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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.032.26:519.2

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Алгоритмічно-програмний метод для прогнозування даних з мультимодальним розподілом на основі моделі MDN»

Виконав:
студент ІІ курсу, групи КП-31мн
Федорчук Іван Васильович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н.,
Шкурат Оксана Сергіївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПСКС, к.т.н., доцент,
Тарасенко-Клятченко Оксана Володимирівна _____

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

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Федорчуку Івану Васильовичу

1. Тема дисертації «Алгоритмічно-програмний метод для прогнозування даних з мультимодальним розподілом на основі моделі MDN», науковий керівник дисертації Шкурат Оксана Сергіївна, к.т.н., затверджені наказом по університету № 1219-С від «21» березня 2025 р.

2. Термін подання студентом дисертації «19» травня 2025 р.

3. Об'єкт дослідження: процес прогнозування даних із мультимодальним розподілом.

4. Предмет дослідження: методи та програмне забезпечення прогнозування даних з мультимодальним розподілом, зокрема ймовірнісні моделі та моделі нейронних мереж, методи оптимізації функцій втрат та способи наближеного обчислення інтегралів для покращення точності прогнозування.

5. Перелік завдань, які потрібно розробити:
- провести аналіз методів прогнозування даних з мультимодальних розподілом та даних з невизначеністю;
- дослідити способи оптимізації функції втрат моделі MDN;
- розробити та дослідити модифіковану модель MDN, яка використовую ймовірнісну функцію втрат;
- дослідити методи інтерпретації результуючого виходу моделі для отримання остаточних результатів із утвореного ймовірнісного розподілу;
- дослідити можливі способи вдосконалення та потенційні сфери застосування запропонованого методу;
- провести оцінки ефективності запропонованого методу у різних ситуаціях та провести порівняльний аналіз з іншими методами.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- графіки динамік змін метрик під час навчання моделей для вирішення різних задач;
- порівняльна таблиця оцінки запропонованого методу;
- алгоритм побудови моделі нейронної за запропонованим методом;
- архітектура програмного забезпечення реалізації запропонованого методу;
- порівняльна таблиця методів інтерпретації ймовірнісних розподілів за ключовими характеристиками;
- порівняльна таблиця методів наближеного обчислення визначеного інтегралу для суміші гаусових розподілів.

7. Орієнтовний перелік публікацій:
- стаття у фаховому журналі за спеціальністю 121 «Інженерія програмного забезпечення».
- Тези доповіді “ Алгоритмічно-програмний метод для прогнозування даних з мультимодальним розподілом на основі моделі MDN ”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «10» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	13.10.2023	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	07.12.2023	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	19.05.2024	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	23.09.2024	
5.	Проведення наукового дослідження; підготовка матеріалів доповіді на конференції ПМК-2024	18.10.2024	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; робота над статтею за результатами наукового дослідження;	12.12.2024	
7.	Завершення роботи над основною частиною магістерської дисертації;	01.03.2025	
8.	Оформлення текстової та графічної частини магістерської дисертації	05.05.2025	

Студент Іван ФЕДОРЧУК

Науковий керівник Оксана ШКУРАТ

РЕФЕРАТ

Актуальність теми. Мультимодальні розподіли даних є поширеним явищем у багатьох сучасних прикладних задачах – від медичної діагностики до автономного керування. Їх наявність ускладнює процес прогнозування, оскільки традиційні методи часто зводяться до усереднення значень, що не відображає реальної структури даних. Використання моделей змішаних щільностей (Mixture Density Network, MDN) дозволяє більш точно визначати результат прогнозування даних з мультимодальним розподілом, однак класичні реалізації MDN мають обмеження, пов’язані з чисельною нестабільністю та труднощами інтерпретації результатів. Актуальність даного дослідження полягає в необхідності вдосконалення методів прогнозування мультимодальних даних, зокрема шляхом застосування ймовірнісної функції втрат для навчання моделі MDN, щоб підвищити стабільність та точність прогнозування. Прогнозування даних з мультимодальним розподілом є важливою задачею для багатьох сфер, де рішення приймаються в умовах невизначеності.

