоке навчання може використовуватися для анотування місць доріг на аерофотознімках [8]. Вихід повинен мати структуру дзеркально відображену як на вході настільки ж схожим, як у завданнях у стилі анотації. Наприклад, у підписуванні зображень, комп’ютерна програма спостерігає за зображенням та виводить природною мовою речення, що описує зображення [9, 10, 11]. Ці завдання називаються завданнями структурованого виходу, тому що програма повинна вивести кілька значень, які всі щільно взаємопов'язані. Наприклад, слова, створені програмою підпису зображень повинна сформувати дійсне речення.

- Виявлення аномалій: У цьому типі завдань комп'ютерна програма просіює набір подій або об’єктів і позначає деякі з них як незвичні або нетипові. Прикладом завдання виявлення аномалії є виявлення шахрайства по кредитній картці. Моделюючи звички покупок, компанія, що випустила кредитну картку, може виявити транзакції по картці, що не співпадають з моделлю. Якщо злодій краде інформацію про вашу кредитну картку або саму кредитну картку, покупки злодія часто йдуть від іншого розподілу ймовірностей над типами покупок, ніж ваш власний. Компанія з кредитних карт може запобігти шахрайство шляхом утримання рахунку, як тільки ця карта була використана на нехарактерну покупку [12].

- Синтез та вибірки: У цьому типі завдань алгоритм машинного навчання має створити нові екземпляри, подібні до тих даних, на яких алгоритм навчається. Синтез та вибірки через машинне навчання можуть бути корисними для медіа-додатків, де створення великих об’ємів контенту вручну може бути дорогим або нудним. Наприклад, відеоігри можуть автоматично створювати текстури для великих об’єктів чи пейзажів, а не вимагати від дизайнера вручну позначити кожен піксель [13]. У деяких випадках потрібно, щоб процедура вибірки чи синтезу генерувала певний вид виходу на основі заданого входу. Наприклад, у задачі синтезу мовлення ми надаємо письмове речення та вимагаємо від програми видати звукову форму, що

містить розмовна версія цього речення. Це вид завдання структурованого виходу, але з доданим застереженням, що немає єдиного правильного виходу на кожен вхід, і ми явно бажаємо великої кількості варіацій у виході, для того, щоб результат здався більш природним та реалістичним.

- Імпутація пропущених значень: У цьому типі завдань алгоритму машинного навчання надається новий екземпляр $\boldsymbol{x} \in \mathbb{R}^n$, але з деякими записами x_i з пропущених $\boldsymbol{x}$. Алгоритм повинен забезпечувати прогнозування значень пропущених записів.

- Позначення: У цьому типі завдань алгоритму машинного навчання надається пошкоджений екземпляр $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$, отриманий невідомим пошкоджуючим процесом з правильного екземпляру $\boldsymbol{x} \in \mathbb{R}^n$. Алгоритм повинен передбачити правильний екземпляр $\boldsymbol{x}$ зі своєї пошкодженої версії $\tilde{\boldsymbol{x}}$, або загалом передбачити умовний розподіл ймовірності $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$.

- Оцінка щільності або оцінка функції масової ймовірності: У проблемі оцінки щільності алгоритм машинного навчання має вивчити функцію $p_{\text{model}} : \mathbb{R}^n \to \mathbb{R}$, де $p_{\text{model}}(\boldsymbol{x})$ можна інтерпретувати як функцію щільності ймовірністі (якщо $\boldsymbol{x}$ є безперервною) або функція масової ймовірності (якщо $\boldsymbol{x}$ є дискретною) на простір, з якого були складені екземпляри. Щоб зробити таке завдання добре, алгоритм повинен вивчити структуру даних, що він аналізував. Він повинен знати, де екземпляри щільно кластеризуються і де вони навряд чи трапляться. Більшість завдань, описаних вище, вимагають, щоб алгоритм навчання принаймні неявно охопив структуру розподілу ймовірностей. Оцінка щільності дозволяє нам чітко фіксувати цей розподіл. В принципі, ми можемо виконувати обчислення на цьому розподілі, щоб вирішити й інші завдання. Наприклад, якщо ми виконали оцінку щільності для отримання розподілу ймовірності $p(\boldsymbol{x})$, ми можемо використовувати цей розподіл для вирішення задачі щодо внесення відсутнього значення. Якщо значення x_i відсутнє, а всі інші значення, позначені $\boldsymbol{x}_{-i}$, надаються, тоді ми знаємо, що розподіл над ним задається $p(x_i \mid \boldsymbol{x}_{-i})$. На практиці, Оцінка щільності не завжди

дозволяє нам вирішити всі ці пов'язані завдання, тому що в багатьох випадках необхідні операції на $p(\boldsymbol{x})$ є обчислювально-непереборними.

2.1.2 Міра ефективності типу P

Для того, щоб оцінити здібності алгоритму машинного навчання, потрібно розробити кількісний показник його ефективності. Зазвичай цей показник ефективності P є характерним для завдання T, яке виконує система.

Для таких завдань, як класифікація, класифікація з відсутніми входами та транскрипція, часто вимірюється точність моделі. Точність – це лише пропорція екземплярів, для яких модель дає правильний вихід. Також можна отримати еквівалентну інформацію, вимірюючи коефіцієнт помилок, частку екземплярів, для яких модель дає неправильний вихід. Часто мається на увазі коефіцієнт помилок, коли кажуть очікувана втрата 0-1. Втрата 0-1 на конкретному екземплярі дорівнює 0, якщо це правильно класифікований та 1, якщо ні. Для таких завдань, як оцінка щільності, не має сенсу вимірювати точність, коефіцієнт помилок або будь-який інший тип втрат 0-1. Натомість ми повинні використовувати інший показник ефективності, який надає моделі безперервне значення оцінки за кожен екземпляр. Найпоширеніший підхід – звітувати про середню log-ймовірність, що модель призначає для деяких екземплярів.

Зазвичай нас цікавить, наскільки добре працює алгоритм машинного навчання на даних, яких він раніше не бачив, оскільки це визначає, наскільки добре він буде працювати, коли він буде розгорнутий в реальному світі. Тому оцінюються ці показники ефективності з використанням даних, які є окремими від даних, що використовуються для навчання системи машинного навчання.

Вибір показників ефективності може здатися простим та об'єктивним, але часто важко вибрати міру ефективності, яка добре відповідає бажаної поведінки системи.

У деяких випадках це відбувається тому, що важко вирішити, що слід вимірювати. Наприклад, виконуючи завдання з транскрипції, чи слід вимірювати точність системи при транскрибуванні цілих послідовностей або потрібно

використовувати більш дрібну зернистість показників ефективності, який дає часткову довіру на отримання деяких елементів послідовність правильно? Виконуючи регресійне завдання, чи слід штрафувати систему більше, якщо вона часто робить середні помилки або якщо вона рідко робить дуже великі помилки? Такі види вибору залежать від застосування додатку.

В інших випадках ми знаємо, яку кількість ми в ідеалі хотіли б виміряти, але вимірювати це недоцільно. Наприклад, це часто виникає в контексті оцінки щільності. Багато найкращих імовірнісних моделей представляють розподіли ймовірностей лише неявно. Обчислення фактичного значення ймовірності, призначеного конкретній точці простору у багатьох подібних моделях непереборні. У цих випадках потрібно розробити альтернативний критерій, який все ще відповідає цілям проектування, або спроектувати добре наближення до потрібного критерію.

2.1.3 Знання типу E

Алгоритми машинного навчання можуть бути широко віднесені до категорії некерованих або керованих за знаннями, якими вони можуть мати під час навчального процесу.

Один з найдавніших наборів даних, який вивчали статистики та дослідники машинного навчання – це набір даних про «Ірис» [14]. Це сукупність вимірювань різних частини 150 рослин ірису. Кожна окрема рослина відповідає одному екземпляру. Особливістю кожного екземпляру є вимірювання кожної з частин рослини: довжина чашолистка, ширина чашолистка, довжина пелюсток і ширина пелюсток. Набір даних також фіксується, до якого виду належала кожна рослина. Три різні види представлені у наборі даних.

Некеровані алгоритми навчання вивчають набір даних, що містить безліч функцій, потім вивчають корисні властивості структури цього набору даних. У контексті глибокого навчання, зазвичай хочуть дізнатись весь розподіл ймовірностей, що генерується з набору даних, аніж явно як в оцінці щільності чи неявно для таких завдань як синтез чи позначання. Деякі інші непідтримувані алгоритми навчання

виконують інші ролі, такі як кластеризація, яка складається з поділу набору даних на кластери схожих екземплярів.

Керовані алгоритми навчання вивчають набір даних, що містять функції, але кожен екземпляр також пов'язаний з міткою або ціллю. Наприклад, набір даних «Ірис» описується до видів кожної рослини ірису. Керований алгоритм навчання може вивчити набір ірисів і навчитися класифікувати рослини ірису на три різні види на основі їх вимірювань.

Грубо кажучи, некероване навчання передбачає спостерігання кількох прикладів випадкового вектора **x** і намагається неявно або явно дізнатися розподіл ймовірності $p(\mathbf{x})$, або деякі цікаві властивості цього розподілу, в той час як контрольоване навчання включає спостереження за кількома прикладами випадкових векторів **x** і асоційоване значення або вектор **y**, і вчити прогнозувати **y** від **x**, як правило, оцінюючи $p(\mathbf{y} \mid \mathbf{x})$. Термін кероване навчання походить з точки зору цільового **y**, що надається інструктором або вчителем, який показує системі машинного навчання, що робити. У некерованому навчанні немає інструктора або викладача, і алгоритм повинен навчитися розуміти дані без керівництва.

Некероване навчання та кероване навчання формально не визначені термінами. Границі між ними часто розмиті. Багато технологій машинного навчання можуть використовуються для виконання обох завдань. Наприклад, ланцюгове правило вірогідності станів, що для вектора $x \in \mathbb{R}^n$ спільний розподіл можна розкласти як

$$p(\mathbf{x}) = \prod_{i=1}^{n} p\left(x_i \mid x_1, \ldots, x_{i-1}\right) \tag{2.1}$$

Цей розпад означає, що ми можемо вирішити нібито непідвладну задачу моделювання $p(\mathbf{x})$ шляхом поділу на n задач, що контролюються. Крім того, ми можемо вирішити контрольовану проблему навчання $p(y \mid \mathbf{x})$, використовуючи традиційні непідконтрольні технології навчання для вивчення спільного розподілу $p(\mathbf{x}, y)$ і як результат

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')} \tag{2.2}$$

Хоча некероване навчання та кероване навчання не є повністю формальними або різними поняттями, вони допомагають приблизно класифікувати деякі речі, які ми робимо за допомогою алгоритмів машинного навчання. Традиційно люди звертаються до регресії, класифікації та проблем структурованих виходів як кероване навчання. Оцінка щільності в підтримку інших завдань зазвичай вважається некерованим навчанням.

Можливі й інші варіанти парадигми навчання. Наприклад, у напівкерованому навчанні, деякі екземпляри включають ціль керування, а інші ні. У навчанні з кількома екземплярами вся колекція прикладів позначена як містить або не містить екземпляр класу, але окремі члені колекції не маркуються [15].

Деякі алгоритми машинного навчання не просто мають фіксований набір даних. Наприклад, алгоритми навчання посилення взаємодіють із середовищем, тому існує цикл зворотного зв'язку між системою навчання та її досвідом.

Більшість алгоритмів машинного навчання просто мають набір даних. Набір даних може описуватися багатьма способами. У всіх випадках набір даних – це сукупність прикладів, які в свою чергу є колекціями функцій.

Один поширений спосіб опису набору даних – це дизайн-матриця. Дизайн матриця – матриця, що містить різний екземпляр у кожному рядку. Кожен стовпець матриці відповідає іншій ознаці. Наприклад, набір даних «Ірис» містить 150 екземплярів з чотирма ознаками для кожного екземпляру. Це означає, що ми можемо представляти набір даних з проектною матрицею $\boldsymbol{X} \in \mathbb{R}^{150\times 4}$, де $X_{i,1}$ – довжина чашолистку рослини і, $X_{i,2}$ – ширина чашолистку рослини і тощо.

Звичайно, щоб описати набір даних як дизайн матрицю, це має бути можливо описати кожен екземпляр як вектор, і кожен з цих векторів повинен бути однакового розміру. Це не завжди можливо. Наприклад, якщо є колекція фотографій з різною

шириною та висотою, то різні фотографії будуть містити різні кількість пікселів, тому не всі фотографії можуть бути описані однаковою довжиною вектора. У таких випадках, ніж опис набору даних як матриця з m рядками, ми опишемо її як набір, що містить m елементів: $\left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(m)}\right\}$. Це позначення не означає, що будь-які два приклади векторів $\boldsymbol{x}^{(i)}$ і $\boldsymbol{x}^{(j)}$ мають однаковий розмір.

У випадку з керованим навчанням екземпляр містить мітку або ціль як а також колекцію функцій. Наприклад, якщо ми хочемо використовувати алгоритм навчання щоб виконати розпізнавання об'єктів з фотографій, нам потрібно вказати, який об’єкт з'являється на кожній із фотографій. Ми можемо зробити це за допомогою числового коду, де 0 позначає людину, 1 означає автомобіль, 2 означає кішку тощо. Часто під час роботи з набором даних, що містить дизайн матрицю спостережень за характеристиками $\boldsymbol{X}$, також забезпечується вектор міток $\boldsymbol{y}$, причому y_i забезпечує мітку, наприклад, і.

Звичайно, іноді мітка може бути більше, ніж просто одне число. Наприклад, якщо ми хочемо навчити систему розпізнавання мовлення транскрибувати усі речення, тоді мітка для кожного екземпляру речень – це послідовність слів.

Так само як не існує офіційного визначення керованого та некерованого навчання, не існує жорсткої систематики наборів даних або знань. Описані тут структури охоплюють більшість випадків, але завжди можна створити нові для нових додатків.

2.1.4 Лінійна регресія

Визначення алгоритму машинного навчання як алгоритму, який здатний покращувати продуктивність комп’ютерної програми при виконанні деяких завдань за допомогою знань дещо абстрактне. Для конкретизації, наведемо приклад простого алгоритму машинного навчання: лінійна регресія.

Як випливає з назви, лінійна регресія вирішує проблему регресії. Інакше кажучи, мета полягає в тому, щоб побудувати систему, яка може приймати вектор $\boldsymbol{x} \in \mathbb{R}^n$ як вхідний і передбачити значення скалярного $y \in \mathbb{R}$ як його вихід. У випадку лінійної регресії, вихід – це лінійна функція входу. Нехай $\hat{y}$ – це значення, яке модель прогнозує, що у прийме. Позначимо вихід як

$$\hat{y} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} \tag{2.3}$$

де $\boldsymbol{w} \in \mathbb{R}^n$ – вектор параметрів.

Параметри – це значення, що контролюють поведінку системи. У цьому випадку w_i є коефіцієнт, який ми множимо на ознаку x_i, перед тим як підбити підсумки від усіх особливостей. Ми можемо розглядати $\boldsymbol{w}$ як набір ваг, що визначають, як кожна ознака впливає на передбачення. Якщо функція x_i отримує позитивну вагу w_i, то збільшення значення цієї ознаки збільшує значення нашого прогнозу $\hat{y}$. Якщо функція отримує негативну вагу, збільшуючи значення цієї функції зменшує значення нашого прогнозу. Якщо маса функції велика, то це має великий вплив на передбачення. Якщо вага функції дорівнює нулю, вона не має вплив на прогнозування.

Таким чином, у нас є визначення нашої задачі T: передбачити y від $\boldsymbol{x}$ шляхом виведення $\hat{y} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$. Далі потрібно вивести визначення міри ефективності, P.

Припустимо, є дизайн матриця з m вхідних прикладів, які не будуть використовуватися для навчання, лише для оцінки ефективності моделі. Також є вектор регресійних цілей, що забезпечує правильне значення y для кожного з цих екземплярів. Оскільки цей набір даних буде використовуватися лише для оцінки, ми називаємо це тестовим набором. Позначимо дизайн матрицю входів як $\boldsymbol{X}^{(\text{test})}$, а вектор регресії цілі як $\boldsymbol{y}^{(\text{test})}$.

Один із способів вимірювання продуктивності моделі – це обчислення середньо-квадратичного значення похибки моделі на тестовому наборі. Якщо $\hat{\boldsymbol{y}}^{(\text{test})}$ дає прогнози моделі на тестовому наборі, тоді середньо-квадратична помилка задається рівнянням

$$\text{MSE}_{\text{test}} = \frac{1}{m}\sum_i \left(\hat{\boldsymbol{y}}^{(test)} - \boldsymbol{y}^{(test)}\right)_i^2 \tag{2.4}$$

Очевидно, що ця міра помилок зменшується до 0, коли $\hat{\boldsymbol{y}}^{(test)} = \boldsymbol{y}^{(test)}$. Також видно, що

$$\text{MSE}_{\text{test}} = \frac{1}{m}\left\|\hat{\boldsymbol{y}}^{(test)} - \boldsymbol{y}^{(test)}\right\|_2^2 \tag{2.5}$$

тому помилка збільшується кожен раз, коли евклідова відстань між передбаченням і ціллю збільшуються.

Щоб скласти алгоритм машинного навчання, потрібно розробити такий алгоритм, що поліпшить ваги $\boldsymbol{w}$ таким чином, щоб зменшувався MSE_{test} коли алгоритму дозволяється набувати знань, спостерігаючи за навчальним $\left(\boldsymbol{X}^{(\text{train})}, \boldsymbol{y}^{(\text{train})}\right)$. Один очевидний спосіб зробити це якраз мінімізувати середню квадратичну помилку на навчальному наборі, $\text{MSE}_{\text{train}}$.

Щоб мінімізувати $\text{MSE}_{\text{train}}$, ми можемо просто вирішити, де його градієнт дорівнює 0:

$$\nabla_w \text{MSE}_{\text{train}} = 0 \tag{2.6}$$

$$\Rightarrow \nabla_w \frac{1}{m}\left\|\hat{\boldsymbol{y}}^{(\text{train})} - \boldsymbol{y}^{(\text{train})}\right\|_2^2 = 0 \tag{2.7}$$

$$\Rightarrow \frac{1}{m}\nabla_w \left\| \boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right\|_2^2 = 0 \tag{2.8}$$

$$\Rightarrow \nabla_w \left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right)^{\mathrm{T}} \left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{2.9}$$

$$\Rightarrow \nabla_w \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{w}^{\mathrm{T}} \boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{y}^{(\text{train})} + \boldsymbol{y}^{(\text{train})\mathrm{T}} \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{2.10}$$

$$\Rightarrow 2\boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{y}^{(\text{train})} = 0 \tag{2.11}$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\mathrm{T}} \boldsymbol{y}^{(\text{train})} \tag{2.12}$$

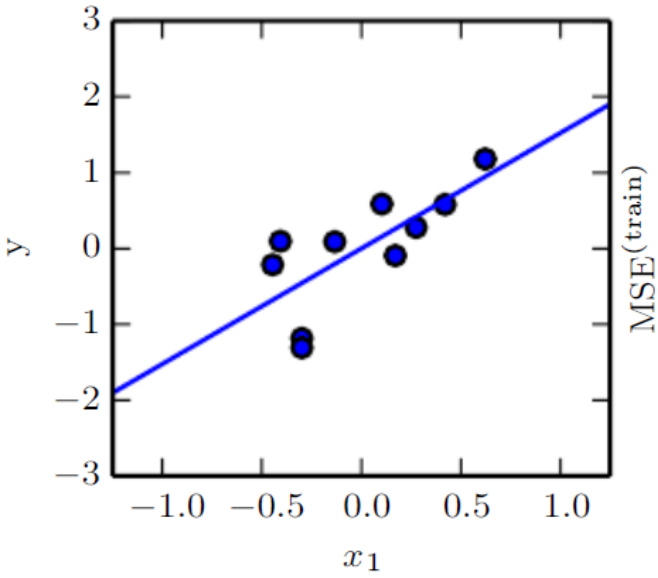


Рисунок 2.1 – Приклад лінійної регресії

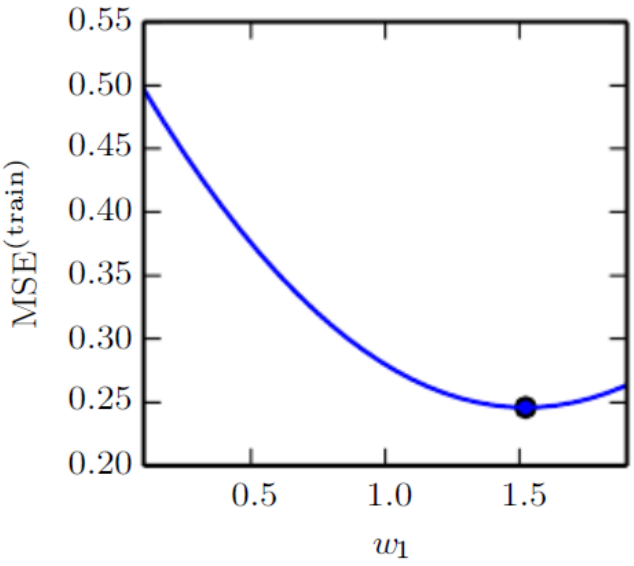


Рисунок 2.2 – Оптимізація w

Лінійна проблема регресії з навчальним набором, що складається з десяти точок даних, кожна з яких містить одну особливість. Оскільки є лише одна ознака, ваговий вектор w містить лише один параметр для навчання, w_1. Лінійна регресія вчиться встановлювати w_1 таким чином, щоб лінія $y = w_1 x$ максимально наближалась до проходження через усі навчальні точки. Нанесена точка показує значення w_1, знайдене нормальними рівняннями, які мінімізують середню квадратичну помилку на навчальному наборі.

Система рівнянь, рішення якої задано рівнянням 2.12 відома як нормальні рівняння. Оцінка рівняння 2.12 являє собою простий алгоритм навчання. Приклад алгоритму навчання лінійної регресії в дії наведено рисунках 2.1 та 2.2.

Варто зазначити, що термін лінійна регресія часто використовується для позначення трохи більш досконалої моделі з одним додатковим параметром – перехоплення виразу b. У цій моделі

$$\hat{y} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b \tag{2.13}$$

тому відображення від параметрів до прогнозів все ще є лінійною функцією, але відображення від функцій до прогнозів тепер є афінною функцією. Це розширення до афінної функції означає, що графік прогнозів моделі все ще виглядає як лінія, але вона не повинна проходити через початкову. Замість додавання параметра зміщення b, можна продовжувати використовувати модель лише з вагами, але збільшити $\boldsymbol{x}$ з додатковим записом, який завжди встановлено в 1. Вага, що відповідає 1 додатковому запису грає роль параметра зміщення.

Термін перехоплення b часто називають параметром зміщення афінного перетворення. Ця термінологія виходить з тієї точки зору, що вихідні перетворення упереджене до b у відсутності будь-якого входу. Цей термін відрізняється від ідеї статистичного зміщення, в якій статистична оцінка очікуваної оцінка кількості алгоритму не дорівнює справжній кількості.

2.1.5 Ємність, перенавчання та недонавчання

Основна проблема машинного навчання полягає в тому, що ми повинні добре працювати на нових, раніше невідомих входах – не лише ті, на яких навчалась наша модель. Здатність добре працювати на раніше не помічених входах називається узагальненням.

Як правило, під час навчання моделі машинного навчання ми маємо доступ до навчального набору, ми можемо обчислити деяку міру помилок на навчальному

наборі, яка називається помилка навчання, і ми зменшуємо цю помилку навчання. Поки що те, що було описано, просто проблема оптимізації. Що відрізняє машинне навчання від оптимізації, це те що помилка узагальнення, яка також називається помилкою тесту, була також низькою. Помилка узагальнення визначається як очікуване значення помилки на новому вході. Тут очікування береться з різних можливих входів, що з'являються з розподілу входів, з якими, як очікується, система зіткнеться на практиці.