Об’єктом дослідження є процес прогнозування даних із мультимодальним розподілом.

Предметом дослідження є методи та програмне забезпечення прогнозування даних з мультимодальним розподілом, зокрема ймовірнісні моделі та моделі нейронних мереж, методи оптимізації функцій втрат та способи наближеного обчислення інтегралів для покращення точності прогнозування.

Метою дослідження є підвищення точності прогнозування даних з мультимодальним розподілом на основі моделі машинного навчання архітектури MDN та ймовірнісної функції втрат для навчання моделі.

Наукова новизна. Уперше запропоновано алгоритмічно-програмний метод прогнозування даних з мультимодальним розподілом на основі моделі машинного навчання архітектури MDN, характерною особливістю

якого є застосування ймовірнісної функції втрат для навчання моделі, що дозволяє збільшити точність прогнозування на 4,56% для синтетичних наборів даних з чітко визначеними мультимодальними характеристиками, а також на 11,57% та 2,47% для двох наборів даних, що відображають прикладні задачі.

Практична значущість. Запропонований метод прогнозування даних з мультимодальним розподілом може використовуватись як самостійна система прогнозування в умовах невизначеності. Запропонований метод зберігає сумісність із класичною архітектурою MDN, що дозволяє легко інтегрувати його в існуючі системи прогнозування даних. Реалізація моделі на основі Python забезпечує гнучкість інтеграції у прикладні системи аналізу та прогнозування, а також дозволяє адаптувати метод під конкретні задачі шляхом зміни гіперпараметра ширини інтервалу.

Апробація роботи. Основні положення та результати виконаної роботи висвітлені в науковій статті «Algorithmic and Software Method for Predicting Data with Multimodal Distribution based on the MDN Model» у фаховому журналі «Herald of Khmelnytskyi National University. Technical sciences» категорії «Б» та представлені на XVII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК–2024 (20-21 листопада 2024 р., м. Київ).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі обґрунтовано актуальність теми, визначено мету, об'єкт і предмет дослідження.

У першому розділі проведений аналіз сучасних методів прогнозування даних з мультимодальним розподілом, виявлено переваги та недоліки існуючих методів, поставлені наукові задачі.

У другому розділі представлено запропонований метод прогнозування даних з мультимодальним розподілом на основі моделі

машинного навчання архітектури MDN та ймовірнісної функції втрат для тренування моделі.

У третьому розділі описано реалізацію запропонованого методу мовою програмування Python з урахуванням особливостей використаних бібліотек і технологій.

У четвертому розділі відображені результати експериментальних досліджень, проведена оцінка результатів прогнозування та аналіз ефективності запропонованого методу в порівнянні з класичною реалізацією MDN.

У висновках зазначено результати аналізу ефективності використання запропонованого алгоритмічно-програмного методу.

Робота виконана на 98 аркушах, містить 3 додатки та послання на список використаних літературних джерел з 26 найменувань. У роботі наведено 23 рисунків та 9 таблиць.

Ключові слова: інженерія програмного забезпечення, дані з мультимодальним розподілом, розподіл Гауса, модель MDN, функція втрат.

ABSTRACT

Theme urgency. Multimodal data distributions are a common challenge in many modern applied tasks, ranging from medical diagnostics to autonomous control. Their presence complicates the prediction process, as traditional methods often reduce outcomes to average values, which fail to capture the true structure of the data. Mixture Density Networks (MDNs) provide a more accurate way to model such data; however, classical MDN implementations suffer from numerical instability and difficulty in interpreting the results. The relevance of this study lies in the need to improve the prediction of multimodal data by introducing a probabilistic loss function into the training process of MDNs, enhancing both the stability and accuracy of predictions. Predicting data with multimodal distributions is crucial in many domains where decisions must be made under uncertainty.

The object of research is the process of predicting data with a multimodal distribution.

The subject of research includes methods and software for predicting such data, especially probabilistic and neural network models, optimization techniques for loss functions, and approaches to approximate integral calculation for improved prediction accuracy.

The research objective is to improve the accuracy of predicting data with a multimodal distribution based on the MDN architecture and a probabilistic loss function for training.

Scientific novelty. For the first time, an algorithmic and software method for predicting data with a multimodal distribution based on the machine learning model of the MDN architecture has been proposed, a characteristic feature of which is the use of a probabilistic loss function for model training. This approach improves prediction accuracy by 4.56% on synthetic datasets with clearly defined multimodal characteristics and by 11.57% and 2.47% on real-world datasets.

Practical value. The proposed method can be used as a standalone prediction system under uncertainty. It remains compatible with the classical MDN architecture, facilitating integration into existing prediction systems. The implementation in Python ensures flexibility for applied analysis and forecasting systems and allows for task-specific adaptation via the interval width hyperparameter.

Approbation. The main concepts and results were presented at the 17th Scientific Conference for Master's and PhD students "Applied Mathematics and Computing" PMK–2024 (November 20–21, 2024, Kyiv) and published in the professional journal "Herald of Khmelnytskyi National University. Technical Sciences" (Category "B") in the article "Algorithmic and Software Method for Predicting Data with Multimodal Distribution Based on the MDN Model."

Structure and content of the thesis. The master's thesis consists of an introduction, four chapters, conclusions, and appendices.

The introduction outlines the relevance of the topic and defines the aim, object, and subject of the study.

Chapter one analyzes current methods for predicting multimodal data, identifies their strengths and limitations, and sets out the scientific objectives.

Chapter two introduces the proposed method based on the MDN architecture and the probabilistic loss function.

Chapter three details the implementation of the method in Python, including the libraries and technologies used.

Chapter four presents the results of experimental research, compares the performance of the proposed method with the classical MDN approach, and analyzes its effectiveness.

The conclusions summarize the effectiveness of the proposed algorithmic and software solution.

The work is completed on 98 sheets and contains 3 appendices and references to the list of used literary sources of 26 titles. The work contains 23 figures and 9 tables.

Keywords: software engineering, data with multimodal distribution, Gaussian distribution, MDN model, loss function.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API (Application Programming Interface) – інтерфейс прикладного програмування, який забезпечує взаємодію між компонентами програмного забезпечення.

BIC (Bayesian Information Criterion) – байєсівський інформаційний критерій, що використовується для оцінки якості моделей при наявності обмеженого набору даних.

CDF (Cumulative Distribution Function) – кумулятивна функція розподілу, яка визначає ймовірність того, що випадкова величина набуде значення, меншого або рівного заданому.

Dropout – техніка регуляризації в глибокому навчанні, яка випадково «відключає» частину нейронів під час тренування, щоб запобігти перенавчанню.

GMM (Gaussian Mixture Model) – гаусова змішана модель, яка представляє складні розподіли даних як суміш кількох нормальних (гаусових) розподілів.

HRI (Human-Robot Interaction) – взаємодія людини та робота, галузь, яка вивчає способи ефективної комунікації та співпраці між людьми та роботами.

KL-дивергенція (Kullback-Leibler Divergence) – міра відмінності між двома ймовірнісними розподілами, яка часто використовується для регуляризації моделей.

MDN (Mixture Density Network) – мережа змішаних щільностей, нейронна мережа, яка генерує параметри суміші гаусових розподілів замість одного вихідного значення.

ML (Machine Learning) – машинне навчання, галузь штучного інтелекту, що створює моделі, які здатні навчатися на основі даних.

MoG (Mixture of Gaussians) – суміш гаусових розподілів, базова концепція для MDN, що дозволяє моделювати багатомодальні розподіли.

NLL (Negative Log-Likelihood) – від'ємна логарифмічна правдоподібність, стандартна функція втрат у MDN, яка максимізує ймовірність передбачуваних результатів.

PgMDN (Physics-guided Mixture Density Network) – фізично орієнтована мережа змішаних щільностей, яка інтегрує фізичні знання в архітектуру або функцію втрат.

ReLU (Rectified Linear Unit) – функція активації, що дорівнює нулю для від'ємних значень та зростає лінійно для додатних.

RL (Reinforcement Learning) – навчання з підкріпленням, тип машинного навчання, де модель вчиться на основі нагород за свої дії.