Зазвичай ми оцінюємо похибку узагальнення моделі машинного навчання вимірюючи його ефективність на тестовому наборі прикладів, які були зібрані окремо від навчального набору.

У нашому прикладі лінійної регресії ми тренували модель, зводячи до мінімуму помилку навчання,

$$\frac{1}{m^{(train)}}\left\|\boldsymbol{X}^{(train)}\boldsymbol{w}-\boldsymbol{y}^{(train)}\right\|_2^2 \tag{2.14}$$

але насправді нас хвилює помилка тесту $\frac{1}{m^{(test)}}\left\|\boldsymbol{X}^{(test)}\boldsymbol{w}-\boldsymbol{y}^{(test)}\right\|_2^2$

Як ми можемо впливати на продуктивність тестового набору, коли ми будемо спостерігати лише за навчальним набором? Сфера статистичної теорії навчання дає деякі відповіді. Якщо навчання та тестовий набір зібрані довільно, ми можемо мало чого зробити. Якщо нам дозволяють зробити деякі припущення щодо того, як навчання та тестові набори збираються, тоді ми можемо досягти певного прогресу.

Навчальні та тестові дані, що формуються розподілом вірогідності по наборам даних називається процесом генерації даних. Зазвичай ми робимо відомий набір припущень у сукупності, також відомий як i.i.d. припущення (independent identically disturbed (незалежно ідентично розподілені)). Ми припускаємо, що приклади у кожному наборі даних є незалежними один від одного, і що навчальний набір і тестовий набір розподіляються однаково, виведені з того ж розподілу ймовірності, що

і один одного. Це припущення дозволяє описати процес генерації даних з розподілом ймовірності на одному прикладі. Такий же розподіл потім використовується для створення кожного навчального прикладу та кожного тестового прикладу. Ми називаємо цей спільний базовий розподіл – розподіл генерації даних, позначений як p_{data}. Це ймовірнісні рамки та i.i.d. припущення дозволяють нам математично вивчити взаємозв'язок між помилкою навчання та помилкою тесту.

Один безпосередній зв'язок, який можна спостерігати між помилкою навчання та тесту полягає в тому, що очікувана похибка навчання випадково вибраної моделі дорівнює очікуваній помилці тесту цієї моделі. Припустимо, у нас є розподіл ймовірностей $p(\boldsymbol{x}, y)$, і ми відбираємо з нього кілька разів, щоб створити навчальний набір і тестовий набір. Для деякого фіксованого значення $\boldsymbol{w}$ очікувана помилка навчального набору точно така ж, як і очікувана помилка тестового набору, оскільки обидва очікування формуються, використовуючи однаковий процес вибірки даних. Єдина різниця між двома умовами – це ім'я, яке ми присвоюємо набору даних, який ми відбираємо.

Звичайно, коли ми використовуємо алгоритм машинного навчання, ми не виправляємо параметри достроково, а потім відбираємо обидва набори даних. Ми проводимо вибірку навчального набору, потім користуємося нею, щоб вибрати параметри для зменшення помилки навчального набору, а потім робимо вибірку тестового набору. У цьому процесі очікувана похибка тесту більша або дорівнює очікуваному значенню помилки навчання. Фактори, що визначають, наскільки добре працює алгоритм машинного навчання буде працювати, є його здатність:

- Зробити помилку навчання малою.
- Зробити розрив між помилкою навчання та тесту малим.

Ці два фактори відповідають двом головним завданням машинного навчання: перенавчання та недонавчання. Недонавчання виникає тоді, коли модель не в змозі отримати достатньо низьке значення помилки на навчальному наборі. Перенавчання виникає, коли розрив між помилкою навчання та помилкою тесту занадто великий.

Ми можемо контролювати поведінку моделі – перенавчання або недонавчання, змінивши її ємність. Грубо кажучи, ємність моделі – це її здатність підходити до найрізноманітніших функцій. Моделям з низькою ємністю потрібно робити зусилля, щоб вона підходила навчальному набору. Моделі з високою ємністю може перевершити, запам'ятовуючи властивості тренувального набору, які не дуже підходять для тестового набору.

Один із способів контролювати ємність алгоритму навчання – це вибір його простір гіпотез, набір функцій, дозволених алгоритму навчання вибрати як рішення. Наприклад, алгоритм лінійної регресії має сукупність усіх лінійних функцій його вводу як простору гіпотез. Ми можемо узагальнити лінійну регресію, щоб включити в неї поліноми, а не просто лінійні функції у його просторі гіпотез. Це збільшує ємність моделі.

Поліном першого ступеня дає нам модель лінійної регресії з передбаченням

$$\hat{y} = b + wx \tag{2.15}$$

Вводячи x^2 як ще одну ознаку, надану моделі лінійної регресії, можна навчити модель, квадратичну як функцію x:

$$\hat{y} = b + w_1 x + w_2 x^2 \tag{2.16}$$

Хоча ця модель реалізує квадратичну функцію свого вводу, вихід є як і раніше лінійною функцією параметрів, тому ми можемо використовувати звичайні рівняння для навчання моделі у закритому вигляді. Ми можемо продовжувати додавати більше ступенів *x* як додаткові особливості, наприклад для отримання поліному ступеня 9:

$$\hat{y} = b + \sum_{i=1}^{9} w_i x^i \tag{2.17}$$

Алгоритми машинного навчання, як правило, найкращі, коли їх ємність є доцільним з огляду на справжню складність завдання, яке їм потрібно виконати та обсяг навчальних даних, які їм надаються. Моделі з недостатньою ємністю не в змозі вирішити складні завдання. Моделі з високою ємністю можуть вирішити складні завдання, але коли їх ємність вища, ніж потрібно для вирішення поточного завдання, вони можуть перенавчатись.

Рисунок 2.3 показує цей принцип дії. Ми порівнюємо лінійний, квадратичний та передбачувач ступеня 9, що намагаються підійти під задачу, де справжня основна функція є квадратичною. Лінійна функція не в змозі зафіксувати кривизну в справжній проблемі, тож це недонавченність. Пердбачувач ступеня 9 здатний представляти правильну функцію, але він також здатний представляти нескінченно багато інших функцій, які проходять саме через навчальні точки, тому що у нас більше параметрів, ніж навчальних прикладів. У нас мало шансів вибрати рішення що добре узагальнює, коли існує багато різних рішень. У цьому прикладі квадратична модель ідеально відповідає справжній структурі завдання, так що добре узагальнює нові дані.

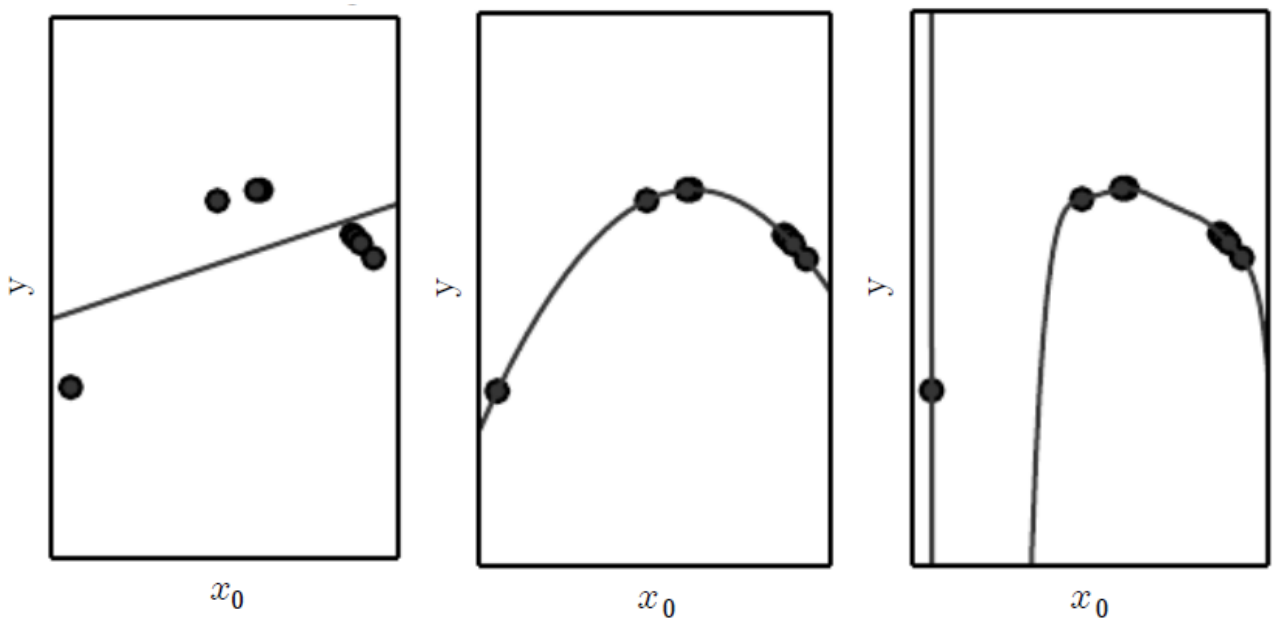


Рисунок 2.3 – Недонавчання, доцільна ємність та перенавчання

Поки було лише описано зміну ємності моделі шляхом зміни кількості вхідних функцій, які він має (і одночасно додаючи нові параметри пов'язані з тими ознаками). Насправді існує багато способів зміни ємності моделі. Ємність не визначається лише вибором моделі. Модель визначає, яке сімейство функцій алгоритм навчання може вибрати, під час зміни параметрів з метою зменшення навчальної мети. Це називається репрезентативна ємність моделі. У багатьох випадках пошук найкращої функції всередині цієї сім'ї дуже складна проблема оптимізації. На практиці навчання алгоритм насправді не знаходить найкращої функції, а лише ту, що суттєво зменшує помилку навчання. Ці додаткові обмеження, такі як недосконаліть алгоритму оптимізації, означають, що ефективна ємність алгоритму навчання може бути меншою, ніж репрезентативний потенціал сімейства моделей.

Наші сучасні ідеї щодо вдосконалення узагальнення машинного навчання моделі – це уточнення думки, що датується філософам як мінімум за часів Птоломея. Багато ранніх науковців посилаються на принцип скупості, який зараз найбільш широко відомий як бритва Оккама. Цей принцип стверджує, що серед конкуруючих гіпотез, які однаково добре пояснюють відомі спостереження, слід вибрати найпростіший. Ця ідея була формалізована та зроблена більш точно у 20 столітті засновниками статистичної теорії навчання [16].

Статистична теорія навчання пропонує різні засоби кількісної оцінки ємності моделі. Серед них найбільш відомий – *вимір Вапника-Червоненкіса*, або ВК вимір. ВК вимір вимірює ємність бінарного класифікатора. ВК вимір визначається як найбільше можливе значення m, для якого існує навчальний набір з m різних $\boldsymbol{x}$ точок, який класифікатор може мітити довільно.

Кількісне визначення ємності моделі дозволяє статистичній теорії навчання робити кількісні прогнози. Найважливіші результати у теорії статистичного навчання показує, що невідповідність між помилкою навчання та помилкою узагальнення зверху обмежується величиною, яка зростає в міру зростання ємності моделі, але скорочується, коли кількість навчальних прикладів збільшується. Ці межі забезпечують інтелектуальне обґрунтування того, що алгоритми машинного

навчання можуть працювати, але вони рідко використовуються на практиці при роботі з алгоритмами глибокого навчання. Це частково тому, що межі часто досить розмиті, а частково тому, що може бути важко визначити ємність алгоритмів глибокого навчання. Проблема визначення ємності моделі глибокого навчання особливо важко, оскільки ефективна ємність обмежена можливостями алгоритму оптимізації.

Треба пам’ятати, що в той час як більш прості функції швидше узагальнені (щоб мати невеликий розрив між навчанням та тестовою помилкою), ми все одно повинні вибрати досить складну гіпотезу для досягнення низької помилки навчання. Зазвичай, помилка навчання зменшується, поки вона не асимптотизується до мінімально можливого значення помилки в той час як ємність моделі збільшується (якщо припустити, що міра помилки має мінімальне значення). Як правило, похибка узагальнення має U-подібну криву як функцію ємності моделі. Це і проілюстровано на рисунку 2.4.

Щоб досягти найбільш крайнього випадку довільно високої потужності, ми вводимо концепцію непараметричних моделей. Поки ми бачили лише параметричні моделі, такі як лінійна регресія. Параметричні моделі вивчають описану функцію вектором параметрів, розмір якого кінцевий і фіксований до того, як спостерігаються будь-які дані. Непараметричні моделі такого обмеження не мають.

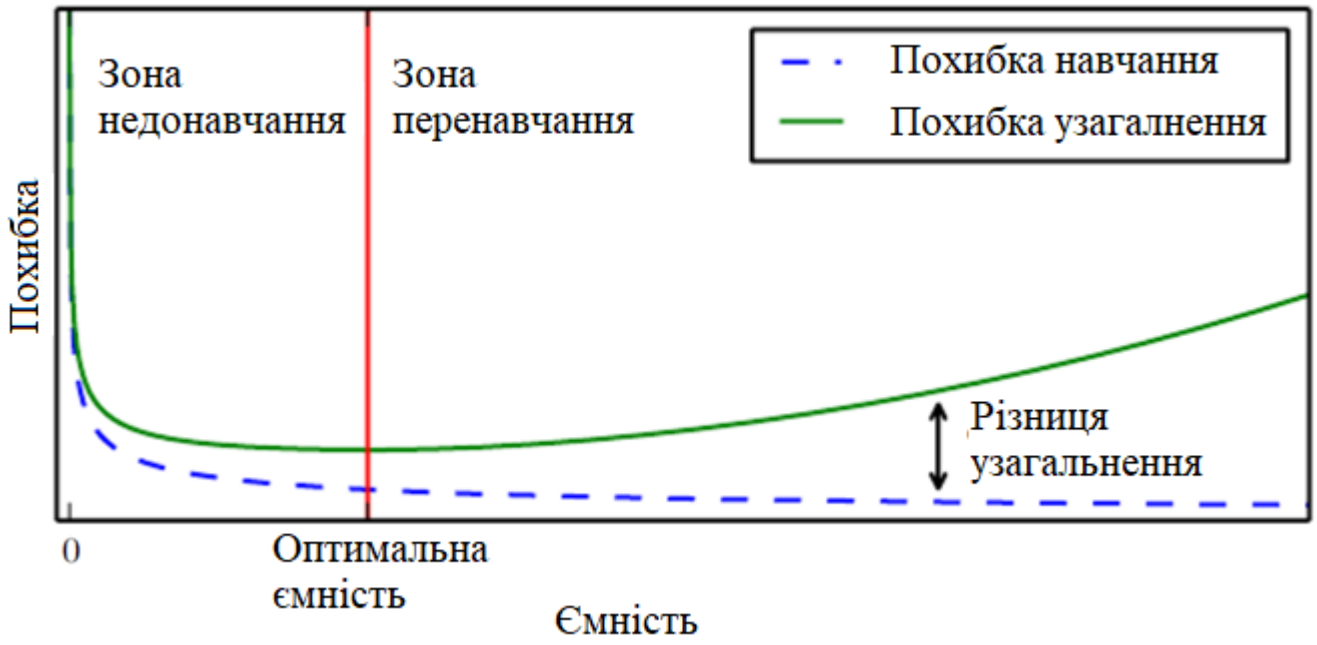


Рисунок 2.4 – Типовий взаємозв'язок між ємністю та помилкою.

Помилка навчання та тесту поводяться по-різному. У лівому кінці графіка помилка навчання та помилка узагальнення обидва високі. Це режим недонавченності. Коли ми збільшуємо ємність, помилка навчання зменшується, але розрив між помилкою навчання та генералізацією збільшується. Врешті-решт, розмір цього розриву переважає зменшення помилки у навчанні, і ми входимо в режим перенавченності, де ємність занадто велика, вище оптимальної ємності.

Іноді непараметричні моделі – це просто теоретичні абстракції (такі як алгоритм, який здійснює пошук усіх можливих розподілів ймовірностей), який не може реалізувати на практиці. Однак ми можемо також розробити практичні непараметричні моделі, роблячи їх складність функцією розміру навчального набору. Один приклад такого алгоритму є регресія найближчого сусіда. На відміну від лінійної регресії, яка має вектор ваг фіксованої довжини, модель регресії найближчого сусіда просто зберігає $\boldsymbol{X}$ і $\boldsymbol{y}$ з навчального набору. На запит класифікувати тестову точку $\boldsymbol{x}$, модель шукає найближчий запис у навчальному наборі та повертає пов'язану ціль регресії. Іншими словами, $\hat{y} = y_i$, де $i = \arg\min \left\| \boldsymbol{X}_{i,:} - \boldsymbol{x} \right\|_2^2$. Алгоритм також може бути узагальнений на метрику відстані,

відмінну від норми L^2, наприклад як засвоєні показники відстані [17]. Якщо алгоритм дозволений розірвати зв'язки шляхом усереднення значень y_i для всіх $\boldsymbol{X}_{i,:}$, які прив'язуються до найближчих, значить цей алгоритм здатний домогтися мінімальної можливої помилки в навчанні (що може бути більше нуля, якщо два однакових входи пов'язані з різними виходами) на будь-якому наборі регресії.

Нарешті, ми також можемо створити непараметричний алгоритм навчання, обернувши параметричний алгоритм навчання всередині іншого алгоритму, що збільшує кількість необхідних параметрів. Наприклад, ми могли б уявити зовнішній цикл навчання, що змінює ступінь полінома, засвоєного лінійною регресією поверх поліноміального розширення входу.

Ідеальна модель – це оракул, який просто знає справжній розподіл ймовірностей, що генерує дані. Навіть така модель все ще матиме деякі помилки у багатьох проблемах, оскільки все ще може бути якийсь шум у розподілі. У разі контрольованого навчання, відображення від $\boldsymbol{x}$ до $\boldsymbol{y}$ може бути по суті стохастичним, або у може бути детермінованою функцією, яка включає інші змінні, крім них включений у $\boldsymbol{x}$. Помилка, допущена оракулом, роблячи передбачення від істинного розподілу $p(\boldsymbol{x}, y)$ називається помилкою Байєса.

Помилка тренувань та узагальнення змінюється залежно від розміру навчального набору. Очікувана помилка узагальнення ніколи не може зростати, як кількість навчальних прикладів збільшується. Для непараметричних моделей більше даних дає кращу узагальненість до того часу як досягається найкраща можлива помилка. Будь-яка фіксована параметрична модель з меншою ніж оптимальна ємність буде асимптотом до значення помилки, що перевищує помилку Байєса. Модель може мати оптимальну ємність, але все ще є великий розрив між помилкою навчання та узагальнення. У цій ситуації ми можемо зменшити цей розрив, набравши більше навчальних прикладів.

2.1.6 Керовані алгоритми навчання

Керовані алгоритми навчання є, грубо кажучи, алгоритми навчання, які вчаться асоціювати деякий вхід з деяким результатом, задавши навчальний набір прикладів входів $\boldsymbol{x}$ та виходів $\boldsymbol{y}$. У багатьох випадках виходи $\boldsymbol{y}$ може бути важко зібрати автоматично, і його має забезпечити людина "Керівник", але цей термін все ще застосовується, навіть якщо цілі навчального набору були зібрані автоматично.

2.1.6.1 Імовірнісне кероване навчання

Більшість керованих алгоритмів навчання в цій книзі базуються на оцінці а розподіл ймовірності $p(y \mid \boldsymbol{x})$. Ми можемо це зробити просто, використовуючи максимальну оцінку ймовірності щоб знайти найкращий параметр вектора $\boldsymbol{\theta}$ для параметричного сімейства розподілів $p(y \mid \boldsymbol{x};\boldsymbol{\theta})$.

Ми вже бачили, що лінійна регресія відповідає сімейству

$$p(y \mid \boldsymbol{x};\boldsymbol{\theta}) = N\left(y;\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x},\boldsymbol{I}\right) \tag{2.18}$$

Можна узагальнити лінійну регресію до сценарію класифікації, визначивши різні сімейні розподіли ймовірностей. Якщо у нас є два класи, клас 0 і клас 1, тоді нам потрібно лише вказати ймовірність одного з цих класів. Імовірність класу 1 визначає ймовірність класу 0, оскільки ці два значення мають бути в сумі 1.

Нормальний розподіл за реальними числами, які ми використовували для лінійних регресій параметризований середнім значенням. Будь-яке значення, яке ми пропонуємо для цього, є дійсним. Розподіл на бінарну змінну трохи складніше, тому що її значення завжди має бути від 0 до 1. Одним із способів вирішення цієї проблеми є використання логістичної сигмоїдної функції для виведення лінійної функції в інтервал (0, 1) і інтерпретувати це значення як вірогідність:

$$p(y = 1 \mid \boldsymbol{x};\boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}\right) \tag{2.19}$$

Цей підхід відомий як логістична регресія (дещо дивна назва, оскільки використовується модель для класифікації, а не регресії).

У випадку лінійної регресії нам вдалося знайти оптимальні ваги за допомогою розв’язування нормальних рівнянь. Логістична регресія дещо складніша. Тут не є рішенням закритих форм для оптимальних ваг. Натомість треба шукати їх, максимізуючи логарифм ймовірності. Ми можемо це зробити, мінімізуючи логарифм ймовірності з використанням градієнтного спуску.

Ця ж стратегія може бути застосована до будь-якої проблеми, що контролюється, записуючи параметричне сімейство умовних розподілів ймовірностей на правильний вид змінних вводу та виводу.

2.1.6.2 Допоміжні векторні машини (Support Vector Machine, SVM)

Одним з найбільш впливових підходів до контрольованого навчання є допоміжна векторна машина [18]. Ця модель схожа на логістичну регресію тим, що керується лінійною функцією $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$. На відміну від логістичної регресії, векторний апарат підтримки не забезпечує ймовірностей, а лише виводить ідентичність класу. SVM прогнозує, що позитивний клас присутній, коли $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ додатний. Так само він передбачає, що негативний клас присутній коли $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ від’ємний.

Одним із ключових нововведень, пов'язаних із підтримкою векторних машин, є хитрість ядра. Трюк ядра полягає в тому, щоб спостерігати, що багато алгоритмів машинного навчання можуть писати виключно у вигляді точкових продуктів між прикладами. Наприклад, це може бути показано, що лінійна функція, використовувана машиною підтримки вектора, може бути переписаний як

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=b+\sum_{i=1}^{m}a_i\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}^{(i)} \tag{2.20}$$

де $\boldsymbol{x}^{(i)}$ – навчальний приклад, $\boldsymbol{a}$ – вектор коефіцієнтів. Переписування алгоритму навчання таким чином дозволяє нам замінити $\boldsymbol{x}$ на вихід даної ознаки функції $\phi(\boldsymbol{x})$ і крапковий добуток з функцією $k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right) = \phi(\boldsymbol{x}) \cdot \phi\left(\boldsymbol{x}^{(i)}\right)$ називається ядро. Оператор «·» являє собою внутрішній добуток, аналогічний $\phi(\boldsymbol{x})^{\mathrm{T}} \cdot \phi\left(\boldsymbol{x}^{(i)}\right)$. Для деяких функціональних просторів ми можемо не використовувати буквально векторний внутрішній продукт. В деяких нескінченних розмірних просторів, нам потрібно використовувати інші види внутрішніх виробів, для Наприклад, внутрішні продукти, засновані на інтеграції, а не підсумовуванні. Повний розвиток цих видів внутрішніх продуктів виходить за рамки цієї книги.

Замінивши крапкові добутки з оцінками ядра, ми можемо робити прогнози за допомогою функції

$$f(\boldsymbol{x}) = b + \sum_i a_i k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right) \tag{2.21}$$

Ця функція нелінійна щодо $\boldsymbol{x}$, але співвідношення між $\phi(\boldsymbol{x})$ і $f(\boldsymbol{x})$ лінійне. Також зв'язок між $\boldsymbol{\alpha}$ і $f(\boldsymbol{x})$ є лінійним. Функція на основі ядра точно рівнозначна попередній обробці даних шляхом застосування $\phi(\boldsymbol{x})$ на всі входи, потім вивчаючи лінійну модель у новому перетвореному просторі.