Softmax – функція, яка перетворює набір чисел у ймовірності, використовується в MDN для нормалізації вагових коефіцієнтів суміші.

VAE (Variational Autoencoder) – варіаційний автокодер, ймовірнісна модель, яка вивчає латентні представлення даних та генерує нові зразки.

ε (епсілон) – гіперпараметр у запропонованому методі, що визначає ширину інтервалу навколо цільового значення для обчислення ймовірності.

μ (мю) – середнє значення компонента гаусового розподілу у MDN.

σ (сигма) – стандартне відхилення компонента гаусового розподілу у MDN.

π (пі) – вага або ймовірність вибору відповідного компонента в суміші MDN, яка нормалізується функцією softmax.

ВСТУП

Ймовірнісні моделі відіграють ключову роль у сучасному машинному навчанні, особливо в ймовірнісному моделюванні та статистичному висновку. Ці моделі забезпечують сувору математичну основу для отримання змістовної інформації з даних, особливо в умовах невизначеності або неповної інформації. Вони є особливо корисними в ситуаціях, коли одному входу логічно можуть відповідати кілька правильних виходів – випадки, що часто трапляються в реальних завданнях, таких як обробка зображень, розпізнавання шаблонів та ухвалення рішень.

Яскравим прикладом є семантична сегментація зображень, основною метою якої є призначення кожному пікселю зображення певного наперед визначеного класу. Цей процес стає особливо складним у прикордонних регіонах, де пікселі часто представляють перехідні або перекривні області між різними класами, що призводить до значної неоднозначності класифікації. Така невизначеність ще більше ускладнюється, коли пікселі можуть правдоподібно належати до кількох категорій одночасно через перекриття об’єктів або складність сцени. Дані з подібною невизначеністю зазвичай мають мультимодальний розподіл, що ускладнює точну класифікацію за допомогою звичайних детерміністичних моделей.

Щоб впоратися з такими мультимодальними даними, були успішно застосовані підходи, засновані на ймовірнісному моделюванні, оскільки вони надають передбачення у вигляді ймовірностей, присвоюючи ймовірнісні оцінки різним можливим результатам замість обмеження передбачення одним детерміністичним результатом. Зокрема, одна з найефективніших нейронних мереж такого типу – це мережа змішаних щільностей (Mixture Density Network, MDN) [1]. MDN використовує суміш нормальних розподілів для оцінки ймовірностей кількох потенційних результатів, що дозволяє ефективно відображати складні, мультимодальні простори виходів. Кількість

компонентів Гауса – важливий гіперпараметр – визначає гнучкість і здатність MDN точно моделювати різноманітні розподіли даних.

Незважаючи на успішне використання MDN для представлення мультимодальних даних, її стандартна реалізація здебільшого покладається на передбачення подібності, що може обмежити ефективність у певних складних сценаріях, особливо при роботі з чітко вираженими бімодальними або різко мультимодальними розподілами даних. Щоб подолати це обмеження, у цій роботі досліджується модифікація структури MDN шляхом інтеграції ймовірнісних методів моделювання з метою підвищення здатності до навчання та точності MDN, спеціально адаптованої для точного опису даних із бімодальним та мультимодальним розподілом. Запропоноване ймовірнісне розширення покликане забезпечити кращу диференціацію між кількома допустимими виходами, тим самим підвищуючи точність і стійкість передбачень у складних задачах мультимодальної класифікації.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Вивчення проблематики та логіко-структурний аналіз проблеми

У сучасному світі методи класифікації, засновані на машинному навчанні, є надзвичайно актуальними через швидке зростання обсягів даних та необхідність їхньої ефективної обробки. Різні сфери діяльності активно використовують ці методи для автоматизації, підвищення точності аналізу та прийняття рішень.

Класифікація – це техніка, яка включає класифікацію даних у окремі класи. Це рекурсивний процес, який розпізнає та групує об’єкти даних у попередньо визначені категорії або мітки [2]. Класифікація може застосовуватися щодо найрізноманітніших типів даних, що значно розширює його область застосування, зокрема:

- числові дані;
- текстові дані;
- категоріальні дані;
- зображення;
- відео;
- аудіо та інші.