Хитрість ядра є потужною з двох причин. По-перше, це дозволяє нам вивчати моделі які є нелінійними як функція $\boldsymbol{x}$ з використанням опуклих методів оптимізації, які є гарантовано ефективно сходитися. Це можливо, тому що ми вважаємо ϕ фіксованим і оптимізуємо лише $\boldsymbol{\alpha}$, тобто алгоритм оптимізації може переглядати функцію рішення як лінійність в іншому просторі. По-друге, функція k ядра часто допускає реалізація, яка є значно більш ефективною в обчисленні, ніж наївна побудова двох $\phi(\boldsymbol{x})$ векторів і явно приймаючи їх точковий добуток.

У деяких випадках $\phi(\boldsymbol{x})$ навіть може бути нескінченним розміром, що призведе до нескінченної обчислювальної вартості наївного, явного підходу. У багатьох випадках $k(\boldsymbol{x}, \boldsymbol{x}')$ – нелінійна, відслідковуєма функція $\boldsymbol{x}$, навіть коли $\phi(\boldsymbol{x})$ нерозв'язна. Як приклад простору нескінченного розміру з простежуваним ядром, ми побудуємо особливість відображення $\phi(x)$ на невід'ємні цілі числа x. Припустимо, що це відображення повертає вектор, що містить x одиниць, за якими слідує нескінченно багато нулів. Ми можемо записати функцію ядра $k\left(x, x^{(i)}\right) = \min\left(x, x^{(i)}\right)$, що точно рівнозначно до відповідного нескінченномірного крапкового добутку.

Найбільш часто використовуваним ядром є ядро Гаусса

$$k(\boldsymbol{u}, \boldsymbol{v}) = N\left(\boldsymbol{u} - \boldsymbol{v}; 0, \sigma^2 \boldsymbol{I}\right) \tag{2.22}$$

де $N(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ – стандартна нормальна щільність. Це ядро також відоме як *функція радіальної основи*, оскільки його значення зменшується по лініях в $\boldsymbol{v}$ простір, що випромінюється назовні від $\boldsymbol{u}$. Ядро Гаусса відповідає крапковому добутку у нескінченномірному просторі, але виведення цього простору менше прямо, ніж у нашому прикладі мінімального ядра за цілими числами.

Ми можемо вважати, що ядро Гаусса є виконанням свого роду узгодження шаблону. Приклад навчання $\boldsymbol{x}$, пов'язаний з навчальною міткою y, стає шаблоном для класу y. Коли точка тесту знаходиться біля x відповідно до евклідової відстані, гауссове ядро має великий відгук, що вказує на те, що $\boldsymbol{x}'$ дуже схожий на шаблон $\boldsymbol{x}$. Потім модель додає велику вагу на пов'язаній навчальній етикетці y. Загалом, прогноз поєднує в собі багато таких навчальних міток, зважених на схожість відповідних навчальних прикладів.

Підтримка векторних машин – не єдиний алгоритм, який можна вдосконалити використовуючи трюк ядра. Багато інших лінійних моделей можна вдосконалити таким чином. Категорія алгоритмів, що використовують фокус ядра, відома як ядерні машини або ядерні методи [19].

Основним недоліком ядерних машин є те, що вартість оцінки функції лінійна рішення в кількості навчальних прикладів, тому що і-й приклад вносить у функцію рішення термін $a_i k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right)$. Підтримка векторних машин здатні пом'якшити це, вивчаючи $\boldsymbol{\alpha}$-вектор, який містить в основному нулі. Класифікація нового прикладу потребує оцінювання функції ядра лише для навчальних прикладів, які мають ненульовий a_i. Ці приклади навчання відомі як вектори підтримки.

Ядерні машини також страждають від високих обчислювальних витрат на навчання, коли набір даних великий. Ядерні машини с загальними ядрами намагаються добре узагальнити. Сучасне втілення глибокого навчання було покликане подолати ці обмеження ядерних машин. Нинішній ренесанс глибокого навчання розпочався, коли [20] продемонстрував, що нейронна мережа може перевершити SVM ядра RBF на еталоні MNIST.

2.1.6.3 Інші прості керовані алгоритми навчання

Ми вже коротко стикалися з іншим неімовірнісним контрольованим навчанням алгоритм, регресія найближчого сусіда. Загалом, k-найближчі сусіди сімейство прийомів, які можна використовувати для класифікації чи регресії. Як непараметричний алгоритм навчання, k-найближчих сусідів не обмежується фіксованим кількість параметрів. Зазвичай ми думаємо про алгоритм k-найближчих сусідів як не має жодних параметрів, а скоріше реалізує просту функцію дані про навчання. Насправді навіть не існує насправді навчального етапу чи процесу навчання. Натомість у тестовий час, коли ми хочемо отримати вихід у для нового тестового вводу x, ми знаходимо k-найближчих сусідів до $\boldsymbol{x}$ у навчальних даних $\boldsymbol{X}$. Потім повертаємо середнє значення відповідних значень у навчальному наборі. Це по суті працює будь-який тип контрольованого навчання, де ми можемо визначити середнє значення для значень у. У випадку класифікації ми можемо середньо оцінювати по одних гарячих кодових векторах с з $c_y = 1$ і $c_y = 0$ для всіх інших значень і. Тоді ми можемо інтерпретувати середні показники одне гарячі коди як

надання розподілу ймовірностей по класам. Як непараметричний алгоритм навчання, k-найближчий сусід може досягти дуже високої потужності. Наприклад, припустимо, у нас є класифікаційна задача багатокласового рівняння та вимірювати ефективність з 0-1 втрати. У цьому налаштуванні-найближчий сусід збігається, щоб подвоїти 1 помилку Байєса як кількість прикладів тренінгу наближається до нескінченності. Помилка, що перевищує Байєса помилка є результатом вибору одного сусіда, розриваючи зв'язки між собою далекі сусіди випадковим чином. Коли є нескінченні дані про навчання, усі бали x матиме нескінченно багато тренувальних сусідів на відстані нуля. Якщо ми дозволимо алгоритм використання всіх цих сусідів для голосування, а не випадкового вибору одного з них процедура сходить до рівня помилок Байєса. Висока потужність k-найближчі сусіди дозволяють йому отримати високу точність з урахуванням великого навчального набору. Однак це робиться за великих обчислювальних витрат, і це може узагальнити дуже погано дали невеликий, обмежений навчальний набір. Одна слабкість k-найближчих сусідів полягає в тому, що це не можна дізнатися, що одна ознака є більш дискримінаційною, ніж інша. Наприклад, уявіть, що у нас є регресійне завдання з $\boldsymbol{x} \in \square^{100}$, виведене з ізотропного розподілу Гаусса, але для виходу має значення лише одна змінна x_1. Припустимо Крім того, ця функція просто кодує вихід безпосередньо, тобто $y = x_1$ у всіх справ. Найближчий регрес сусіда не зможе виявити цю просту картину. Найближчий сусід більшості точок $\boldsymbol{x}$ визначатиметься великою кількістю має від x_2 до x_{100}, а не по одинокій функції x_1. Таким чином вихід на малий навчальні набори, по суті, будуть випадковими.

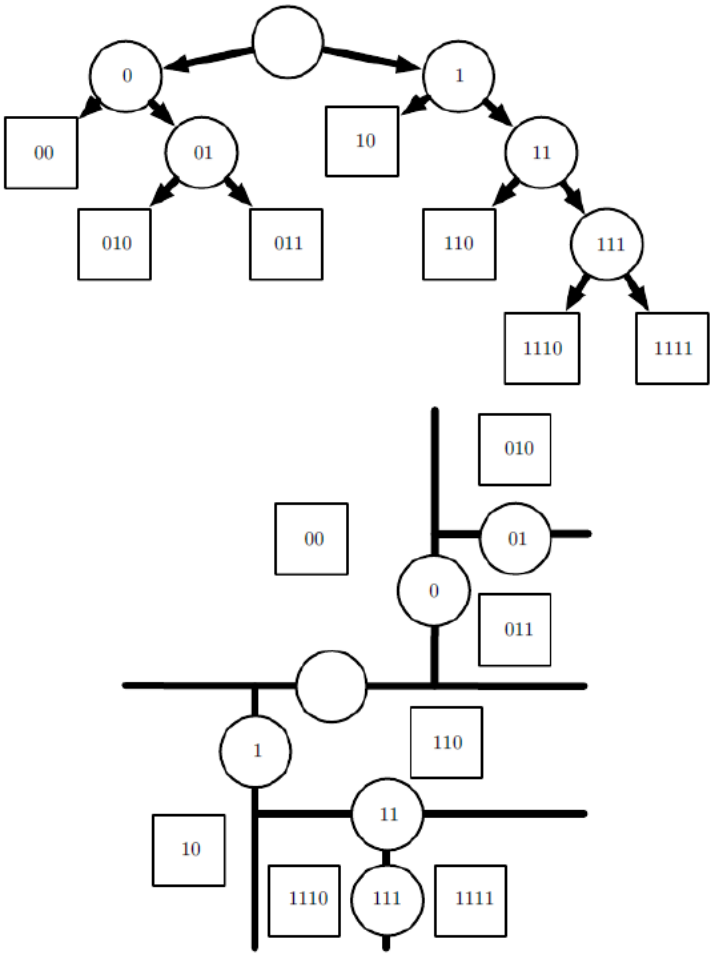


Рисунок 2.5 – Діаграма, що описує роботу дерева рішень

Кожен вузол дерева вибирає надіслати приклад введення дочірньому вузлу зліва (0) або дочірньому вузлу справа (1). Внутрішні вузли малюються у вигляді кіл, а листкові вузли – у вигляді квадратів. Кожен вузол є відображається з отриманим двійковим ідентифікатором рядка, відповідним його положенню в дереві додавши трохи до свого батьківського ідентифікатора (0 = виберіть лівий або верхній, 1 = виберіть правий або нижній). (Внизу) Дерево ділить простір на регіони. 2D площина показує, як дерево рішень може розділити $\square^2$. Вузли дерева побудовані в цій площині, з кожним внутрішнім вузлом намальовані вздовж лінії, що розділяється, вона використовується для категоризації прикладів та вузлів листів, намальованих у центр регіону прикладів, які вони отримують. Результатом є кусочно-постійна функція, з однією штукою на листочок. Кожен лист вимагає хоча б одного прикладу навчання,

щоб визначити, так воно і є не можливо, щоб дерево рішень дізналося функцію, яка має більше локальних максимумів, ніж кількість прикладів навчання.

Інший тип алгоритму навчання, який також розбиває вхідний простір на регіони і має окремі параметри для кожного регіону – це дерево рішень [21] та його безліч варіантів. Як показано на рис. 3.5, кожен вузол дерева рішень асоціюється з областю у вхідному просторі, і внутрішні вузли розбивають цю область в один підрегіон для кожного дочірнього вузла (як правило, використовуючи вирівнювання по осі вирізати). Простір, таким чином, поділяється на регіони, що не перекриваються, один на один відповідність між листковими вузлами та областями введення. Кожен вузол листя зазвичай карта кожну точку своєї області вводу до одного виходу. Дерева рішень зазвичай навчені спеціалізованими алгоритмами, які виходять за межі цієї книги. Алгоритм навчання можна вважати непараметричним, якщо дозволено вивчати дерево довільного розміру, хоча дерева рішень зазвичай регулюються обмеженнями розміру які перетворюють їх на параметричні моделі на практиці. Дерева рішень такими, якими вони є як правило, використовується з розрізами по осі і постійними виходами всередині кожного вузла, боротьба за вирішення деяких проблем, легких навіть для логістичного регресу. Для Наприклад, якщо у нас є двокласна проблема, і позитивний клас виникає де завгодно, межа рішення не є вирівняною по осі. Таким чином, дерево рішень буде потрібно наблизити межу рішення з багатьма вузлами, реалізуючи крок функція, яка постійно переходить вперед-назад по справжній функції прийняття рішень з вирівняними по осях кроками.

Як ми бачили, для передбачувачів найближчих сусідів та дерев рішень є багато обмежень. Тим не менш, вони корисні алгоритми навчання при обчисленні ресурси обмежені. Ми також можемо побудувати інтуїцію для більш досконалих вивчення алгоритмів, обмірковуючи подібність та відмінності між складні алгоритми та базові лінії k-NN або дерева рішень.

2.1.7 Некеровані алгоритми навчання

Некеровані алгоритми – це лише ті, що вивчають ознаки, але не сигнал керування. Відмінність між керованими та некерованими алгоритмами формально та жорстко не визначені, оскільки немає об'єктивного тесту на розрізнення того, чи є значення ознакою чи ціллю, що надається керівником. Неофіційно некероване навчання стосується більшості спроб видобутку інформації з розподілу, що не потребує людської праці для коментування прикладів. Термін, як правило, пов'язаний з оцінкою щільності, навчанням перетягувати зразки з розподілу, навчання позначати дані з деякого розподілу, знаходження множини, що знаходиться поблизу даних, або групування даних на групи пов'язаних прикладів.

Класичне завдання некерованого навчання – знайти найкраще представлення даних. Під «найкращим» можуть означати різні речі, але в цілому шукається представлення, яке зберігає якомога більше інформації про $\boldsymbol{x}$ при цьому дотримуючись певного штрафу чи обмеження, спрямованого на спрощення представництва або доступніший, ніж сам $\boldsymbol{x}$.

Існує кілька способів визначення більш простого представлення. Три з найпоширеніших – маломірні представлення, розріджені зображення і незалежні представлення. Маломірні представлення намагаються зробити стиснути якомога більше інформації про x у меншому представленні. Розріджені представлення [22] вбудовують набір даних у представлення, записи якого є переважно нулі для більшості входів. Зазвичай використання розріджених представлень вимагає збільшення розмірності представлень, так що представлення, ставши в основному нулями, не відкидає занадто багато інформації. Це призводить до отримання загальної структури представлень, яка прагне розподіляти дані по осях простору представлення. Незалежні представлення намагаються роз'єднатися джерела варіації, що лежать в основі розподілу даних, так, що розміри представництва статистично незалежні.

Звичайно, ці три критерії, безумовно, не виключають один одного. Маломірні представлення часто дають елементи, які мають меншу або слабку залежність ніж оригінальні багатовимірні дані. Це тому, що один із способів зменшити розмір представлення – це знайти та видалити надлишки. Ідентифікація і видалення більшої

надмірності дозволяє алгоритму зменшення розмірності досягти більшої компресії, аніж відкидаючи менше інформації.

2.1.7.1 Аналіз основних компонентів (Principal Component Analysis, PCA)

Алгоритм аналізу основних компонентів забезпечує засоби стискання даних. Ми також можемо розглядати PCA як некерований алгоритм навчання, який вивчає представлення даних. Це представлення засноване на двох з критеріїв простого представлення. PCA вивчає представлення, яке має менші розміри, ніж заданий вхід. Він також вчить представлення, елементи якого не мають лінійної кореляції один з одним. Це і є першим кроком до критерію навчальних представлень, елементи яких є статистично незалежні. Для досягнення повної незалежності, алгоритм навчання представлень також повинен видаляти нелінійні зв'язки між змінними. Робота PCA продемонстрована на рисунку 2.6.

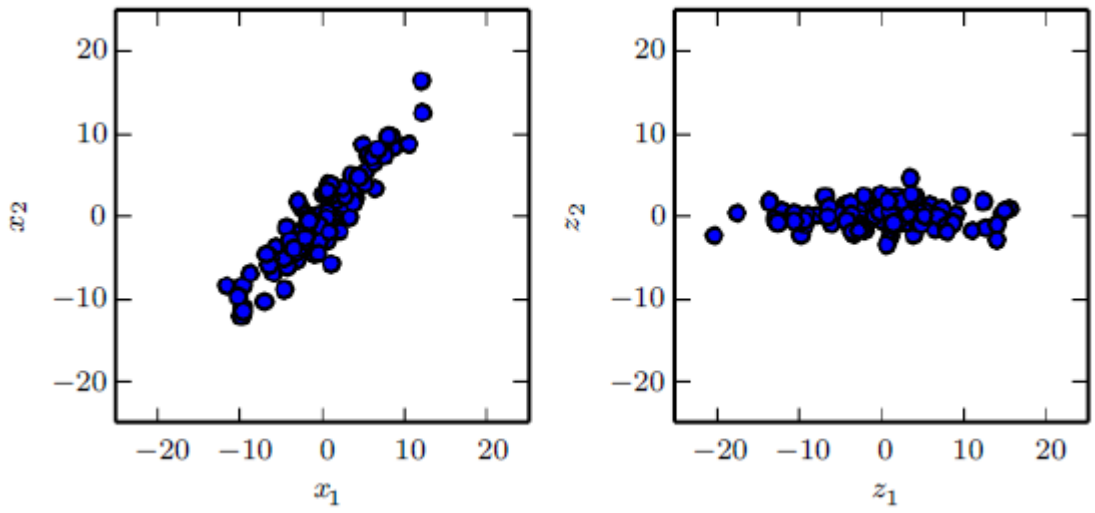


Рисунок 2.6 – Робота PCA

PCA вивчає лінійну проекцію, яка вирівнює напрямок найбільшої дисперсії з осями нового простору. Оригінальні дані складаються з зразків $\boldsymbol{x}$. У цьому просторі, дисперсія може виникати вздовж напрямків, не орієнтованих на осі. Трансформовані

дані $\boldsymbol{z} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}$ тепер сильно змінюються вздовж осі z_1. Напрямок другої дисперсії тепер вже уздовж z_2.

PCA вивчає ортогональне, лінійне перетворення даних, що проектує вхід ***x*** до подання *z*. Можна вивчити одновимірне представлення, яке найкраще реконструює оригінальні дані (в сенсі середньої квадратичної помилки) і що це представлення насправді відповідає першому принциповому компоненту даних. Таким чином, можна використовувати PCA як простий та ефективний метод зменшення розмірності, який зберігає стільки інформації в даних, наскільки це можливо.

Розглянемо дизайн-матрицю *m* × *n* розмірності ***X***. Будемо вважати, що дані мають середнє нульове значення, $\mathbb{E}\left[\boldsymbol{x}\right] = 0$. Якщо це не так, дані можуть легко центруватись, віднімаючи середнє значення з усіх прикладів на етапі попередньої обробки.

Неупереджена матриця коваріації зразка, пов'язана з ***X***, задається:

$$\mathrm{Var}\left[\boldsymbol{x}\right] = \frac{1}{m-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \tag{2.23}$$

PCA знаходить подання (через лінійне перетворення) $\boldsymbol{z} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}$, де $\mathrm{Var}\left[z\right]$– діагональ.

Основними компонентами дизайн-матриці ***X*** задані власними векторами $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. З цього погляду

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^{\mathrm{T}} \tag{2.24}$$

Основні компоненти також можуть бути отримані шляхом розкладання сингулярного значення. Зокрема, вони є правильними сингулярними векторами ***X***. Щоб побачити це, нехай ***W*** – правильні сингулярні вектори при розкладанні

$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}}$. Потім ми відновимо початкове рівняння власного вектора з $\boldsymbol{W}$ як основою власного вектора:

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}}\right)^{\mathrm{T}}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}} = \boldsymbol{W}\boldsymbol{\Sigma}^{2}\boldsymbol{W}^{\mathrm{T}} \tag{2.25}$$

SVD корисно показати, що PCA призводить до діагоналі Var[z]. Використання SVD $\boldsymbol{X}$, ми можемо виразити дисперсію $\boldsymbol{X}$ як:

$$\mathrm{Var}\left[\boldsymbol{x}\right] = \frac{1}{m-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \tag{2.26}$$

$$= \frac{1}{m-1}\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}}\right)^{\mathrm{T}}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}} \tag{2.27}$$

$$= \frac{1}{m-1}\boldsymbol{W}\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^{\mathrm{T}} \tag{2.28}$$

$$= \frac{1}{m-1}\boldsymbol{W}\boldsymbol{\Sigma}^{2}\boldsymbol{W}^{\mathrm{T}} \tag{2.29}$$

де використовується той факт, що $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{I}$, тому що матриця $\boldsymbol{U}$ визначення одиничного значення визначається як ортонормальне. Це показує, що якщо взяти $\boldsymbol{z} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}$, можна гарантувати, що коваріація $\boldsymbol{z}$ є діагональною, як потрібно:

$$\mathrm{Var}\left[z\right] = \frac{1}{m-1}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z} \tag{2.30}$$

$$= \frac{1}{m-1}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{W} \tag{2.31}$$

$$= \frac{1}{m-1}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{\Sigma}^{2}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} \tag{2.32}$$

$$= \frac{1}{m-1}\boldsymbol{\Sigma}^{2} \tag{2.33}$$

де цього разу використовується той факт, що $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{I}$, знову ж таки з визначення SVD.

Наведений вище аналіз показує, що коли проектуються дані $\boldsymbol{x}$ на $\boldsymbol{z}$, через лінійне перетворення $\boldsymbol{W}$, отримане подання має діагональну матрицю коваріації (за даними $\boldsymbol{\Sigma}^2$), що негайно означає, що окремі елементи $\boldsymbol{z}$ є взаємно некорельовані.

Ця здатність PCA перетворювати дані в представлення, де елементи взаємно некорельовані є дуже важливою властивістю PCA. Це просто приклад представлення, яке намагається роз'єднати невідомі фактори варіації, що лежать в основі даних. У випадку PCA це роз'єднання займає місце знаходження обертання вхідного простору (описана $\boldsymbol{W}$), що вирівнює основні дисперсійні осі з основою нового пов'язаного простору представлення зі $\boldsymbol{z}$.

Хоча кореляція є важливою категорією залежності між елементами даних ми також зацікавлені в навчанні представлень, які більше роз'єднують більш складні форми залежностей ознак. Для цього знадобиться більше, ніж можна зробити за допомогою простого лінійного перетворення.

2.1.7.2 Кластеризація методом *k*-середніх

Інший приклад простого алгоритму навчання подання – це кластеризація методом *k*-середніх. Алгоритм кластеризації методом *k*-середніх ділить навчальний набір на *k* різних кластерів прикладів, які знаходяться поруч один з одним. Таким чином, можна вважати, що алгоритм надає *k*-мірний вектор унітарного коду $\boldsymbol{h}$, що представляє вхід $\boldsymbol{x}$. Якщо $\boldsymbol{x}$ належить кластеру *i*, тоді $h_i = 1$ і всі інші записи представлення $\boldsymbol{h}$ є нульові.

Унітарний код, наданий кластеризацією методом *k*-середніх, є прикладом рідкого представлення, оскільки більшість його записів дорівнює нулю для кожного входу. Унітарні коди є крайнім прикладом розріджених уявлень, які втрачають багато переваг розподіленого представлення. Унітарний код все ще надає деякі статистичні переваги (це передає думку, що всі приклади в одному кластері є схожі між собою) і

це надає обчислювальну перевагу, що все представлення може бути захоплено одним цілим числом.

Алгоритм кластеризації методом *k*-середніх працює шляхом ініціалізації *k* різних центроїдів $\left\{\boldsymbol{\mu}^{(1)},...,\boldsymbol{\mu}^{(k)}\right\}$ до різних значень, потім чергуючи два різних кроки до зближення. На одному кроці кожен приклад навчання присвоюється кластеру *i*, де *i* – індекс найближчого центроїду $\boldsymbol{\mu}^{(i)}$. На іншому кроці кожен центроїд $\boldsymbol{\mu}^{(i)}$ оновлюється до середнього значення всіх прикладів навчання $\boldsymbol{x}^{(j)}$, призначених кластеру *i*.