В свою чергу алгоритми машинного навчання дозволяють на основі реальних наборів даних навчати моделі, які уміють розпізнавати складні зв’язки всередині даних та виконувати прогнозування щодо тих вхідних даних, які відсутні у тренувальній базі та присвоювати їм відповідні категорії. Саме через цю властивість методи машинного навчання найчастіше застосовують до даних, які мають складну структуру, зокрема до медіа даних, де зв’язки між ними та класами є складними та менш очевидними.

Машинне навчання відіграє ключову роль у розв’язанні задач класифікації, оскільки дозволяє автоматично аналізувати великі обсяги даних, виявляти закономірності та приймати обґрунтовані рішення.

Серед практичних застосувань методів, що базуються на машинному навчанні, враховуючи різний тип вхідних даних, можемо навести наступні приклади у контексті розв’язання задач класифікації [3]:

- Числові дані – передбачення стану здоров’я пацієнта, спираючись на його медичні числові метрики; оцінка кредитоспроможності клієнтів за рахунок їх класифікації у категорії «надійний» та «ненадійний», враховуючи їх фінансові показники; аналіз та класифікація даних із сенсорів на підприємствах для виявлення аномалій у системі запобігання поломок і критичних ситуацій.
- Текстові дані – аналіз історії цін акцій та основних чинників для передбачення їх подальшого тренду, а саме зростання чи спадання; розпізнавання емоційного тону коментарів, зокрема негативний чи позитивний, для можливості застосування їх фільтрації та покращення досвіду користувачів; розпізнавання тематики тексту та віднесення його до однієї з категорій, наприклад: «політика», «історія», «економіка», «спорт», тощо, для обробки та впорядкування великого набору старих документів, а також автоматизації процесу для нових.
- Зображення – розпізнавання об’єктів на зображеннях за категоріями; класифікація МРТ або рентгенівських знімків для виявлення аномальних ділянок для асистування у визначенні наявності хвороб та встановлення діагнозу; автоматичне розпізнавання тексту на зображеннях, зокрема і рукописного; аналіз супутникових знімків планети для проведення моніторингу віддалених ділянок та виявлення аномалій, наприклад, лісових пожеж чи повеней.
- Відео – розпізнання підозрілої поведінки людей на записах камер відеоспостережень; визначення типу дій спортсменів під час трансляції спортивних матчів; класифікація відеоматеріалів як реальні чи згенеровані штучним інтелектом для запобігання

дезінформації, розпізнавання об’єктів військової техніки на відео, які забезпечують розвідувальні системи.

- Аудіо – визначення емоційного стану людини за звуковими характеристиками аудіозапису її розмови; класифікація музики за жанрами; класифікація аудіозапису за мовою.

Окремим випадком задачі класифікації є задача прогнозування даних, що мають мультимодальний розподіл. Мультимодальний розподіл – це розподіл даних, який має кілька піків (мод), що свідчить про наявність декількох підгруп або кластерів у вибірці [4]. У задачах класифікації такі розподіли ускладнюють застосування традиційних методів, особливо якщо підгрупи значно відрізняються між собою. Таке явище відбувається через різні причини, зокрема через природні особливості джерела даних, змішування кількох груп даних або складні взаємозв’язки між ознаками. Серед основних причин виникнення таких наборів даних можемо виділити наступні [5]:

- різні частини даних можуть походити з різних груп, відповідно, визначальні характеристики елементів однієї групи можуть відрізнятися від іншої;
- дані відображають декілька різних режимів поведінки;
- штучне змішування декількох вибірок з різними закономірностями.

Наведемо практичні приклади наборів даних з мультимодальним розподілом:

- У наборі, який містить дані як про чоловіків так і жінок значення розподілу зросту (чи деяких інших фізіологічних показників) матиме два піки.
- В залежності від вподобання людей та через різні сценарії взаємодії з системою розподіл часу перегляду одного і того ж відео (чи групи відео з однієї категорії) може мати декілька піків, наприклад, короткий перегляд (менше однієї хвилини) та довгий перегляд (близький до довжини самого відео).

- Через різні поведінкові шаблони набір даних про швидкість водіїв на дорогах може мати декілька піків, оскільки, деякі з водіїв дотримуються швидкості, рекомендованої правилами дорожнього руху швидкості, інші ж її перевищують.
- У наборі даних про динаміку росту дерев може бути декілька пікових значень через штучне поєднання кількох вибірок, а саме, якщо дані збираються на основі дерев декількох різних видів (рис. 1).
- Під час аналізу даних навантаження інтернет мережі у житлових будинках в залежності від часу доби можемо отримати один пік у ранкові години, перед тим як жителі будинку вирушають на роботу, а інший – у вечірні, коли повертаються з роботи.
- У випадку аналізу набору даних, що містить оцінки студентів за іспит, варто враховувати, що їх розподіл може бути бімодальним за рахунок поділу студентів за поведінковою моделлю на тих добре підготувався і отримали високі бали та тих, хто не готувався, або виділили мало часу і отримали низькі бали (рис. 2).

Важлива особливість, яку потрібно враховувати під час прогнозування таких даних, полягає у тому, що нас може задовільнити значення однієї або кількох мод, взятих окремо, проте усереднене значення усіх мод зазвичай не є коректним кінцевим результатом. В контексті прикладу з даними про оцінки студентів можна вділити значення мод щодо оцінок, які мають найбільшу частоту рівне 74 та 88, проте їх усереднене значення, рівне 81, не є задовільним результатом в такому випадку (рис. 3).

Крім цього мультимодальний розподіл даних часто зустрічається у контексті зворотніх проблем – задачах, в яких на основі наявних наслідків необхідно з’ясувати причини [6]. В таких ситуаціях часто може виникати невизначеність та неоднозначність у прогнозованих результатах. Такі задачі виникають у різних сферах людської життєдіяльності.