Одна з труднощів, пов’язаних із кластеризацією, полягає в тому, що проблема кластеризації погано поставлена, в тому сенсі, що не існує єдиного критерію, який визначає, наскільки добре кластеризація даних відповідає реальному світу. Можна виміряти властивості кластеризації, такі як середня евклідова відстань від центру кластера до члена кластеру. Це дозволяє нам сказати, наскільки добре ми здатні реконструювати навчальні дані із кластерних завдань. Ми не знаємо, наскільки добре призначення кластерів відповідають властивостям реального світу. Тим більше, що там може бути багато різних кластерів, які всі добре відповідають певній властивості реального світу. Ми можемо сподіватися знайти кластеризацію, яка стосується однієї функції, але отримати іншу, однаково допустиму кластеризацію, яка не стосується нашого завдання. Наприклад, припустимо, що ми запускаємо два алгоритми кластеризації на наборі даних, що складається з зображення червоних вантажівок, зображення червоних автомобілів, зображення сірих вантажівок та зображення сірих автомобілів. Якщо ми попросимо кожен алгоритм кластеризації знайти два кластери, один алгоритм може знайти групу автомобілів та групу вантажних автомобілів, а інша може знайти групу червоних транспортних засобів та групу сірих транспортних засобів. Припустимо, ми також запускаємо третій алгоритм кластеризації, якому дозволяється визначати кількість кластерів. Це може призначити приклади до чотирьох кластерів: червоні автомобілі, червоні вантажівки, сірі автомобілі та сірі вантажівки. Ця нова кластеризація зараз принаймні фіксує інформацію про обидва

атрибути, але вона втратила інформацію про схожість. Червоні автомобілі знаходяться в іншій групі від сірих автомобілів, так само, як вони в іншій групі від сірих вантажних автомобілів. Вихід алгоритму кластеризації не говорить нам про те, що червоні автомобілі більше схожі на сірі автомобілі ніж на сірі вантажівки. Вони відрізняються від обох речей, і це все, що ми знаємо.

Ці речі показують частину причин, чому ми віддаємо перевагу розподіленим представленням, а не унітарним представленням. Розподілене представлення могло мати два атрибути для кожного транспортного засобу: один представляє його колір і один представляє чи це машина, чи вантажівка. Досі не зовсім зрозуміло, яке розподілене представлення є оптимальним (алгоритм навчання не може знати, які саме атрибути нас цікавлять – колір і вид транспортного засобу чи виробник і вік), але наявність багатьох ознак знижує навантаження на алгоритм, щоб відгадати, який саме атрибут нас хвилює, і дозволяє нам вимірювати подібність між предметами дрібнозернистим шляхом порівняння багатьох ознак замість того, щоб просто перевірити, чи відповідає один атрибут.

2.1.8 Побудова алгоритму машинного навчання

Майже всі алгоритми глибокого навчання можна охарактеризувати як конкретні екземпляри досить простого рецепту: поєднайте специфікацію набору даних, функцію витрат, процедуру оптимізації та модель.

Наприклад, алгоритм лінійної регресії об'єднує набір даних, що складається з $\boldsymbol{X}$ і $\boldsymbol{y}$, функції витрат

$$J(\boldsymbol{w},b) = -\mathbb{E}_{\mathbf{x},\mathrm{y}\sim \hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(y \mid \boldsymbol{x}), \tag{2.34}$$

специфікації моделі $p_{\mathrm{model}}(y \mid \boldsymbol{x}) = \mathcal{N}(y; \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w} + b, 1)$, і, в більшості випадків, алгоритму оптимізації, визначений рішенням, де градієнт вартості дорівнює нулю використовуючи звичайні рівняння.

Зрозумівши, що ми можемо замінити будь-який з цих компонентів переважно незалежно від інших, ми можемо отримати дуже широке розмаїття алгоритмів.

Функція витрат зазвичай включає щонайменше один вираз, який спричиняє процес навчання виконувати статистичне оцінювання. Найпоширеніша функція витрат – це негативна функція правдоподібності, так що мінімізація функції витрат викликає максимум оцінки ймовірності.

Функція витрат може також включати додаткові вирази, такі як регуляризація умови. Наприклад, ми можемо додати зменшення ваги до функції лінійної регресії отримати

$$J\left(\boldsymbol{w},b\right)=\lambda\left\|\boldsymbol{w}\right\|_2^2-\mathbb{E}_{\mathbf{x},\mathrm{y}\sim\dot{p}_{\mathrm{data}}}\log p_{\mathrm{model}}\left(y\mid\boldsymbol{x}\right) \tag{2.35}$$

Це все ще дозволяє оптимізувати закриту форму.

Якщо ми змінимо модель на нелінійну, то більшість функцій витрат більше не зможуть бути оптимізовані в закритому вигляді. Це вимагає від нас вибрати ітеративну числову процедуру оптимізації, така як спуск градієнта.

Рецепт побудови алгоритму навчання за допомогою комбінування моделей, витрат та алгоритмів оптимізації підтримують як кероване, так і некероване навчання. Приклад лінійної регресії показує, як підтримувати кероване навчання. Некероване навчання може бути підтримане шляхом визначення набору даних, що містить лише ***X,*** та надання відповідних некерованих вартості та моделі. Наприклад, ми можемо отримати перший вектор PCA, вказавши, що наша функція втрати

$$J\left(\boldsymbol{w}\right)=\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left\|\boldsymbol{x}-r\left(\boldsymbol{x};\boldsymbol{w}\right)\right\|_2^2 \tag{2.36}$$

в той час як наша модель визначена мати $\boldsymbol{w}$ з першою нормою та функцією відновлення $r\left(\boldsymbol{x}\right)=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\boldsymbol{w}$.

У деяких випадках функція витрат може бути функцією, яку насправді ми не можемо оцінити з обчислювальних причин. У цих випадках ми ще можемо приблизно мінімізувати його, використовуючи ітераційну чисельну оптимізацію, доки ми маємо певний спосіб наближення його градієнтів.

Більшість алгоритмів машинного навчання використовують цей рецепт, хоча це не може бути одразу очевидним. Якщо алгоритм машинного навчання здається унікальним або ручним, його, як правило, можна розуміти як алгоритму, що використовує оптимізатор спеціального випадку. Деякі моделі, такі як дерева рішень або метод k-середніх, вимагають оптимізаторів спеціального випадку, оскільки їхні функції витрат мають плоскі регіони, що роблять їх непридатними для мінімізації градієнтними оптимізаторами. Розуміння, що більшість алгоритмів машинного навчання можна описати, використовуючи цей рецепт, допомагає бачити різні алгоритми як частину таксономічності методів для виконання суміжних завдань, які працюють з подібних причин ніж довгий перелік алгоритмів, кожен з яких має окремі обґрунтування.

2.2 Глибоке навчання

Цей підрозділ підсумовує стан сучасного глибокого навчання таким, яким він використовується для вирішення практичних застосувань.

Глибоке навчання має довгу історію та багато праць. Кілька підходів були запропоновані, які ще не принесли плодів. Кілька перспективних цілей ще мають бути реалізовані.

Сучасне глибоке навчання забезпечує дуже потужну основу для керованого навчання. Додавши більше шарів і більше одиниць всередині шару, глибока мережа може представляти функції підвищеної складності. Більшість завдань, що складаються з відображення вхідного вектора у вихідний вектор, і людина може легко і швидко їх зробити, можуть виконуватись за допомогою глибокого навчання, даючи досить великі моделі та достатньо великі набори даних мічених навчальних прикладів. Інші завдання, які неможливо описати як асоціювання одного вектора з

іншим, або для людини це досить важко або потребувало би часу на роздуми, щоб виконати завдання, поки що залишаються за рамками глибокого навчання.

2.2.1 Deep Feedforward Networks

Глибокі мережі прямого поширення, їх також часто називають нейронними мережами або багатошаровими перцептронами (MLPs) – це найважливіша модель глибокого навчання. Мета мережі прямого поширення – це уточнення деякої функції f^*. Наприклад, для класифікатору, $y = f^*(\boldsymbol{x})$ відображає вхід $\boldsymbol{x}$ до категорії y. Мережа прямого поширення визначає відображення $y = f(\boldsymbol{x};\boldsymbol{\theta})$ і вивчає значення параметрів $\boldsymbol{\theta}$, які отримують результат в кращому уточненні функції.

Ці моделі називаються прямого поширення тому, що інформація протікає через функцію, що оцінюється з $\boldsymbol{x}$, за допомогою проміжних обчислень, що використовуються для визначення f, і на вихід $\boldsymbol{y}$. Немає зворотного зв'язку, в якому виходи моделі подаються назад у себе. Коли мережі прямого поширення розширяються, щоб включати зв'язки зворотного зв'язку, вони називаються рекурентними нейронними мережами.

Мережі прямого поширення мають надзвичайно важливе значення для тих, хто практикує машинне навчання. Вони складають основу багатьох важливих комерційних додатків. Наприклад, згорткові мережі, що використовуються для розпізнавання об'єктів з фотографій, є спеціалізованим видом мережі прямого поширення. Мережі прямого поширення – це концептуальна сходинка на шляху до рекурентних мереж.

Нейронні мережі прямого поширення, називаються мережами, оскільки вони зазвичай представлені як складені разом багато різних функцій. Модель асоціюється з направленим ациклічним графом, що описує, як функції складаються разом. Наприклад, у нас може бути три функції $f^{(1)}$, $f^{(2)}$, $f^{(3)}$, з'єднані в а ланцюг, утворюючи

$f(\boldsymbol{x}) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)$. Ці ланцюгові конструкцій найбільше поширені як структури нейронних мереж. У цьому випадку $f^{(1)}$ називається першим шаром мережі, $f^{(2)}$ називається другим шаром, тощо. Загальна довжина ланцюга надає глибину моделі. Саме з цієї термінології виникає назва «глибоке навчання». Останній рівень мережі прямого поширення називається вихідний шар. Під час тренувань нейронної мережі ми рухаємо $f(\boldsymbol{x})$, щоб відповідати $f^*(\boldsymbol{x})$. Дані навчання дають нам шумні, приблизні приклади $f^*(\boldsymbol{x})$, оцінених на різних навчальних точках. Кожен приклад $\boldsymbol{x}$ супроводжується міткою $y \approx f^*(\boldsymbol{x})$. Навчальні приклади безпосередньо вказують, що повинен робити вихідний шар у кожній точці $\boldsymbol{x}$; він повинен виробляти значення, близьке до y. Поведінка інших шарів не вказана безпосередньо навчальними даними. Алгоритм навчання повинен вирішити як використовувати ці шари для отримання бажаного результату, але дані навчання не кажуть, що повинен робити кожен окремий шар. Натомість алгоритм навчання повинен вирішити, як використовувати ці шари, щоб найкраще здійснити наближення f^*. Через те, що навчальні дані не показують бажаного результату для кожного з цих шарів, ці шари називають прихованими шарами.

Нарешті, ці мережі називаються нейронними, оскільки вони дещо нагадують нейронауку. Кожен прихований шар мережі, як правило, має векторне значення. Розмірність цих прихованих шарів визначає ширину моделі. Кожен елемент вектора може бути інтерпретований як елемент, що грає роль, аналогічну нейрону. Замість того, щоб думати про шар як такий, що представляє одну функцію вектор-вектор, ми також можемо думати про шар як такий, що складається з безлічі одиниць, які діють паралельно, кожен з яких представляє функцію вектор-скаляр. Кожна одиниця нагадує нейрон в тому сенсі, що воно отримує вхід від багатьох інших одиниць і обчислює власне значення активації. Ідея використання багатьох шарів векторного представлення походить з нейронауки. Вибір функцій $f^{(i)}(\boldsymbol{x})$, що використовуються для обчислення цих представлень також вільно керуються нейронауковими

спостереженнями про функції, які обчислюють біологічні нейрони. Однак сучасна нейронна мережа дослідження керуються багатьма математичними та інженерними дисциплінами та мета нейронних мереж – не ідеально моделювати мозок. Найкраще думати про мережі прямого поширення, як машини наближення функцій, призначені для досягнення статистичного узагальнення, періодично черпаючи певну думку з того, що ми можемо знати про мозок, а не як про моделі функціонування мозку.

Один із способів зрозуміти мережі прямого поширення – почати з лінійних моделей і розглянути, як подолати їх обмеження. Лінійні моделі, такі як логістичні регресія та лінійна регресія є привабливими, оскільки вони можуть бути ефективно придатними і надійними, або в закритому вигляді, або з опуклою оптимізацією. Лінійні моделі також мають очевидний дефект, що ємність моделі обмежена лінійними функціями, тому модель не може зрозуміти взаємодію між двома вхідними змінними.

Для розширення лінійних моделей для представлення нелінійних функцій $\boldsymbol{x}$ ми можемо застосувати лінійну модель не до самого $\boldsymbol{x}$, а до перетвореного входу $\phi(\boldsymbol{x})$, де ϕ – нелінійне перетворення. Також ми можемо застосувати ядерний метод, щоб отримати нелінійний алгоритм навчання, на неявному застосуванні відображення ϕ. Ми можемо вважати ϕ як надання набору функцій, що описують $\boldsymbol{x}$, або як надання нового представлення для $\boldsymbol{x}$.

Питання полягає в тому, як вибрати відображення ϕ.

- Одним із варіантів є використання дуже загального ϕ, такого як нескінченномірне ϕ, що неявно використовується ядерними машинами на основі ядра радіально-базисної функції. Якщо $\phi(\boldsymbol{x})$ є достатньо високовимірною, ми завжди можемо мати достатню ємність, щоб вмістити його навчальний набір, але узагальнення до тестового набору часто залишається поганим. Дуже загальні відображення ознак зазвичай базуються лише на локальному принципі розмитості і не кодують достатньо пріоритетної інформації для вирішення передових проблеми.
- Інший варіант – вручну розробити ϕ. До появи глибокого навчання, це був домінуючий підхід. Цей підхід вимагає десятиліть зусилля людини для кожного

окремого завдання, при тому, що практиканти спеціалізуються на різних сферах, як розпізнавання мови чи комп'ютерний зір, і дуже мала передача між сферами.

- Стратегія глибокого навчання полягає у навчанні ϕ. У такому підході у нас є модель $y = f\left(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{w}\right) = \phi\left(\boldsymbol{x};\boldsymbol{\theta}\right)^{\mathrm{T}}\boldsymbol{w}$. Тепер у нас є параметри $\boldsymbol{\theta}$, які ми використовуємо для вивчення ϕ з широкого класу функцій і параметрів $\boldsymbol{w}$, які відображаються від $\phi\left(\boldsymbol{x}\right)$ до бажаний вихід. Це приклад глибокої мережі прямого поширення, з ϕ, що визначає прихований шар. Цей підхід є лише одним із трьох, що відмовляється від вирішення опуклості проблеми навчання, але переваги переважають недоліки. У цьому підході ми параметризуємо представлення як $\phi\left(\boldsymbol{x};\boldsymbol{\theta}\right)$ і використовуємо алгоритм оптимізації, щоб знайти $\boldsymbol{\theta}$, що відповідає гарному представленню. Якщо ми хочемо, такий підхід може перейняти перевагу першого підходу, будучи дуже загальним – ми робимо це за допомогою дуже широкого сімейства $\phi\left(\boldsymbol{x};\boldsymbol{\theta}\right)$. Такий підхід може також отримати перевагу другого підходу. Практикуючі люди можуть кодувати свої знання, щоб допомогти узагальненню проектуванням сімей $\phi\left(\boldsymbol{x};\boldsymbol{\theta}\right)$, які, як вони очікують, матимуть хороші результати. Перевага полягає в тому, що дизайнеру потрібно лише знайти правильне загальне сімейство функцій, а не просто правильну функцію.

Цей загальний принцип вдосконалення моделей за допомогою вивчення ознак поширюється за межі мереж прямого поширення. Мережі прямого поширення – це застосування цього принципу для вивчення детермінованих відображень від $\boldsymbol{x}$ до $\boldsymbol{y}$, у яких відсутній зворотний зв'язок.

2.2.2 Дизайн архітектури

Ключовим моментом дизайну нейронних мереж є визначення архітектури. Слово архітектура відноситься до загальної структури мережі: скільки одиниць, які він повинен мати, і як ці одиниці повинні бути з'єднані один з одним.

Більшість нейронних мереж організовані в групи одиниць, які називаються шарами. Більшість архітектур нейронних мереж розташовують ці шари у ланцюговій структурі, у якій кожний шар є функцією шару, який передував йому. У цій структурі перший шар задається

$$h^{(1)} = g^{(1)}\left(W^{(1)T}x + b^{(1)}\right) \tag{2.37}$$

другий шар задається

$$h^{(2)} = g^{(2)}\left(W^{(2)T}h^{(1)} + b^{(2)}\right) \tag{2.38}$$

і так далі.

У цих ланцюгових архітектурах основними архітектурні задачами є вибрати глибину мережі та ширину кожного шару. Мережа з навіть одним прихованим шаром є достатньою для розміщення навчального набору. Глибші мережі часто можуть використовувати набагато менше одиниць на рівень і набагато менше параметрів і часто узагальнюють до тестового набору, але також їх часто важче оптимізувати. Ідеальну мережеву архітектуру для завдання потрібно знайти за допомогою експерименту, що керується моніторингом помилки набору перевірки.

2.2.2.1 Властивості універсального наближення та глибина

Лінійна модель, відображаючи від ознак до виходів за допомогою матричного множення, може за визначенням представляти лише лінійні функції. Вона має перевагу в тому сенсі, що її легко навчити, тому що багато функцій втрат призводять до опуклих проблем оптимізації, коли застосовується до лінійних моделей. На жаль, ми часто хочемо навчити нелінійні функції.

На перший погляд, ми можемо припустити, що вивчення нелінійної функції вимагає розробки спеціалізованого сімейства моделей для того виду нелінійності,

який ми хочемо вивчити. На щастя, мережі прямого поширення із прихованими шарами забезпечують універсальну наближаючу структуру. Зокрема, теорема про універсальне наближення [23] стверджує, що мережа прямого поширення з лінійним вихідним шаром і щонайменше одним прихованим шаром з будь-якою функцією активації «розчавлення» (наприклад, логістична функція активації сигмоїдів) може наближати будь-яку вимірювану Борелем функцію від одного кінцево-мірного простору до іншого з будь-якою потрібною ненульовою кількістю помилок за умови надання мережі достатньо прихованих одиниць. Похідні мереж прямого поширення також можуть наближати похідні функцій досить добре [24]. Будь-яка безперервна функція на замкнутій та обмеженій підмножині $\square^n$ вимірюється Борелем і тому може бути апроксимована нейронною мережею. Нейронна мережа може також наближати будь-яке відображення функції з будь-якого дискретного простору кінцевого виміру до іншого. Тоді як початкові теореми вперше були викладені в одиницях з функції активації, які насичують як для дуже негативних, так і для дуже позитивних аргументів, теореми універсального наближення також доведені для ширшого класу функцій активації, що включає в даний час часто використовувану випрямлену лінійну одиницю [25].

Універсальна теорема наближення означає, що незалежно від функції, яку ми намагаємося вивчити, ми знаємо, що великий MLP зможе представити цю функцію. Однак ми не гарантуємо, що алгоритм навчання зможе вивчити цю функцію. Навіть якщо MLP здатний представляти функцію, навчання може провалитися з двох різних причин. По-перше, алгоритм оптимізації, який використовується для навчання, не зможе знайти значення параметрів, яке відповідає бажаній функції. По-друге, алгоритм навчання може обрати неправильну функцію через оздоблення. Мережі прямого поширення забезпечують універсальну систему представлення функцій у тому сенсі, що, з урахуванням функції, існує мережа прямого поширення, яка наближає функцію. Немає універсальної процедури вивчення навчального набору конкретних прикладів та вибір функції, яка буде узагальнювати до точок, що не в навчальному наборі.

Універсальна теорема наближення говорить про те, що існує достатньо велика мережа, щоб досягти будь-якого ступеня точності, якої ми бажаємо, але теорема не каже, наскільки великою буде ця мережа. У [26] надається деякі межі розміру одношарової мережі, необхідної для наближення широкого класу функцій. На жаль, у гіршому випадку може знадобитися експоненціальна кількість прихованих одиниць (можливо з одним прихованим блоком, відповідним кожній вхідній конфігурації, яка повинна бути розрізненою). Це найпростіше побачити у двійковому випадку: кількість можливих бінарних функцій на векторах $\boldsymbol{v} \in \{0,1\}^n$ дорівнює 2^{2^n} і вибір однієї такої функції вимагає 2^n біт, що, як правило, вимагатиме $O\left(2^n\right)$ степенів свободи.

Підсумовуючи, мережі прямого поширення з одним шаром достатньо для представлення будь-якої функції, але шар може бути незрівнянно великим і може не засвоїти і правильно узагальнити. За багатьох обставин використання більш глибоких моделей може зменшити кількість одиниць, необхідних для представлення потрібної функції, і може зменшити значення кількості помилок узагальнення.

Існують сімейства функцій, які можна ефективно наблизити за допомогою архітектури з глибиною більшою, ніж деяка величина *d*, але для якої потрібна значно більша модель, якщо глибина обмежена меншою або дорівнює *d*. У багатьох випадках кількість прихованих одиниць, що вимагаються неглибокою моделлю, експоненціальна в *n*. Такі результати були вперше доведені для моделей, що не нагадують суцільні, диференційовані нейронні мережі, що використовуються для машинного навчання, але з тих пір були розширені до цих моделей. Перші результати були для схем логічних воріт [27]. Подальші роботи розширили ці результати на лінійні порогові одиниці з невід'ємними вагами [28], а потім до мереж з безперервною активацією [29]. Багато сучасних нейронних мереж використовують випрямлені лінійні одиниці. У [25] продемонстровано, що дрібні мережі з широким сімейством функцій активації, які не є поліномами, включаючи випрямлені лінійні одиниці, мають властивості універсального наближення, але ці результати не стосуються питань глибини та ефективності – вони визначають лише, що достатньо широка

мережа, що випрямляє, могла представляти будь-яку функцію. У [30] та [31] показано, що функції, представлені глибокою випрямовуючою сіткою може вимагати експоненціальної кількості прихованих одиниць з неглибокою (один прихований шар) мережою. Точніше, було показано, що шматково-лінійна мережа (яку можна отримати від випрямних нелінійностей або максимумів) може представляти функції з низкою регіонів, які збільшуються експоненціально чим далі у глибину мережі.

Основна теорема у [31] зазначає, що кількість лінійних областей, вирізаних глибокою випрямною мережею з d входами, глибиною l, і n одиниць на прихований шар, становить

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right) \tag{2.39}$$

тобто експоненціальна глибина l. У випадку максимуму мереж з k фільтрів на одиницю, кількість лінійних областей дорівнює

$$O\left(k^{(l-1)+d}\right) \tag{2.40}$$

Звичайно, немає гарантій того, що функції, які ми хочемо, щоб програми машинного навчання вивчили, (і зокрема для AI) діляться такою властивістю.

Ми можемо також захотіти вибрати глибоку модель з статистичних причин. У будь-який час, коли ми вибираємо певний алгоритм машинного навчання, ми неявно констатуємо набір попередніх переконань, які ми маємо про те, яку функцію має алгоритм навчання вивчити. Вибір глибокої моделі впроваджує дуже загальну думку про те, що функція, яку ми хочемо вивчити, повинна включати декілька простіших функцій. Це може інтерпретуватися з точки зору навчання представлення як таке, що ми віримо проблема навчання складається з виявлення набору основних факторів варіації що, в свою чергу, можна описати з точки зору інших, більш простих основних

факторів варіація. По черзі ми можемо трактувати використання глибокої архітектури як виражаючу переконання, що функція, яку ми хочемо вивчити – це комп'ютерна програма, що складається з кількох кроків, де кожен крок використовує вихід попереднього кроку. Ці проміжні результати не обов'язково є чинниками зміни, але можуть бути натомість аналогічні лічильникам або покажчикам, які мережа використовує для організації свого внутрішнього оброблення. Емпірично, більша глибина, схоже, призводить до кращого узагальнення для найрізноманітніших завдань [32].

2.2.2.2 Інші архітектурні міркування

Поки ми описали нейронні мережі як прості ланцюги шарів, з основними поняттями про глибину мережі та ширині кожного шару. На практиці нейронні мережі демонструють значно більшу різноманітність.

Для створення конкретних завдань було розроблено багато архітектур нейронної мережі. Спеціалізовані архітектури для комп'ютерного зору називаються згортковими мережами. Мережі прямого поширення також можуть бути узагальнені до рекурентних нейронних мереж для обробки послідовностей, які мають свої архітектурні поняття.

Взагалі шари не повинні бути з'єднані ланцюгом, хоча це і є найпоширеніша практика. Багато архітектур будують основний ланцюг, але потім додають додаткові архітектурні особливості до нього, такі як пропуск з'єднань, що йдуть від шару i до шару $i + 2$ або вище. Ці пропуски з'єднань полегшують рух градієнта з вихідних шарів до шарів ближче до вхідних.

Емпіричні результати показують, що більш глибокі мережі краще узагальнюються, якщо вони використовуються для розпізнавання багатоцифрових чисел з фотографій адрес. Точність тестового набору постійно зростає із збільшенням глибини. Це можна побачити на рисунку 2.7.

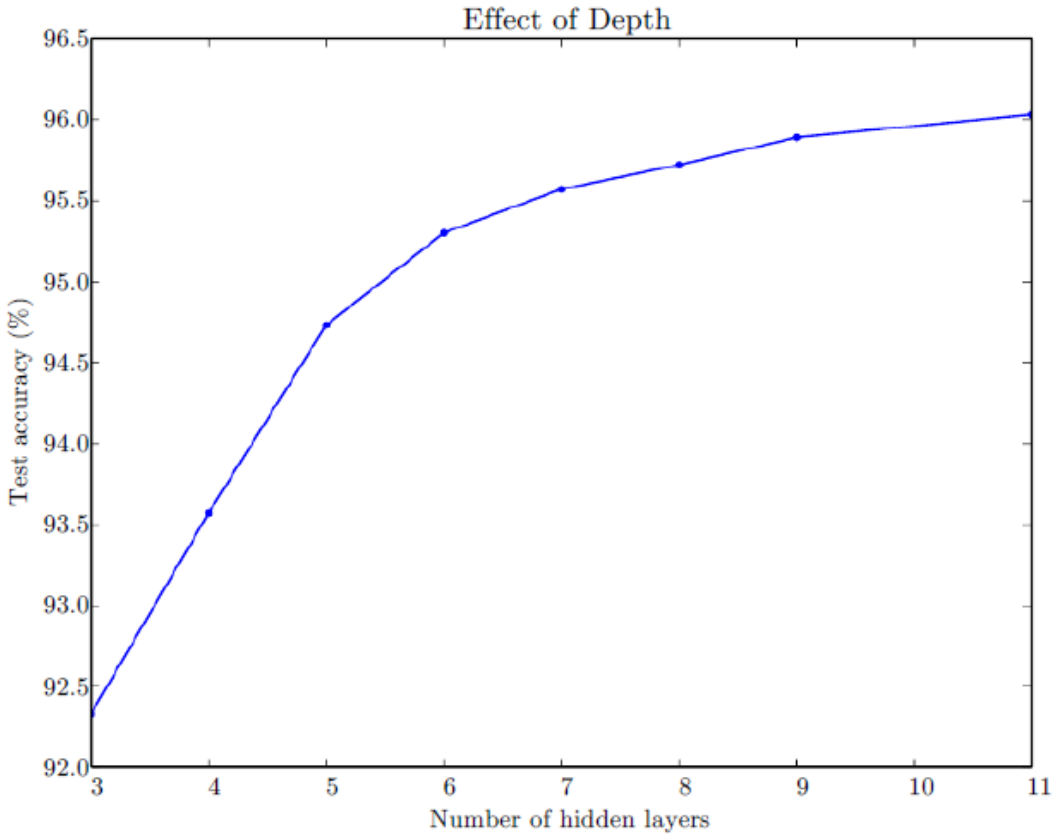


Рисунок 2.7 – Графік співвідношення точності тестів від кількості прихованих шарів

Більш глибокі моделі, як правило, працюють краще. Це не просто тому, що модель є більшою. Рисунок 2.8 показує, що збільшення кількості параметрів у шарах згорткових мереж без збільшення їх глибини не поліпшує їх роботу на тестах. Можна побачити, що неглибокі моделі вже перейшли у перенавчання при приблизно 20 мільйонах параметрів, тоді як глибокі можуть покращуватись від наявності понад 60 мільйонів. Це говорить про те, що використання глибокої моделі виражає корисну перевагу над простором функцій, які модель може засвоїти. Також виражається переконання, що функція повинна складатися з багатьох більш простих функцій, складених разом. Це може призвести або в навчанні представлення, яке складається в свою чергу з більш простих представлень (наприклад, кути, визначені по краях) або в навчанні програми з послідовно залежних кроків.

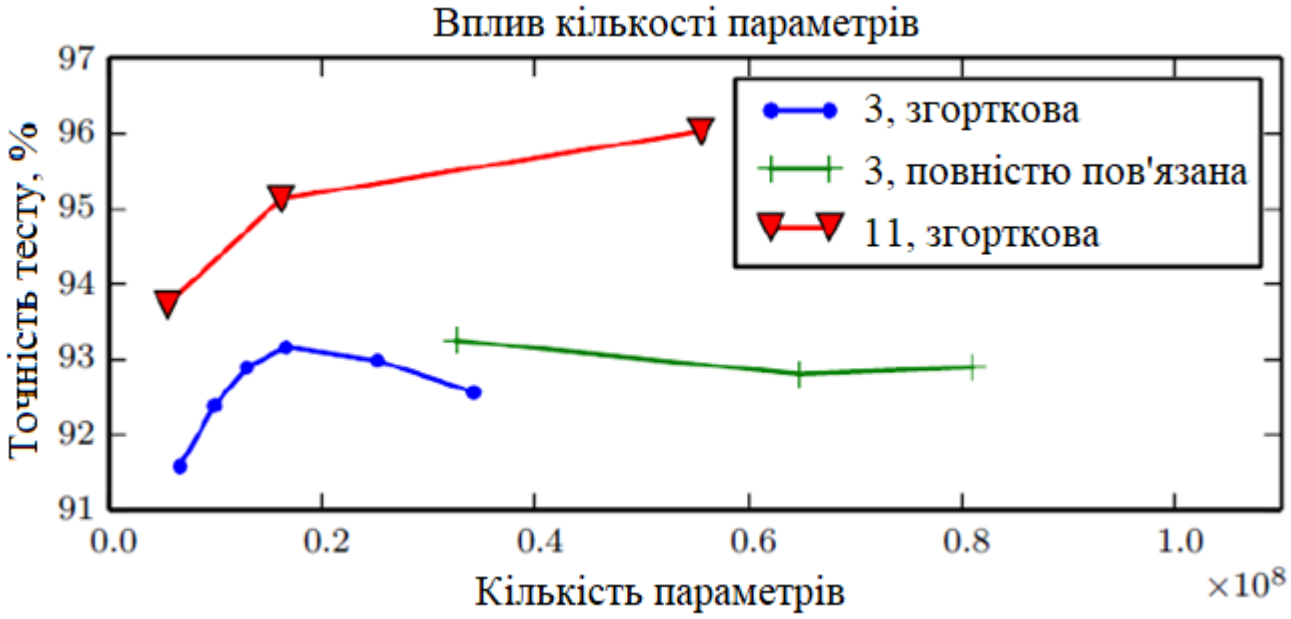


Рисунок 2.8 – Вплив кількості параметрів на продуктивність різних нейромереж

Ще одним ключовим моментом дизайну архітектури є саме те, як підключити пари шарів один до одного. У звичайному шарі нейронної мережі, описаному лінійним перетворенням через матрицю $\boldsymbol{W}$, кожен вхідний блок підключений до кожної одиниці виходу. Багато спеціалізованих мереж мають менше зв’язків , так що кожна одиниця у вхідному шарі з'єднана лише з невеликою підмножиною одиниць у вихідному шарі. Ці стратегії зменшення кількості з'єднань зменшують кількість параметрів і кількість обчислень, необхідних для оцінки мережі, але часто дуже залежать від проблем. Наприклад, згорткові мережі використовують спеціалізовані шаблони розріджених з'єднань які дуже ефективні для завдань, пов’язаних із комп’ютерним зором.

2.2.3 Алгоритми зворотного поширення та інші диференціації

Коли ми використовуємо нейронну мережу прямого поширення, щоб прийняти вхід $\boldsymbol{x}$ та створити вихід $\hat{y}$, інформація протікає вперед по мережі. Входи $\boldsymbol{x}$ забезпечують початкову інформація, яка потім поширюється до прихованих одиниць на кожному шарі і, нарешті, виробляє $\hat{y}$. Це називається прямим поширенням. Під

час навчання пряме поширення може продовжуватися вперед, поки воно не призведе до скалярної вартості $J(\boldsymbol{\theta})$. Алгоритм зворотного поширення [33] дозволяє інформації від витрат потім перетікати назад по мережі, щоб обчислити градієнт.

Вираз зворотне поширення часто неправильно розуміється як значення цілого алгоритму навчання багатошарових нейронних мереж. Насправді, зворотне поширення стосується лише методу обчислення градієнта, тоді як інший алгоритм, наприклад, стохастичний градієнтний спуск, використовується для виконання навчання за допомогою цього градієнта. Крім того, поширення зворотного зв'язку часто неправильно розуміється як специфічне для багатошарових нейронних мереж, але в принципі воно може обчислювати похідні будь-якої функції (для деяких функцій правильною відповіддю є повідомлення про те, що похідна від функції не визначена). У алгоритмах навчання нам найбільше часто потрібен градієнт функції витрат щодо параметрів, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Багато завдань машинного навчання включають також обчислення інших похідних як частина навчального процесу або для аналізу вивченої моделі. Алгоритм зворотного поширення може бути застосований і до цих завдань і не обмежується до обчислення градієнта функції витрат відносно параметрів. Ідея обчислення похідних шляхом поширення інформації через мережу є дуже загальною і може використовуватися для обчислення значень, таких як якобіан функції f з декількома виходами.

2.2.4 Обчислювальні графи

Щоб більш точно описати алгоритм зворотного поширення, корисно мати точнішу мову обчислювальних графів.

Існує багато способів формалізації обчислень як графів.

Тут ми використовуємо кожен вузол у графіку для позначення змінної. Змінна може бути скалярною, векторною, матричною, тензорною або навіть змінною іншого типу.

Для формалізації наших графів нам також потрібно ввести ідею операції. Операція – це проста функція однієї або декількох змінних. Наша мова графів супроводжується набором допустимих операцій. Функції складніші ніж операції в цьому наборі можуть бути описані складанням багатьох операцій разом. Не втрачаючи загальності, ми визначаємо операцію з повернення лише одиничної вихідної змінної.

Це не втрачає загальності, тому що вихідна змінна може мати кілька записів, таких як вектор. Програмна реалізація зворотного поширення зазвичай підтримує операції з декількома виходами, але ми уникаємо цього випадку у описі, оскільки вона містить багато додаткових деталей, які не важливі для концептуального розуміння.

Якщо змінна *y* обчислюється шляхом застосування операції до змінної *x*, то проведемо спрямований край від *x* до *y*.

Приклади обчислювальних графів показані на рисунку 2.9.

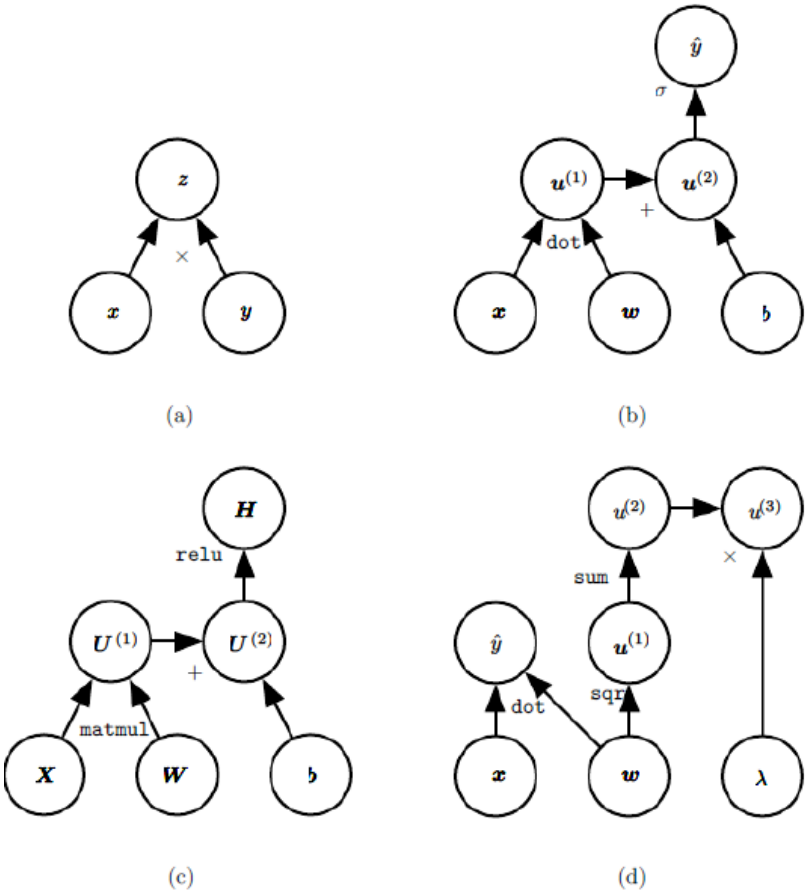


Рисунок 2.9 – Приклади обчислювальних графів

Пояснення до рисунку 2.9: a) граф за допомогою операції × обчислює вираз $z = xy$; (b) граф прогнозування логістичної регресії $\hat{y} = \sigma\left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{w} + b\right)$. Деякі з проміжних виразів не мають імен в алгебраїчному виразі але вони потрібні для графу. Ми просто називаємо *i*-ту змінну $\boldsymbol{u}^{(i)}$; (c) обчислювальний граф для виразу $\boldsymbol{H} = \max\{0, \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}\}$, який обчислює конструкцію матриці випрямлених активацій лінійних одиниць $\boldsymbol{H}$ задана матриця, що містить міні-партію входів $\boldsymbol{X}$; (d) Приклади a-c застосовуються щонайменше до однієї операції до кожної змінної, але можливо застосувати більше однієї операції. Тут показаний граф обчислення, який застосовує більше однієї операції до ваг $\boldsymbol{w}$ лінійної регресійної моделі. Ваги використовуються для прогнозування $\hat{y}$ та штрафу ваги зниження $\lambda \sum_i w_i^2$.

3 РОЗРОБЛЕННЯ СТРУКТУРИ СИСТЕМИ

3.1 Розроблення структурної схеми системи

Схема електрична структурна роботизованого комплексу представлена у додатку А.

Структурна схема розподілена на 3 основні частини: підсистема робота, АРМ-оператора та сервер.

Ідея полягає у тому, що сам робот складається тільки з датчиків та виконуючих механізмів, а сам процес обробки інформації відбувається на сервері.

Завдяки модулю GPS робот орієнтується на місцевості. Робот не може виїхати за границі виділеної для нього області.

Завдяки модулям WI-FI/3G/4G роботом можна керувати за допомогою АРМ-оператора.

Сенсорна панель встановлена для безпосереднього керування роботом, наприклад, при виникненні проблем із передачею даних через мережу Інтернет.

Існують 3 датчики, що вимірюють температуру, тиск та рівень рідини у баку. Інформація з них потрапляє до мікроконтролера. Якщо температура у баку досягає критичної відмітки, то робот припиняє роботу та їде у спеціально призначену будівлю. Аналогічно якщо рівень рідини у баку малий або тиск перевищую задану норму. При цьому у клієнт додатку надходить сповіщення про певну проблему.

Завдяки відеокамері робот може аналізувати та розпізнавати некультурні рослини, що заважають росту культур. Робот відправляє отримані дані з відеокамери на сервер додатку, де відбувається аналіз рослин: чи є культурою чи ні. Якщо так, на робот відправляється логічне повідомлення «так» і завдяки встановленим маніпуляторам обробляє рослину гербіцидами.

3.2 Розроблення функціональної схеми підсистеми робота

Схема електрична функціональна підсистеми робота роботизованого комплексу представлена у додатку Б.

На структурній схемі позначені пристрої вимірювання, виконавчі пристрої, елементи з'єднання та взаємодії з навколишнім середовищем та їх взаємозв'язок.

ДТр – датчик температури гербіциду

ДТ – датчик тиску гербіциду

ДР – датчик рівня гербіциду

П1…П3 – підсилювачі

АЦП1…АЦП3 – аналого-цифрові перетворювачі

DD1 – модуль Wi-Fi

DD2 – модуль 3G/4G

DD3 – модуль GPS

DD4 – відеокамера

DD5 – сенсорна панель керування роботом

М – двигун постійного струму

Р1…Р3 – реле

Н – насос

К1, К2 – клапани

ПМ1… ПМ6 – приводи маніпуляторів

ПР1…ПР4 – приводи руху

DA1…DA10 – драйвери

Для передачі даних з датчиків ДТр, ДТ та ДР використовуються аналого-цифрові перетворювачі, з виходів яких через інтерфейс RS-232 дані передаються на основний мікроконтролер.

Цифрові модулі, такі як 3G/4G, Wi-Fi та GPS підключені через інтерфейс I^2C

Відеокамера високої роздільної здатності використовує USB, оскільки передається великий обсяг даних.

Серед виконавчих пристроїв наявні приводи маніпуляторів ПМ1…ПМ6 (по три на кожний), приводу рушійних двигунів коліс ПР1…ПР4, клапани подачі гербіциду К1 та К2 в кожному маніпуляторі та насос Н для створення тиску в резервуарі з гербіцидом.

Для керування рушійної сили двигунів коліс та руху маніпуляторів використовуються драйвери DA1…DA10. Клапани маніпуляторів відкриваються та закриваються за допомогою реле Р1 та Р2. Також за допомогою реле Р3 вмикається та вимикається насос.

3.3 Розроблення алгоритму роботи підсистеми робота

Схема алгоритму роботи підсистеми робота представлена у додатку В.

Алгоритм роботи робота умовно ділиться на три частини: алгоритм запуску, алгоритм основного циклу та алгоритм завершення роботи.

3.3.1 Алгоритм запуску

Алгоритм запуску описує процес запуску системи. Після подачі електроенергії з акумулятора виконується перевірка на кількість заряду. Якщо заряду нема, то виконуються вивід на сенсорну панель та на АРМ оператора повідомлення про нестачу заряду і вимкнення робота. У іншому випадку виконується перевірка працездатності давачів. Якщо давачі у неробочому стані, виконується вивід на сенсорну панель та на АРМ оператора повідомлення про непрацездатність давачів, а робот залишається у режимі очікування. У іншому випадку вимірюється рівень рідини у баку. Якщо бак порожній, виконується вивід на сенсорну панель та на АРМ оператора повідомлення про пустий бак, а робот залишається у режимі очікування. У іншому випадку виконується увімкнення камери: якщо камера непрацездатна,

виконується вивід на сенсорну панель та на АРМ оператора повідомлення про непрацездатну камеру. У іншому випадку запускається алгоритм основного циклу.

Блок 1: Початок роботи алгоритму. Перехід до блоку 2.

Блок 2: Подача електроенергії з акумулятора. Перехід до блоку 3.

Блок 3: Визначення кількості заряду. Перехід до блоку 4.

Блок 4: Перехід до блоку 5 у випадку нестачі заряду для роботи робота. Перехід до блоку 6 якщо заряду достатньо.

Блок 5: Надсилання повідомлення про нестачу енергії на АРМ оператора. Перехід до блоку 16

Блок 6: Збір інформації з давачів на предмет їх працездатності. Перехід до блоку 7.

Блок 7: Перехід до блоку 9 якщо давачі у робочому стані. Перехід до блоку 8 у випадку їх непрацездатності.

Блок 8: Надсилання повідомлення про непрацездатність давачів на АРМ оператора. Перехід до блоку 16.

Блок 9: Вимірювання рівня рідини у баку. Перехід до блоку 10.

Блок 10: Перехід до блоку 11 якщо бак пустий. Перехід до блоку 12 якщо бак не пустий.

Блок 11: Надсилання повідомлення про пустий бак на АРМ оператора. Перехід до блоку 16.

Блок 12: Увімкнення камери. Перехід до блоку 13.

Блок 13: Перехід до блоку 14 у випадку непрацездатності камери. Перехід до блоку 15 у випадку працездатності.

Блок 14: Надсилання повідомлення про непрацездатну камеру на АРМ оператора. Перехід до блоку 16.

Блок 15: Запуск алгоритму основного циклу. Завершення алгоритму.

Блок 16: Завершення роботи алгоритму.

3.3.2 Алгоритм основного циклу

Алгоритм основного циклу описує роботу робота під час виконування його основних функцій. Користувач повинен ввести параметри роботи робота, тобто яку площу оброблювати і які саме рослини. Після цього робот починає свій рух. З цього моменту починається виконання паралельних дій. Робот має одночасно перевіряти своє місцезнаходження (чи не виїхав за межі заданої області) і відправляти дані з камери на сервер щоб потім оброблювати рослини. Якщо бак порожній то виконується надсилання повідомлення на АРМ оператора про порожній бак і запускається алгоритм завершення роботи.

Блок 15.1: Початок роботи алгоритму. Перехід до блоку 11.2.

Блок 15.2: Користувач задає параметри роботи робота. Перехід до блоку 11.3.

Блок 15.3: Увімкнення двигунів руху. Розпаралелювання процесів.

Блок 15.4: Зчитування даних с GPS. Перехід до блоку 15.5.

Блок 15.5: Перехід до блоку 15.6 при досягненні робота границі заданої області руху. У іншому випадку перехід до блоку 15.4.

Блок 15.6: Управління двигунами руху для виконання повороту.

Блок 15.7: Зчитування даних з камери. Перехід до блоку 15.8.

Блок 15.8: Відправка даних на сервер. Перехід до блоку 15.9.

Блок 15.9: Перехід до блоку 15.7 якщо отримана відповідь від серверу – «ні». У іншому випадку перехід до блоку 15.10.

Блок 15.10: Обробка рослини за допомогою маніпуляторів. Перехід до блоку 15.11.

Блок 15.11: Перехід до блоку 15.7 якщо бак не порожній. Перехід до блоку 15.12 якщо порожній

Блок 15.12: Надсилання повідомлення про порожній бак на АРМ оператора. Перехід до блоку 15.13.

Блок 15.13: Запуск алгоритму завершення роботи. Перехід до блоку 15.14.

Блок 15.14: Завершення роботи алгоритму.

3.3.3 Алгоритм завершення роботи

Алгоритм завершення роботи описує повернення робота «додому». Виконується автоматичне управління двигунами руху для його повернення «додому» при цьому йде зчитування даних з GPS. Якщо робот повернувся «додому», надсилається повідомлення на АРМ оператора про прибуття і робот завершує свою роботу.

Блок 15.13.1: Початок роботи алгоритму завершення роботи. Перехід до блоку 15.13.2.

Блок 15.13.2: Управління двигунами руху для повернення «додому». Перехід до блоку 15.13.3.

Блок 15.13.3: Зчитування даних з GPS. Перехід до блоку 15.13.4.

Блок 15.13.4: Перехід до блоку 15.13.2 якщо робот ще не «вдома». Перехід до блоку 15.13.5 якщо робот вже «вдома».

Блок 15.13.5: Надсилання повідомлення про прибуття додому та завершення роботи на АРМ оператора. Перехід до блоку 15.13.6.

Блок 15.13.6: Перекриття подачі електроенергії. Перехід до блоку 15.13.7.

Блок 15.13.7: Завершення роботи алгоритму.

4 МОДЕЛЮВАННЯ ПІДСИСТЕМИ РОЗПІЗНАВАННЯ РОСЛИН

4.1 Розроблення моделі підсистеми у MATLAB/Simulink

Для моделювання роботи системи розпізнавання рослин, будемо використовувати програмний пакет MATLAB/Simulink [34].

Побудована модель представлена на рисунку 4.1, а пояснення до блоків у таблиці 4.1.

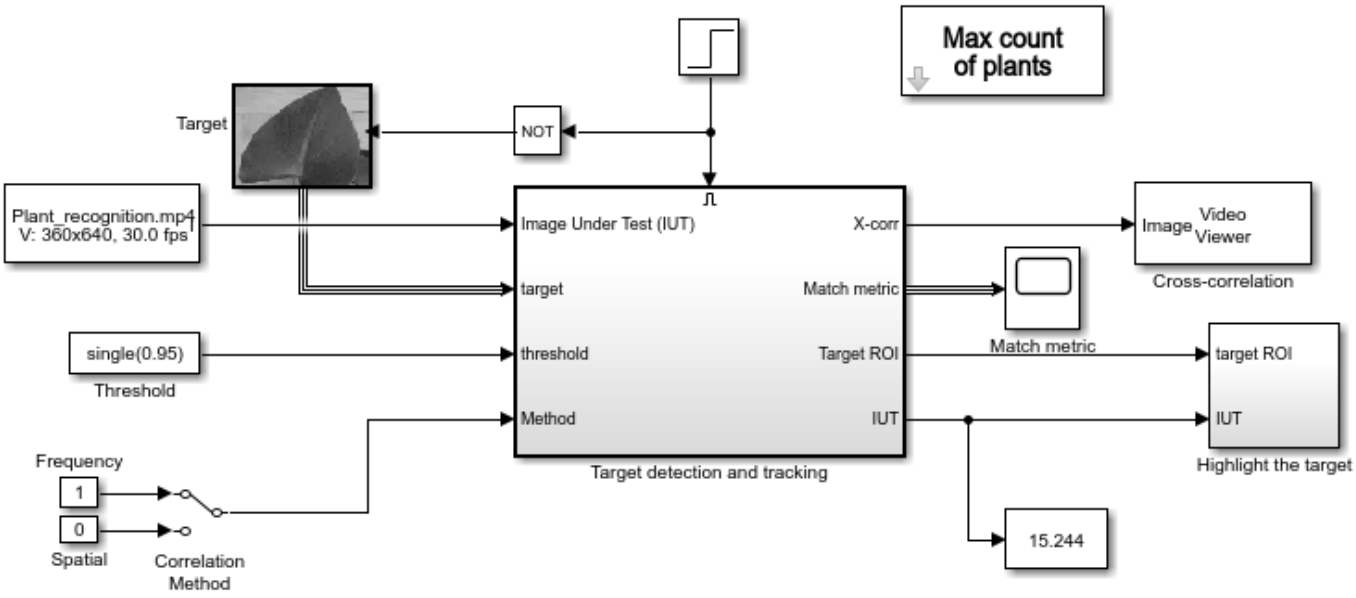
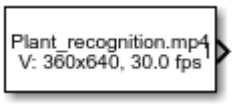
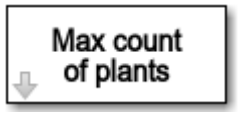


Рисунок 4.1 – Модель MATLAB/Simulink

Таблиця 4.1 – Пояснення до блоків

Блок	Назва блоку	Призначення
	From Multimedia File	використання у моделі мультимедійного файлу, що має у собі відеодоріжку, аудіодоріжку, або і те і інше
	Constant	введення у модель постійного значення
		масштабування вхідного відео, щоб перехресна кореляція

	Pyramiding factor	виконувалася на значно меншому відеокадрі; зміна кількості об'єктів одночасного розпізнавання
	Video viewer	Показ відео із виконаними змінами
	Scope	відображення залежності сигналу від часу
	Manual switch	ручна зміна вихідного сигналу

У даній моделі відео, на якому потрібно розпізнавати рослини, завантажується у блок From Multimedia File, і в подальшому буде згадуватись як Image Under Test. Блок Target, який є підсистемою, що зображена на рисунку 4.2, використовує перший кадр відео як шаблон, у якому позначений об'єкт який потрібно буде розпізнавати впродовж усього відео.

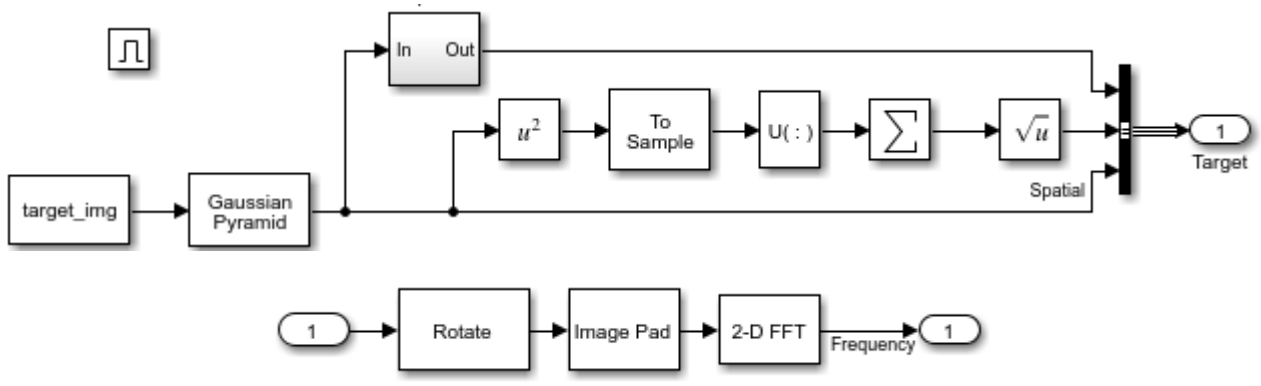
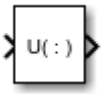
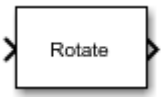


Рисунок 4.2 – Підсистема блоку Target

Таблиця 4.2 – Пояснення до блоків підсистеми Target

Блок	Назва блоку	Призначення
	Gaussian Pyramid	обчислення Гауссового розширення або скорочення піраміди, щоб змінити розмір зображення
	Math	математичний блок для використання логарифму, підношення до степеню, обчислення степеневого кореню і т.д.
	Frame conversion	встановлює режим вибірки вихідного сигналу
	Reshape	зміна виміру вектору або матриці вхідного сигналу
	Sum	додавання або віднімання вхідних сигналів
	Bus Creator	об’єднання вхідних сигналів у єдину шину
	Rotate	обертання зображення на вході на кут у радіанах
	Image Pad	обрамлення зображення
	2-D FFT	виводить комплексне швидке перетворення Фур'є у двох вимірах натурального або комплексного входу.

Блок під назвою Max Count of Plants використовується для позначення скільки листків рослин одночасно можуть бути розпізнанні за один кадр. Блок Constant під

назвою Threshold вказує наскільки точним має бути розпізнавання. У підрозділі 4.2 буде продемонстровано як саме це впливає на розпізнавання. Система блоків Correlation method складається з 2 константних блоків «частотний» та «просторовий» із значеннями 1 і 0 відповідно і ручної зміни використання констант для вибору методу кореляції. Блок Video Viewer під назвою Cross-correlation показує відео, оброблене за допомогою кросс-кореляції. Блок Match Metric показує графік того, наскільки добре працює виявлення об’єктів у вигляді різниці вхідного сигналу від порогу точності Threshold. Підсистема під назвою Highlight the target, схема якої представлена на рисунку 4.3, призначена для виділення виявлених об’єктів за допомогою прямокутників різних кольорів.

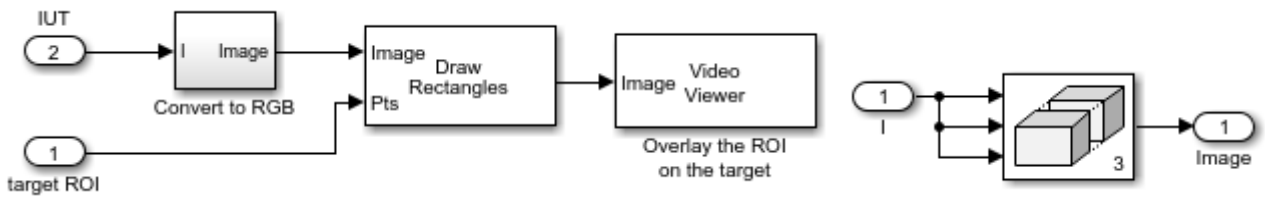


Рисунок 4.3 – Підсистема виділення об’єктів

За допомогою блоку Draw Shapes розпізнанні об’єкти (у подальшому так звані регіони інтересів (Regions of Interest, ROI)) будуть виділятись фігурами різних кольорів, які можна змінювати у вікні властивостей цього блоку (рисунок 4.4).

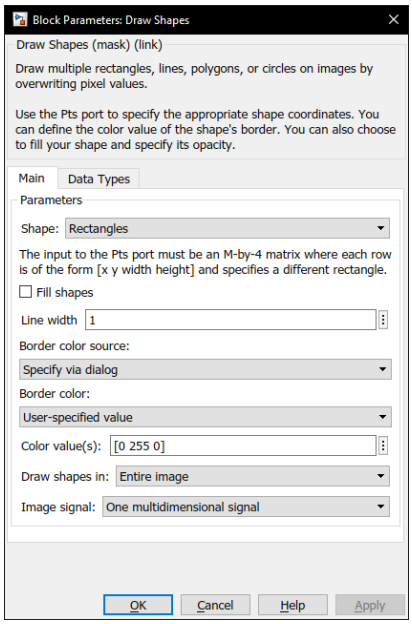


Рисунок 6.4 – Властивості блоку Draw Shapes

Блок Target detection and tracking являється підсистемою, схема якої зображена на рисунку 4.5.

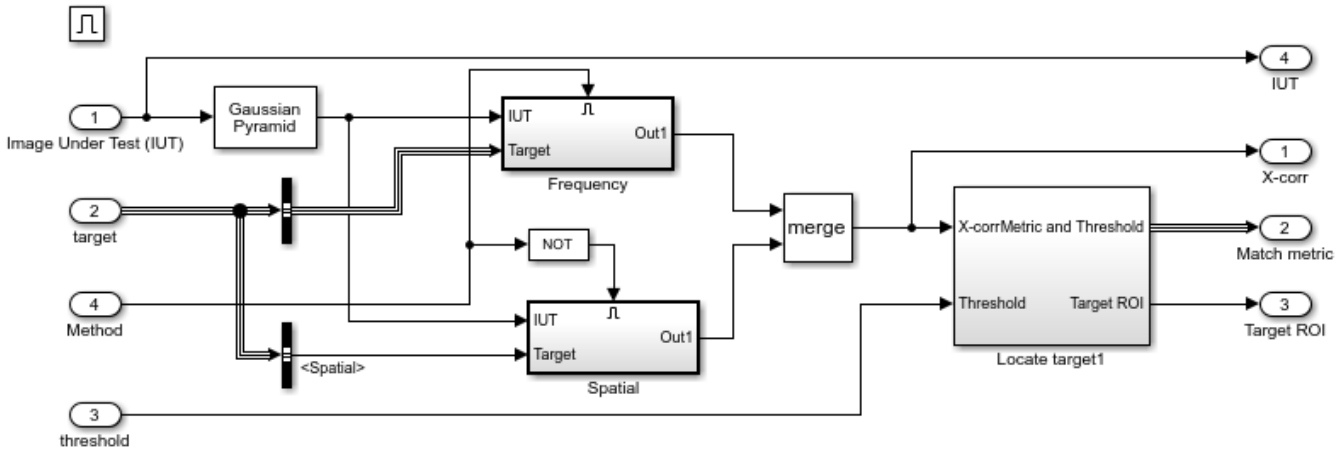


Рисунок 4.5 – Підсистема виявлення та відслідковування об’єктів

У цій підсистемі можна побачити, що тут робота моделі відгалужується при виборі методу кореляції. При виборі перехресної кореляції в частотній області, підсистема використовуватиме для подальших розрахунків блок Frequency, схема якого представлена на рисунках 4.6, 4.7 та 4.8, а інший – Spatial, схема якого представлена на рисунку 4.10 – відключатиме (це видно з 4го вхідного сигналу Method, який має блок NOT, який інвертує вхідний сигнал).

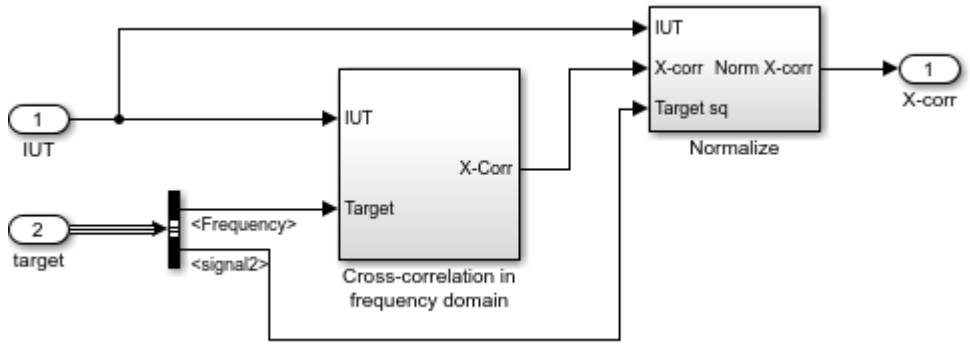


Рисунок 4.6 – Підсистема нормалізованої перехресної кореляції у частотній області

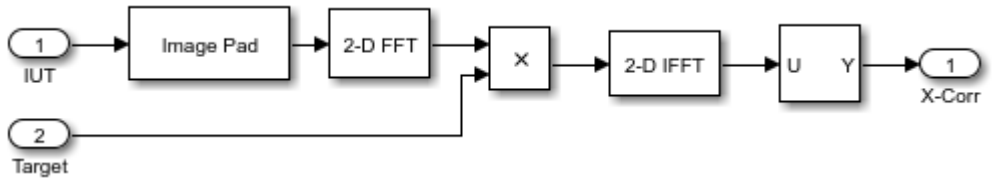


Рисунок 4.7 – Підсистема перехресної кореляції у частотній області

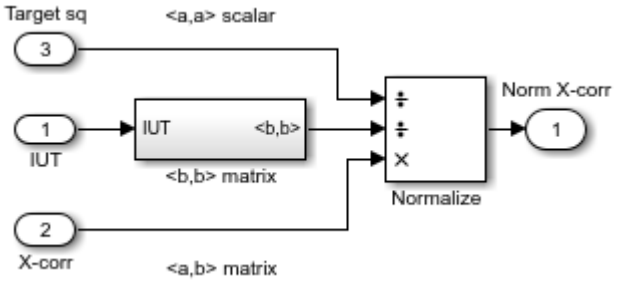


Рисунок 4.8 – Підсистема нормалізації перехресної кореляції у частотній області

Рисунок 4.9 – Підсистема перетворення вхідного зображення

Пояснення до рисунків 4.6, 4.7 та 4.8: кадр відео является одним із входів у підсистемі перехресної кореляції. Іншим входом являється попередньо оброблена ціль, яка видобувається з селекторного блоку, що розділяє шину із комбінованих сигналів. У підсистемі власне кореляції відбувається добуток комплексного швидкого перетворення Фур'є вхідного зображення і цілі, далі з цим добутком відбувається інверсивне комплексне швидке перетворення Фур'є та за допомогою селекторного блоку видобувається дійсна частина кореляції. Це і є виходом перехресної кореляції. Далі цей вихід вже є входом у підсистемі нормалізації, де моделюється нерівність Коші – Буняковського – Шварца

$$-1 \le \frac{\langle a,b \rangle}{\sqrt{\langle a,a \rangle \cdot \langle b,b \rangle}} \le 1 \qquad (4.1)$$

І результатом цієї нормалізації є нормалізована перехресна кореляція у частотній області.

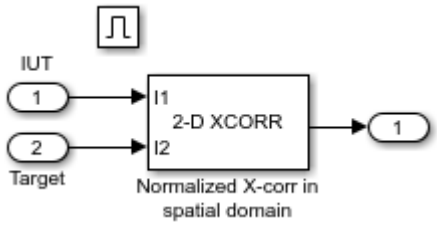


Рисунок 4.10 – Підсистема просторової перехресної кореляції

У даній підсистемі виконується звичайна двовимірна перехресна кореляція між двома входами – кадру відео та ціллю.

На рисунку 4.11 показана схема підсистеми знаходження областей інтересів. Вихід кореляції переходить у режим вибірки, переформовується у блоці Reshape та переходить у вхід до селектора, у якого другим входом є лінійно перетворений за допомогою функції

```
1 function y = fcn(u)
2 l = size(u);
3 y = u(:,l(2));
```

індекс локального максимуму матриці кореляції (рисунки 6.11, 6,12).

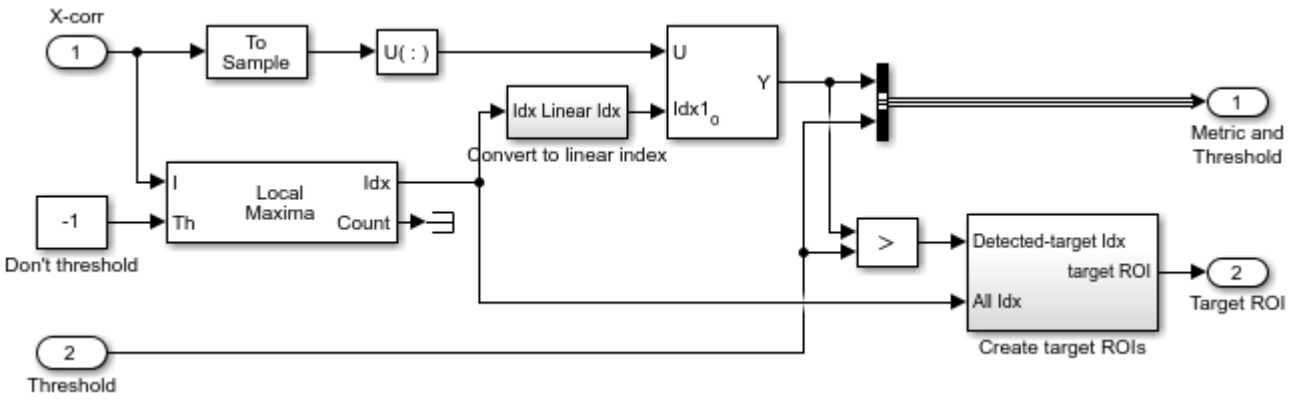


Рисунок 4.11 – Підсистема знаходження областей інтересів

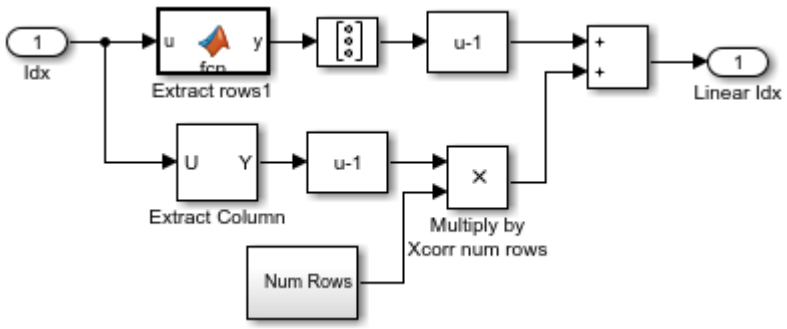


Рисунок 4.12 – Підсистема лінійного перетворення

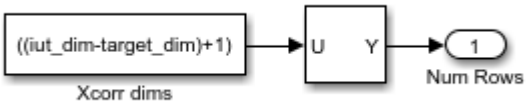


Рисунок 4.13 – Підсистема лінійного перетворення-2

На рисунку 4.14 показана підсистема під назвою Create Target ROIs. У даній підсистемі за допомогою функцій

```
1 function y = fcn(u, Idx)
2 y = u;
3 y(~Idx,:)=0;
```

відбувається процес створення областей інтересів.

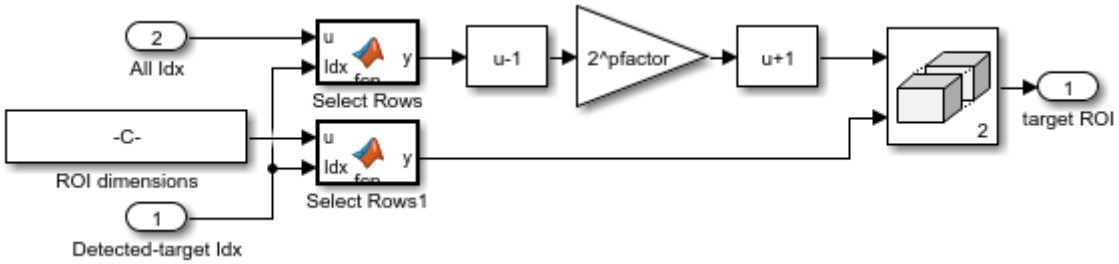


Рисунок 4.14 – Підсистема створення областей інтересів

4.2 Експериментальні дослідження роботи моделі

За допомогою даної моделі було проведено експериментальні дослідження на основі трьох відео файлів. За допомогою блоку From Multimedia File завантажуємо у

модель потрібні відеодоріжки, а за допомогою блоку Target виділяємо лист рослини, який модель повинна знайти на усьому відео.

4.2.1 Розпізнавання паростків Calla Palustris (Образки болотяні)

У цьому пункті буде продемонстровано розпізнавання Calla Palustris при різних методах кореляції, варіювання точності розпізнавання (Threshold) та кількості листків одночасного розпізнавання.

На рисунку 4.15 продемонстровано спосіб виділення листка, який потім буде використовуватись у якості шаблону для розпізнавання інших листків.

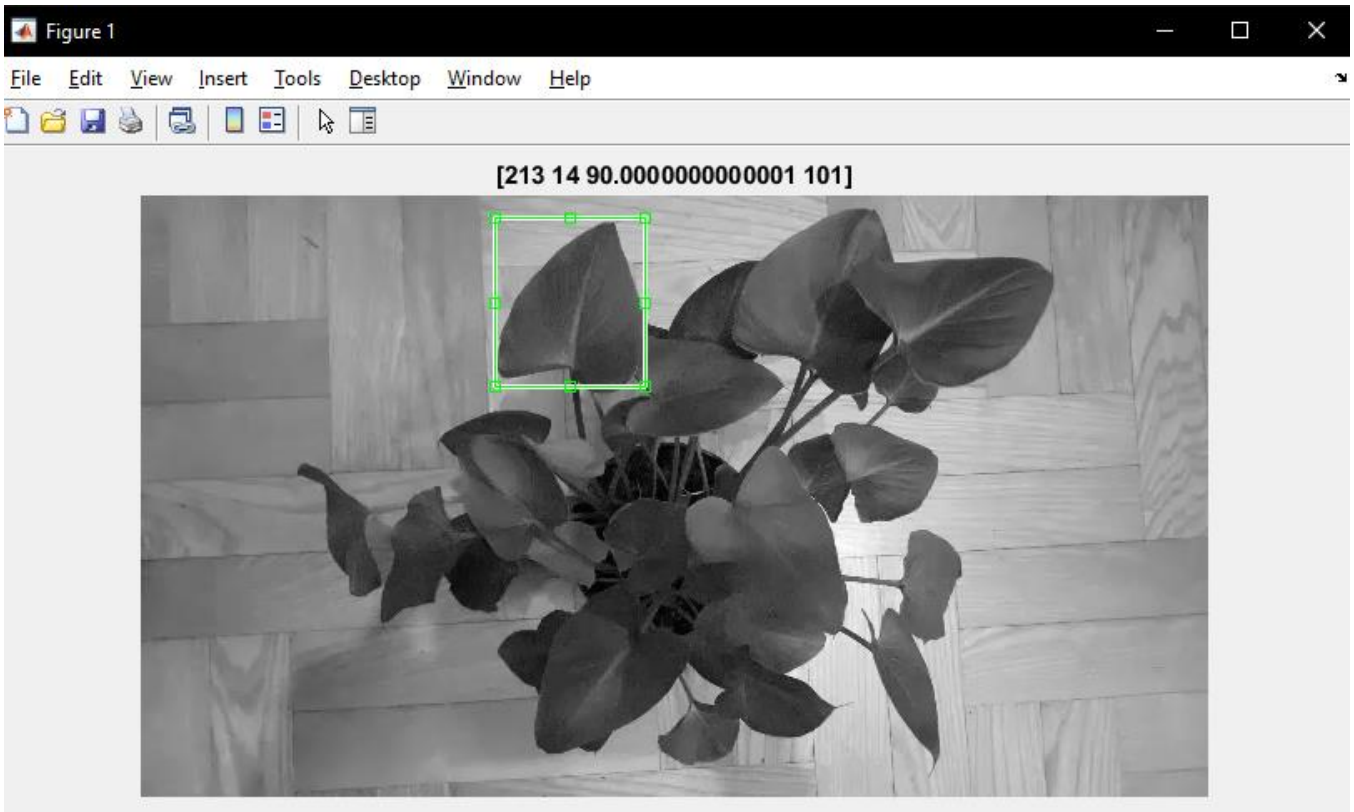


Рисунок 4.15 – Виділення листка Calla Palustris

На рисунку 4.16 показано, як при роботі моделі розпізнаються листки при таких параметрах:

- Точність: 0.96
- Кількість розпізнавальних об’єктів: 2
- Метод кореляції: у частотній області

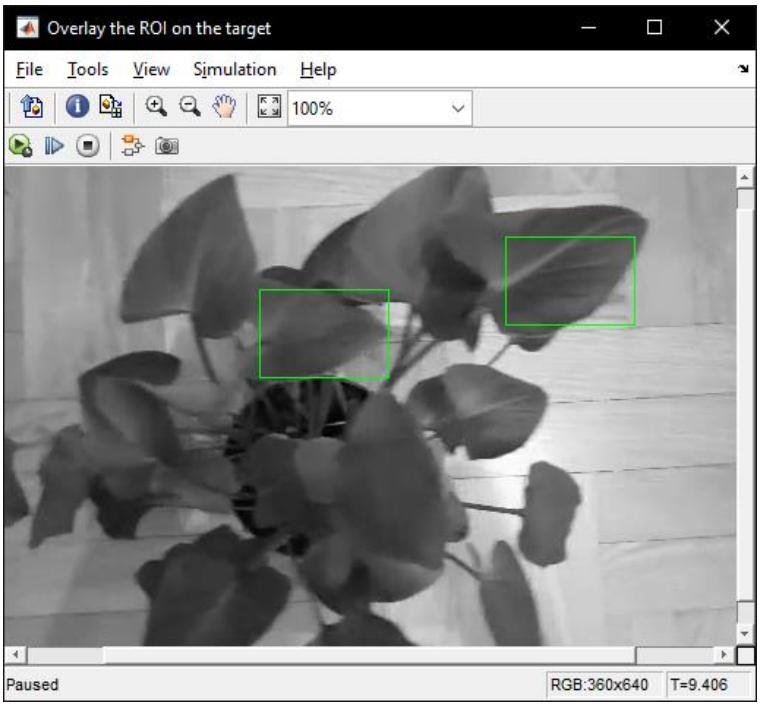


Рисунок 4.16 – Розпізнавання листків

На рисунку 4.17 показано, як виділяються листки на обробленому за допомогою кореляції відео: області інтересів мають яскраві кольори по відношенню до інших областей.

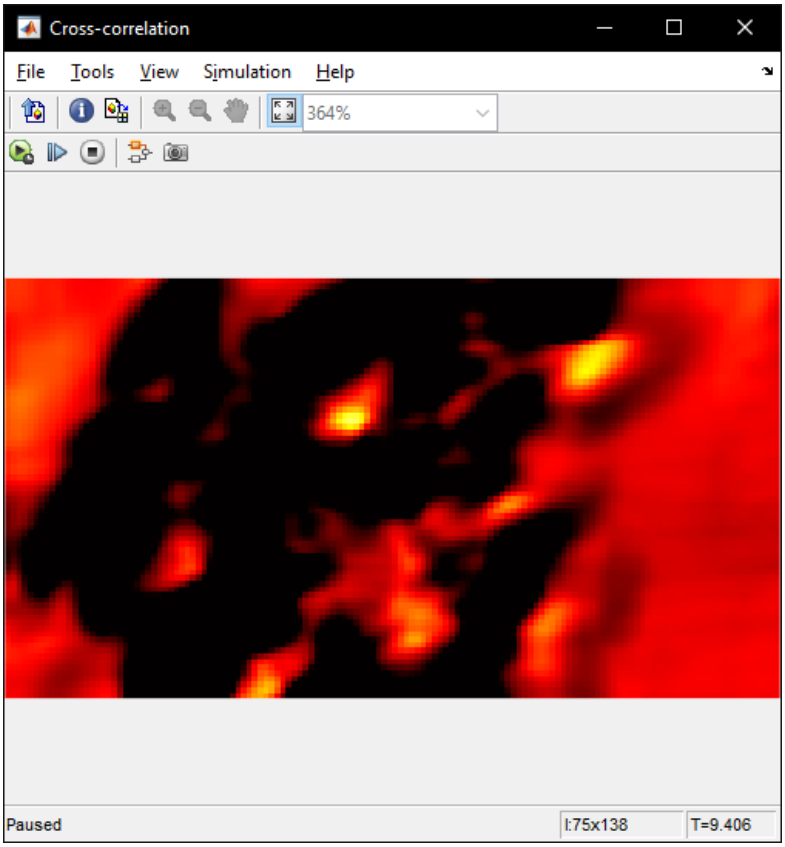


Рисунок 4.17 – Оброблене за допомогою перехресної кореляції зображення

На рисунку 4.18 показано графік співвідношення точності розпізнавання листків по відношенню до заданої точності – 0.96. На графіку видно один стабільний сигнал, що показує задану точність і два сигнали, що являють собою два одночасних розпізнавальних сигнали. Як видно з графіку, у момент зупинки симуляції, коли на рисунках 4.16 і 4.17 були розпізнанні 2 листки, обидва сигнали досягли значень більших ніж задана точність, що підтверджує правильність розпізнавання.

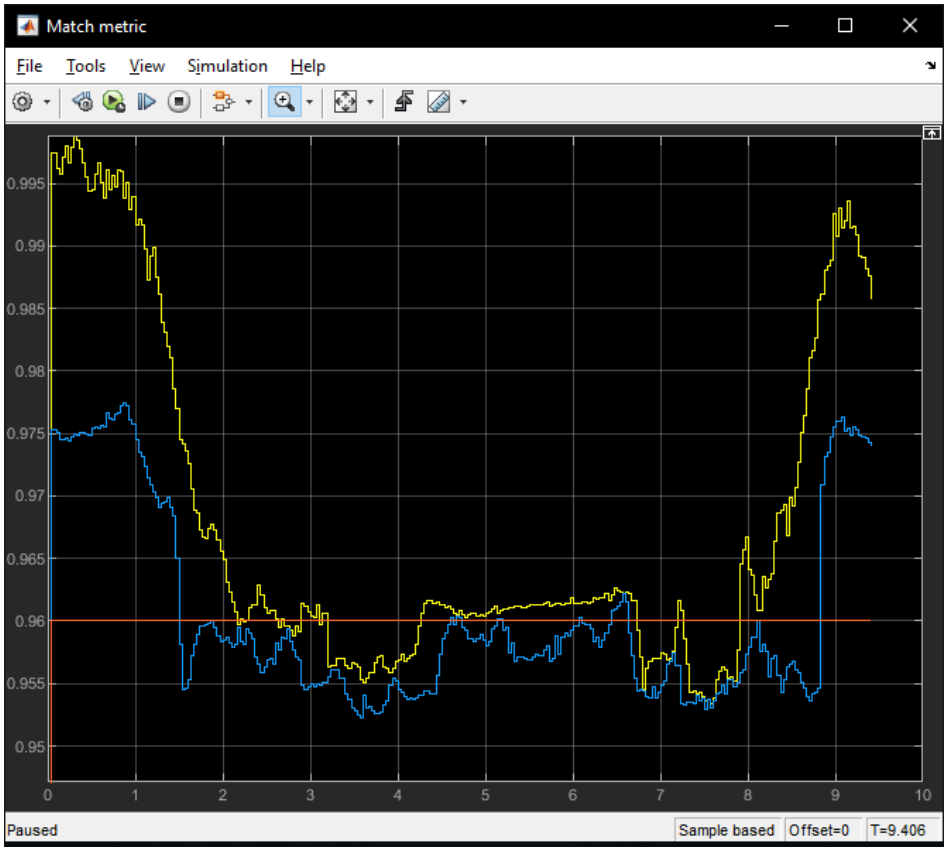


Рисунок 4.18 – Графік співвідношення точності розпізнавання

Тепер змінимо параметри системи на такі:

- Точність: 0.96
- Кількість розпізнавальних об’єктів: 6
- Метод кореляції: просторовий

Результати продемонстровано на рисунках 4.19, 4.20 та 4.21.

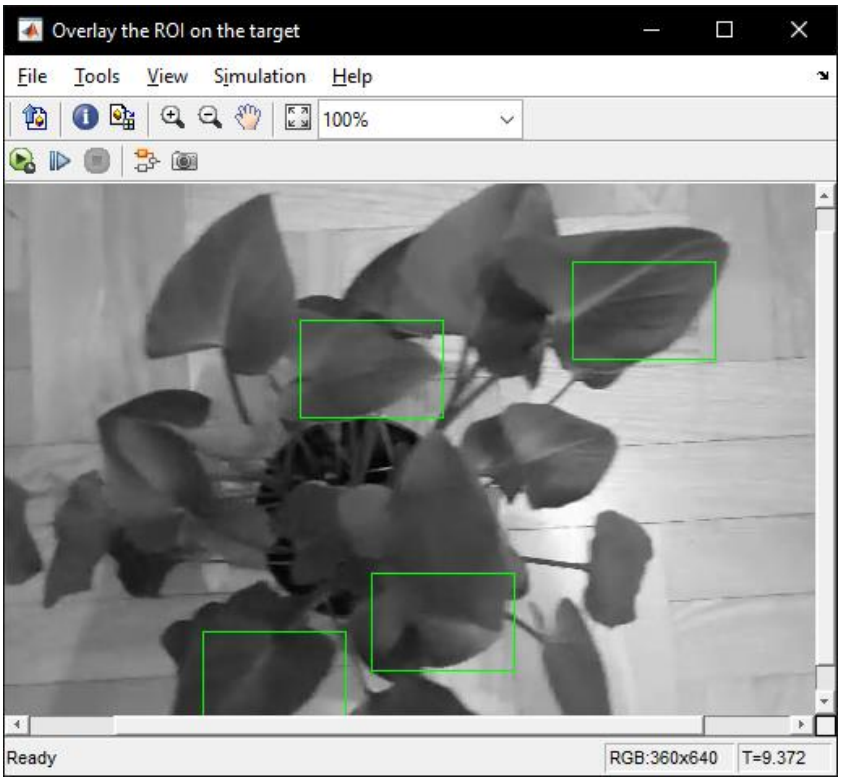


Рисунок 4.19 – Розпізнавання листків при змінених параметрах

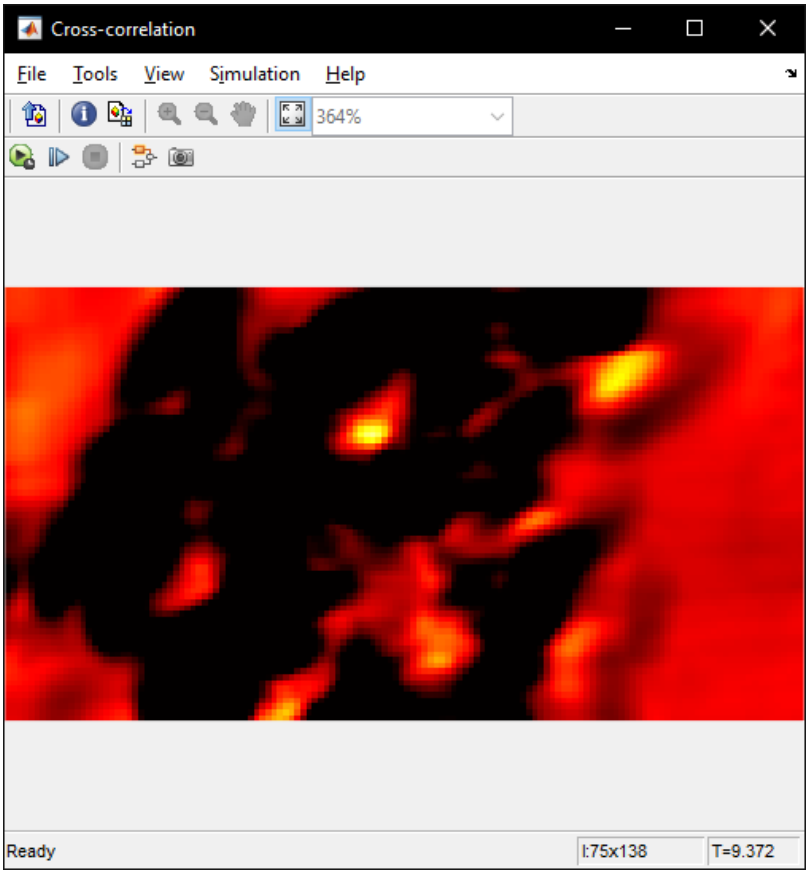


Рисунок 4.20 – Оброблене за допомогою перехресної кореляції зображення при змінених параметрах

Рисунок 4.21 – Графік співвідношення точності розпізнавання при змінених параметрах

Далі було змінено задану точність до 0.9. Результати показані на рисунках 4.22, 4.23 та 4.24.

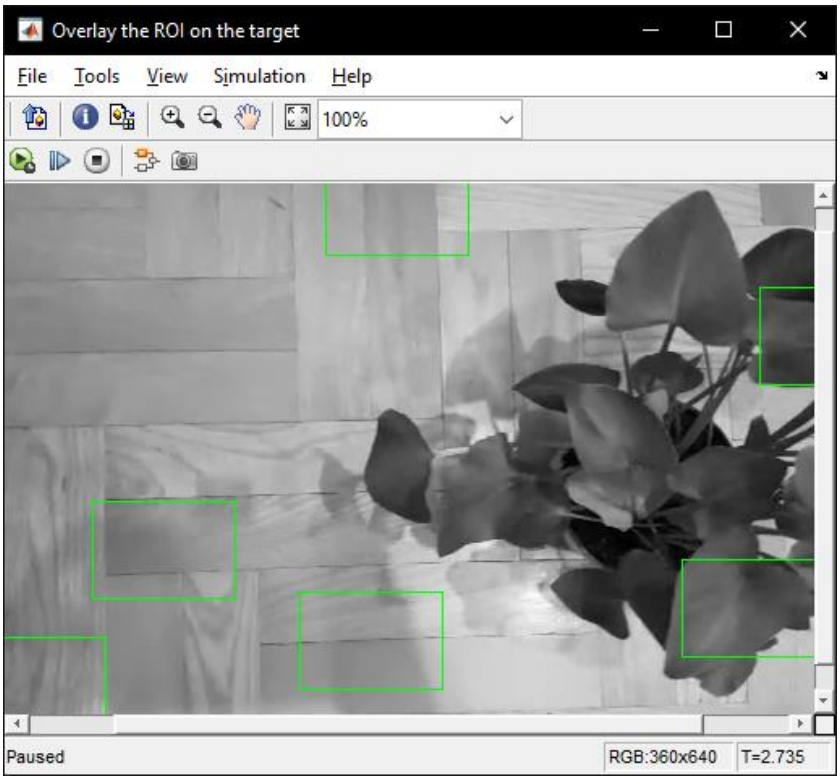


Рисунок 4.22 – Розпізнавання листків при заданій точності 0.9

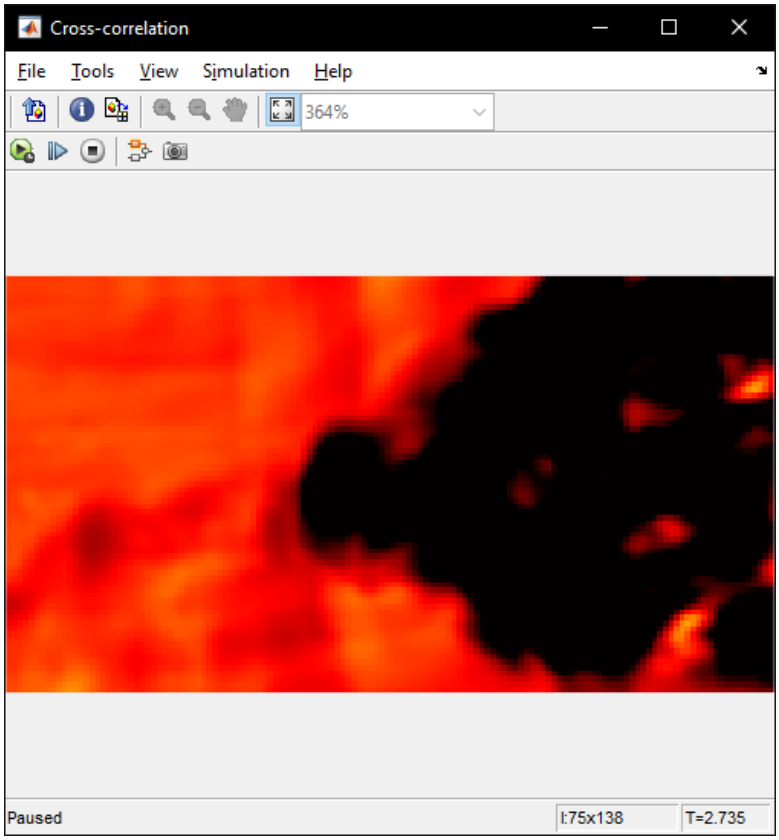


Рисунок 4.23 – Оброблене за допомогою перехресної кореляції зображення при заданій точності 0.9

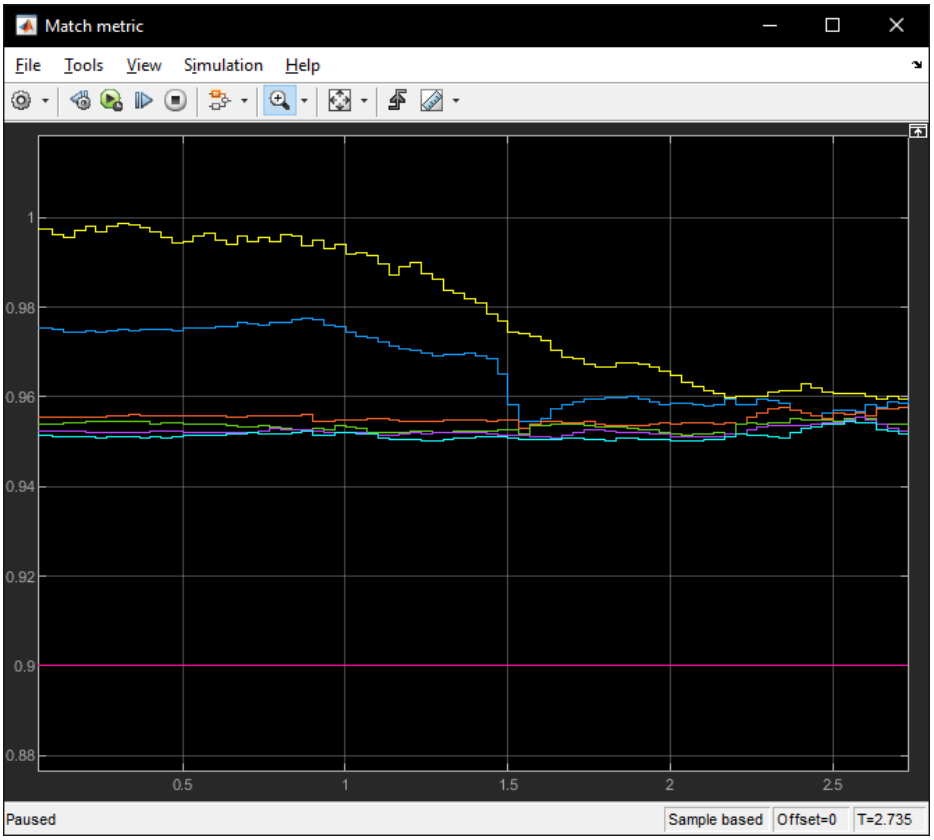


Рисунок 4.24 – Графік співвідношення точності розпізнавання при заданій точності 0.9

4.2.2 Розпізнавання паростків картоплі

Для розпізнавання паростків картоплі запозичимо відео паростків картоплі з YouTube [35]. Настройка параметрів буде ідентичною до пункту 4.2.1. Результати продемонстровано на рисунках 4.25, 4.26 та 4.27.

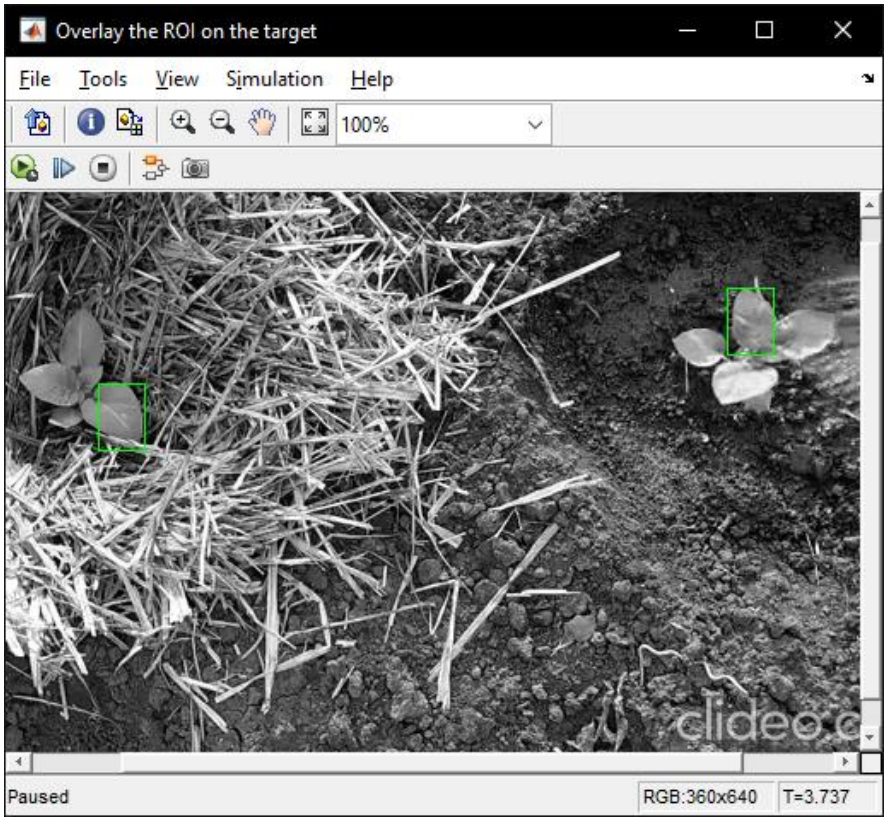


Рисунок 4.25 – Розпізнавання листків

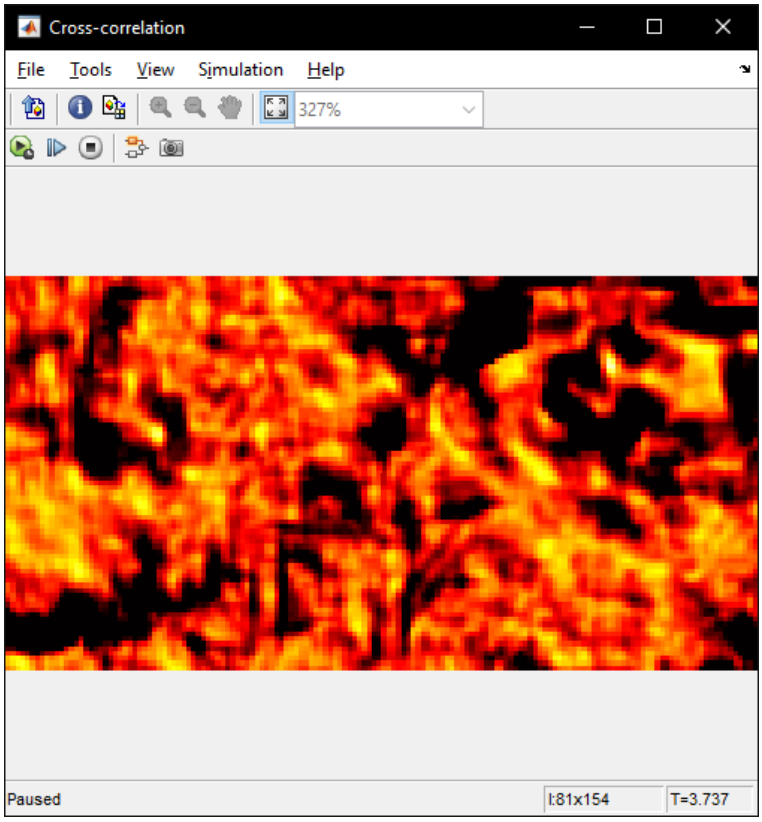


Рисунок 4.26 – Оброблене за допомогою перехресної кореляції зображення

Рисунок 4.27 – Графік співвідношення точності розпізнавання

4.2.3 Розпізнавання паростків буряку

Для розпізнавання паростків буряку запозичимо відео паростків картоплі з YouTube [36]. Через специфічність відеоряду, настройка параметрів буде такою:

- Точність: 0.95
- Кількість розпізнавальних об’єктів: 10
- Метод кореляції: у частотній області

Результати продемонстровано на рисунках 4.28, 4.29 та 4.30

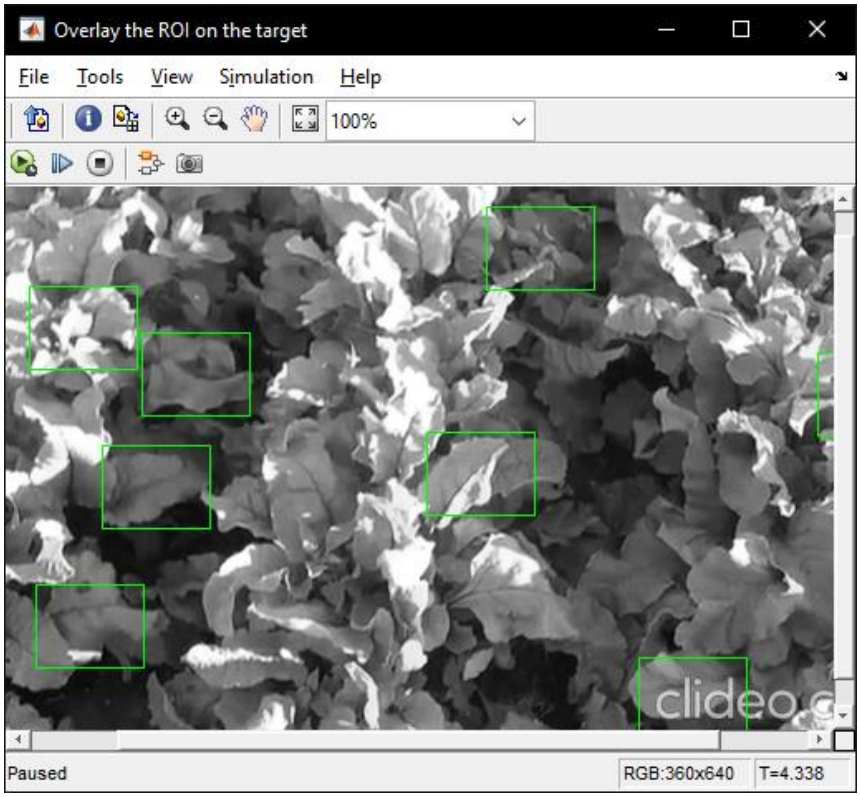


Рисунок 4.28 – Розпізнавання листків

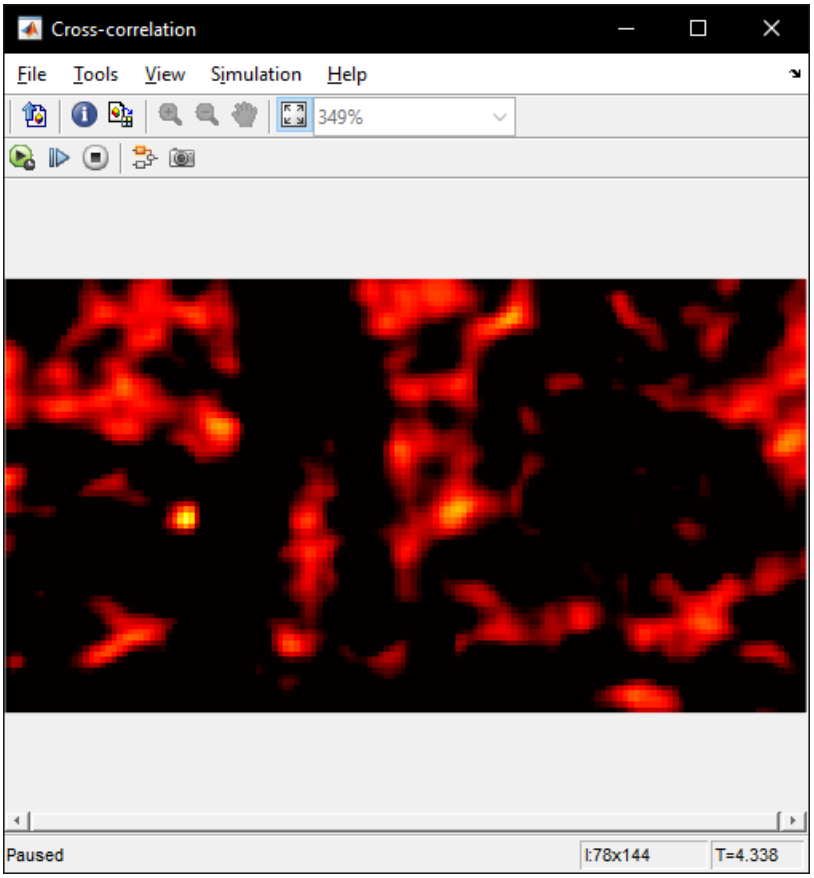


Рисунок 4.29 – Оброблене за допомогою перехресної кореляції зображення

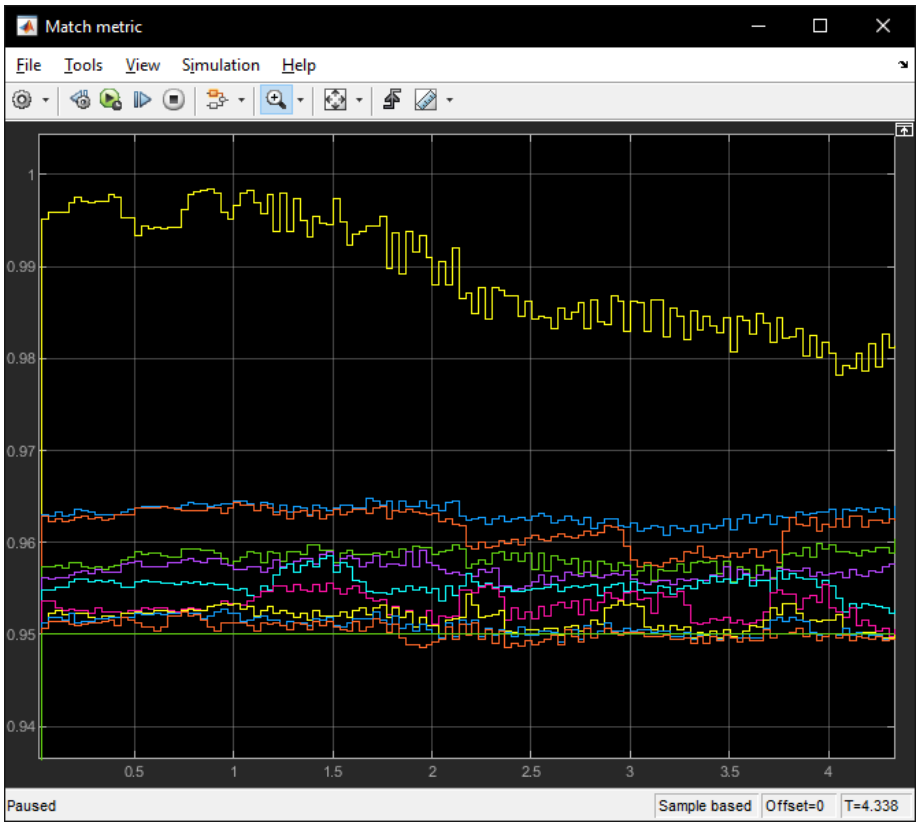


Рисунок 4.30 – Графік співвідношення точності розпізнавання

4.3 Висновки до розділу

У цьому розділі були проведені експериментальні дослідження роботи моделі розпізнавання паростків рослин, розробленої у програмному комплексі MATLAB/Simulink.

Із наведених результатів можна зробити декілька висновків:

- навіть незначна зміна задавальної точності розпізнавання (десяті частини) має великий вплив на точність розпізнавання об’єктів на зображеннях, як це було видно з рисунків 4.22, 4.23 та 4.24;
- зміна максимальної кількості одночасних розпізнавань вимагає більших затрат комп’ютерних ресурсів, в основному обчислювальної потужності центрального процесора і графічного процесора;
- зміна методу кореляції майже не впливає на точність розпізнавання.

Тому можна зробити висновок, що проведені експериментальні дослідження підтвердили правильність побудованої моделі.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту (товару, послуги, технології)

Розробка відноситься до систем автоматичного керування, призначена для автоматичного розбризкування гербіцидів за допомогою системи розпізнавання об’єктів.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Системи автоматичного керування	Збільшення ефективності догляду за сільськогосподарськими угіддями, значне зменшення вартості процесу догляду
	2. Наукова діяльність	Застосування новітніх технологій

Збільшення ефективності догляду за сільськогосподарськими угіддями, значне зменшення вартості процесу догляду робить розробку конкурентоспроможною. Відомо, що існуючі аналоги є дорожчими і не зовсім підходять до мети, з якою було розроблено робота, тому наявність дешевшого та унікального вітчизняного продукту, котрий допомагає в роботі з результатами дослідження є дуже перспективним. В таблиці 5.2 сильні, слабкі та нейтральні характеристики ідеї проекту.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			
		Мій проект	Adigo Field Flux Robot	Ladybird	Rosphere
1.	Вартість	Низька	Висока	Середня	Низька

2.	Надійність	Висока	Висока	Середня	Низька
3.	Трудомітскість виготовлення	Низька	Висока	Висока	Середня
4.	Час роботи	Висока	Середня	Висока	Низька

Продовження таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	W (Слабка сторона)	N (Нейтральна сторона)	S (Сильна сторона)
1.	Вартість			+
2.	Надійність			+
3.	Трудомітскість виготовлення			+
4.	Час роботи			+

5.2 Технологічний аудит ідеї проекту

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технологія її реалізації	Наявність технологій	Доступність технологій
		Ручне конструювання	Наявні	Так
		Фабричне конструювання	Необхідно розробити	Ні
		Моделювання системи за допомогою програмного комплексу MATLAB/Simulink	Наявні	Так

		Відтворення системи за допомогою мови програмування C#	Необхідно розробити	Так
		Розроблення клієнт-серверних додатків на мові програмування Java	Необхідно розробити	Так
		Розроблення клієнт-серверних додатків на мові програмування C#	Необхідно розробити	Так
Обраними технологіями є ручне конструювання робота задля меншої вартості обслуговування, моделювання системи розпізнавання об’єктів за допомогою програмного комплексу MATLAB/Simulink і розроблення клієнт-серверних додатків на мові програмування Java.				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од.	30 тис грн/ум.од.
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Новітня технологія, потребуюча практичної перевірки.
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (по ринку), %	50

Основною споживчою аудиторією є сільськогосподарська промисловість . Товар може розповсюджуватись без дистриб'юторів/представників. Розробка є універсальною. В таблиці 5.5 наведено характеристику потенційних клієнтів стартап-проекту.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Мала вартість товару			
2	Висока ефективність роботи			
3	Мала собівартість			

Серед основних загроз є конкурентоспроможність, тому необхідна реклама, щоб потенційні користувачі дізналися про наявність даного продукту. В таблиці 5.6 розглянуто фактори загроз.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкурентоспро-можність	Невідомий бренд, новий товар на ринку, низька конкурентоспроможність	Реклама, вдалий маркетинговий проект, залучення дилерів, спонсорів до співпраці
2	Невідомий бренд	Новий товар невідомої фірми	Розкрутка товару та розповсюдження серед

			промислових підприємств і в наукових центрах
	Комерційна таємниця	Можливість патентування технології невідомими робітниками	Підписання строгих контрактів з несенням матеріальної компенсації

Основною можливістю є залучення іноземних та вітчизняних спонсорів та можливість реалізації товару за кордоном. В таблиці 5.7 розглянуто фактори можливостей.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Залучення іноземних або вітчизняних спонсорів	Залучення іноземних капіталів, підтримка спонсорів	Укладення договорів про співпрацю та налагодження торгових контактів
2	Налагодження зв’язків з виробниками механічних деталей, поставниками електричних деталей та клієнтами	Залучення дистриб’юторів та наукових представників.	Пошук та налагодження зв’язків з компаніями, виробниками механічних деталей та промисловими підприємствами

В таблиці 5.8 проведено ступеневий аналіз конкуренції на ринку.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Чиста конкуренція	Усі виробники-конкуренти є приватними підприємствами або стартапами, не	Популяризація національного товару та залучення інвесторів.

	затверджені як монополія тощо	
2. Міжнаціональний рівень конкурентної боротьби	Виробники-конкуренти завойовують міжнародний ринок	Вихід на міжнародний ринок
3. Внутрішньогалузева	Розробка може бути використана лише в галузі сільського господарства	
4. Товарно-видова	Конкуренція між роботами, що виконують схожу роботу	
5. Цінова, якісна	Схожі функції, конкуренцію складає лише ціна та якість вироблення	
6. Марочна	Виробники-конкуренти зареєстровані під торговими марками	Реєстрація торгової марки, рекламування

В таблиці 5.9 проведено більш детальний аналіз умов конкуренції в галузі (за моделлю 5 сил М. Портера).

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Відсутні, або закордонні				
Висновки	Відносно невисока конкурентна боротьба				

Можлива робота на ринку, залежить від купівлеспроможності установ. В зв’язку з тим, що вони користуються значно дорожчим товаром закордонного виробництва, можна зробити висновок, що наш товар скоріше за все буде користуватись періодичним попитом. Фактор розповсюдження шляхом постачальників є також важливим. Конкурентна боротьба є середньою з закордонними та низькою з вітчизняними аналогами.

В таблиці 5.10 обґрунтовано фактори конкурентоспроможності.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Точність	Вища точність ніж у аналогів
2	Ціна	Набагато менша ціна товару
3	Якість	Не гірша від аналогів

В таблиці 5.11 проведено Порівняльний аналіз сильних та слабких сторін власного проекту.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін власного проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з власною компанією						
			-3	-2	-1	0	+1	+2	+3
1	Точність	18			+				
2	Ціна	20	+						
3	Якість					+			

В таблиці 5.12 проведено SWOT-аналіз стартап-проекту.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Ціна основана на собівартості Рентабельність Швидкодія	Слабкі сторони: Невідомий постачальник
Можливості: Забезпечення споживчих потреб Вихід на закордонний ринок Високий дохід	Загрози: Викриття комерційної таємниці Недостатньо досконала реалізація

В таблиці 5.13 розглянуто альтернативи ринкового впровадження стартап-проекту.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Практичне використання та вдосконалення алгоритму	+	0,5 року
2	Підключення іноземних фахівців	+	0,5 року
3	Рекламна кампанія	+	1 рік
4	Отримання міжнародних сертифікатів	+	1 рік
5	Вихід на закордонний ринок	+	1 рік

5.4 Розроблення ринкової стратегії проекту

В таблиці 5.14 проведено вибір цільових груп потенційних споживачів.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Промислове сільське господарство	Готові	Великий	Велика	Середня

2	Рядові фермери	Не готові	Великий	Мала	Легка
Обрано промислове сільське господарство					

В таблиці 5.15 проведено визначення базової стратегії розвитку.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Рядові фермери	Рекламна кампанія	- низька ціна - простота використання - значна економія	Стратегія лідерства по витратах

В таблиці 5.16 визначено базової стратегії конкурентної поведінки.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Буде шукати	Ні	Стратегія наслідування лідера

В таблиці 5.17 визначено стратегії позиціонування.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової	Базова стратегія	Ключові	Вибір асоціацій, які мають сформувати

	аудиторії	розвитку	конкурентоспроможні позиції власного стартап-проекту	комплексну позицію власного проекту (три ключових)
1	Низька ціна			
2	Висока ефективність			
3	Простота використання			

5.5 Розроблення маркетингової програми стартап-проекту

В таблиці 5.18 Визначено ключові переваги концепції потенційного товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Дешевизна	Низька ціна	Ціна
2	Висока ефективність	Висока ефективність за рахунок економії робочої сили і ресурсів (гербіциду)	Ефективність
3	Отримати товар з якістю закордонних аналогів та українською ціною.	Національний продукт	Універсальність

В таблиці 5.19 описано трьохрівневу модель товару.

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Покращення ефективності оброблення сільськогосподарських угідь</td></tr>
<tr><td rowspan="5"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Висока ефективність
2. Значна економія
3. Простота використання</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: стандарти, нормативи, параметри тестування тощо
Стандартизація відповідно до ГОСТ, ISO, МЕК.
Регламентується ГОСТ, ISO, МЕК.</td></tr>
<tr><td colspan="3">Пакування присутнє.</td></tr>
<tr><td colspan="3">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td colspan="4">Потенційний товар буде захищено від копіювання шляхом патентування і проходження сертифікатів відповідності.</td></tr>
</table>

В таблиці 5.20 проведено визначення меж встановлення ціни.

Таблиця 5.20 – Визначення меж встановлення ціни (із розрахунку на 1 рік користування)

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
>100 тис. грн.	>50 тис. грн.	20-30 тис. грн.	Верхня: 80 тис.грн. Нижня: 30 тис.грн.

В таблиці 5.21 розглянуто формування системи збуту.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Закупівлю доцільно здійснювати після роз'яснення обмежень та можливостей товару.	Обслуговування проданого товару, проведення маркетингових досліджень	Канал нульового рівня (прямий маркетинг)	Торгівля через власний веб сайт та інтернет магазини, що спеціалізуються на даному типі товару.

В таблиці 5.22 розглянуто концепцію маркетингових комунікацій.

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Довіра до промислових підприємств,	Наукові установи,	Налагодження контактів з	Донести ідею, інформацію	Продемон-струвати переваги

	котрі реко-мендують товар	державні інститути	промисловими підприємствами	про якість, цінову політику	перед існуючим товарами

5.6 Висновки

Ідеєю стартапу є підвищення ефективності оброблення сільськогосподарських угідь гербіцидами шляхом розроблення роботизованого комплексу точкового оброблення.

Є можливість ринкової комерціалізації проекту, так як наявний попит та потреба у даній технології на ринку України та закордоном.

Чиста конкуренція, локальна внутрішньовидова конкуренція, що сприяє для успішного входження на ринок.

Альтернативою можна назвати низьку ціну та постійну технічну підтримку. Подальша імплементація є доцільною для даного проекту.

ВИСНОВКИ

У даній магістерській дисертації розроблено нове рішення актуального наукового та практичного завдання підвищення ефективності оброблення сільськогосподарських угідь гербіцидами шляхом розроблення роботизованого комплексу точкового оброблення.

У першому розділи магістерської дисертації було зроблено огляд існуючих систем, подібних то розроблюваної. Було проаналізовано переваги та недоліки кожної з них та зроблено висновок про необхідний набір функціоналу для системи, що розроблена у магістерській дисертації.

У другому розділі надано теоретичні відомості методів, що лежать в основі системи розпізнавання рослин, вирощування яких виконується системою.

У третьому розділі роботи розроблено структурну схему системи, функціональну схему та алгоритм роботи робота.

В четвертому розділі значну увагу приділено моделюванню системи розпізнавання паростків рослин за допомогою програмного комплексу MATLAB/Simulink. Було розроблено модель системи розпізнавання рослин за допомогою тулбоксу Computer Vision System Toolbox. Для перевірки працездатності системи розпізнавання об’єктів було виконано експериментальні дослідження на основі трьох відеорядів, на яких зображені різні рослини, а саме: образки болотяні, картопля звичайна та буряк звичайний. Дослідження проводились для декількох навчальних вибірок та перевірялися на тестових вибірках. Отримані результати дозволяють зробити висновок про доцільність застосування методів глибинного навчання для досягнення поставленої в дисертації мети.

П’ятий розділ дисертації присвячений розробленню стартап-проекту, в якому виконано аналіз готового рішення з економічної сторони, аналіз ринку попиту на товар та аналіз конкуренції та зроблено висновок про перспективи виведення продукту на ринок та способів його розповсюдження.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

2. Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS'2012*

3. Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In *CVPR'2014* .

4. Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014d). Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In *International Conference on Learning Representations*

5. Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012b). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, **29**(6), 82–97.

6. Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS'2014, arXiv:1409.3215*

7. Collobert, R. (2011). Deep learning for efficient discriminative parsing. In *AISTATS'2011* .

8. Mnih, V. and Hinton, G. (2010). Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*.

9. Kiros, R., Salakhutdinov, R., and Zemel, R. (2014a). Multimodal neural language models. In *ICML'2014*.

10. Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. L. (2015). Deep captioning with multimodal recurrent neural networks. In *ICLR'2015* . arXiv:1410.1090.

11. Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015b). Show and tell: a neural image caption generator. In *CVPR'2015* . arXiv:1411.4555.

12. Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, **41**(3), 15

13. Luo, H., Carrier, P. L., Courville, A., and Bengio, Y. (2013). Texture modeling with convolutional spike-and-slab RBMs and deep extensions. In *AISTATS'2013*

14. Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics* , **7**, 179–188

15. Kotzias, D., Denil, M., de Freitas, N., and Smyth, P. (2015). From group to individual labels using deep features. In *ACM SIGKDD.*

16. Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, **16**, 264–280.

17. Goldberger, J., Roweis, S., Hinton, G. E., and Salakhutdinov, R. (2005). Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS'04)*. MIT Press.

18. Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA. ACM.

19. Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 (NIPS'95)*, pages 514–520. MIT Press, Cambridge, MA.

20. Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554.

21. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA

22. Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, **1**, 295–311.

23. Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* , **2**, 359–366.

24. Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, **3**(5), 551–560.

25. Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* , **6**, 861—867.

26. Barron, A. E. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory*, **39**, 930–945

27. Håstad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing* , pages 6–20, Berkeley, California. ACM Press.

28. Håstad, J. and Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, **1**, 113–129.

29. Maass, W. (1992). Bounds for the computational power and learning complexity of analog neural nets (extended abstract). In *Proc. of the 25th ACM Symp. Theory of Computing* , pages 335–344.

30. Pascanu, R., Montufar, G., and Bengio, Y. (2013b). On the number of inference regions of deep feed forward networks with piece-wise linear activations. Technical report, U. Montreal, arXiv:1312.6098.

31. Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *NIPS'2014*

32. Bengio, Y. (2009). *Learning deep architectures for AI* . Now Publishers.

33. Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

34. MathWorks - Makers of Matlab and Simulink [Електронний ресурс] / MathWorks – Режим доступу до ресурсу: https://www.mathworks.com/.

35. Выращивание картофеля из семян (часть 2) / Выращиваем суперэлиту картофеля из семян [Електронний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=5nk4mZBFBjM.

36. тонкости выращивания свеклы [Електронний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=RsKOlYXzWrs.