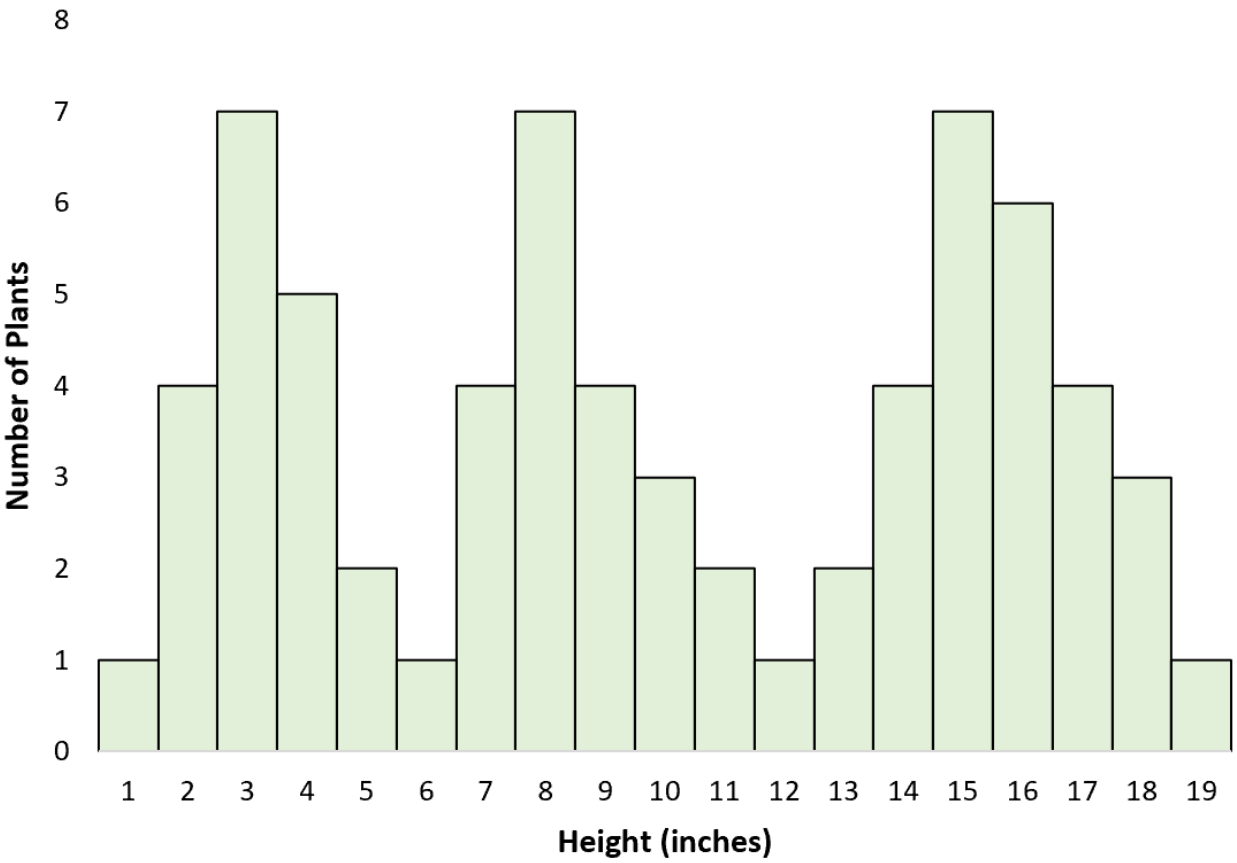


Рис. 1. Приклад набору даних з мультимодальним розподілом

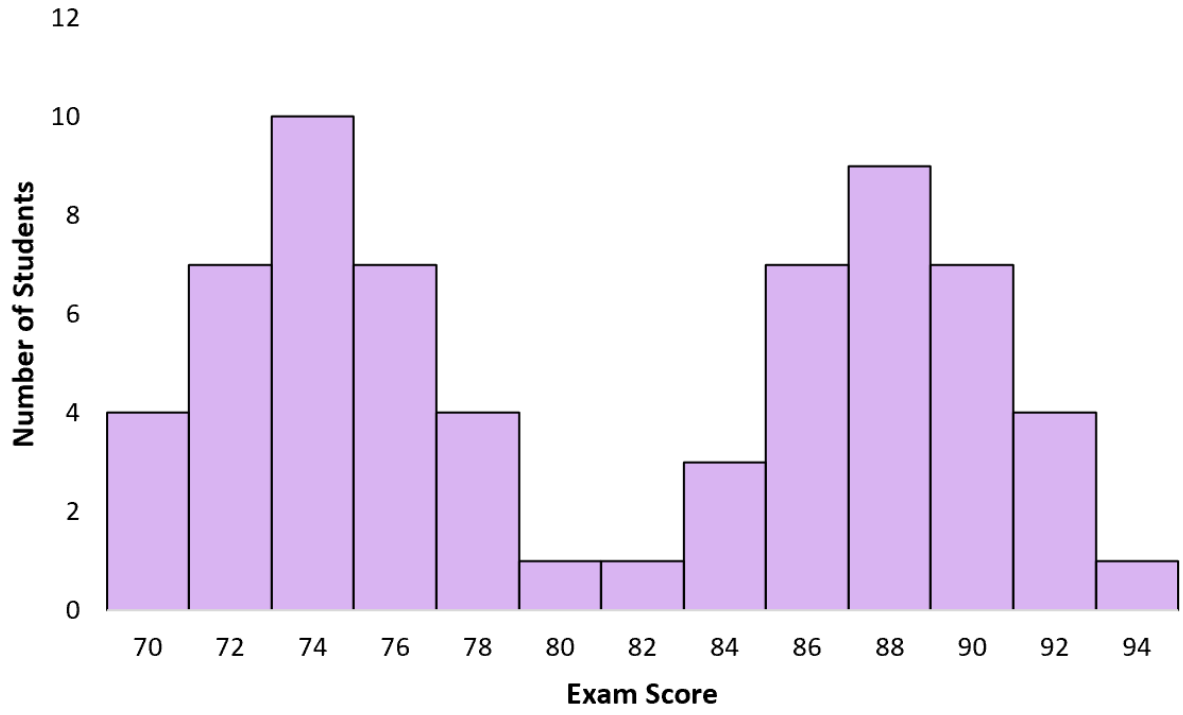


Рис. 2. Приклад набору даних з бімодальним розподілом

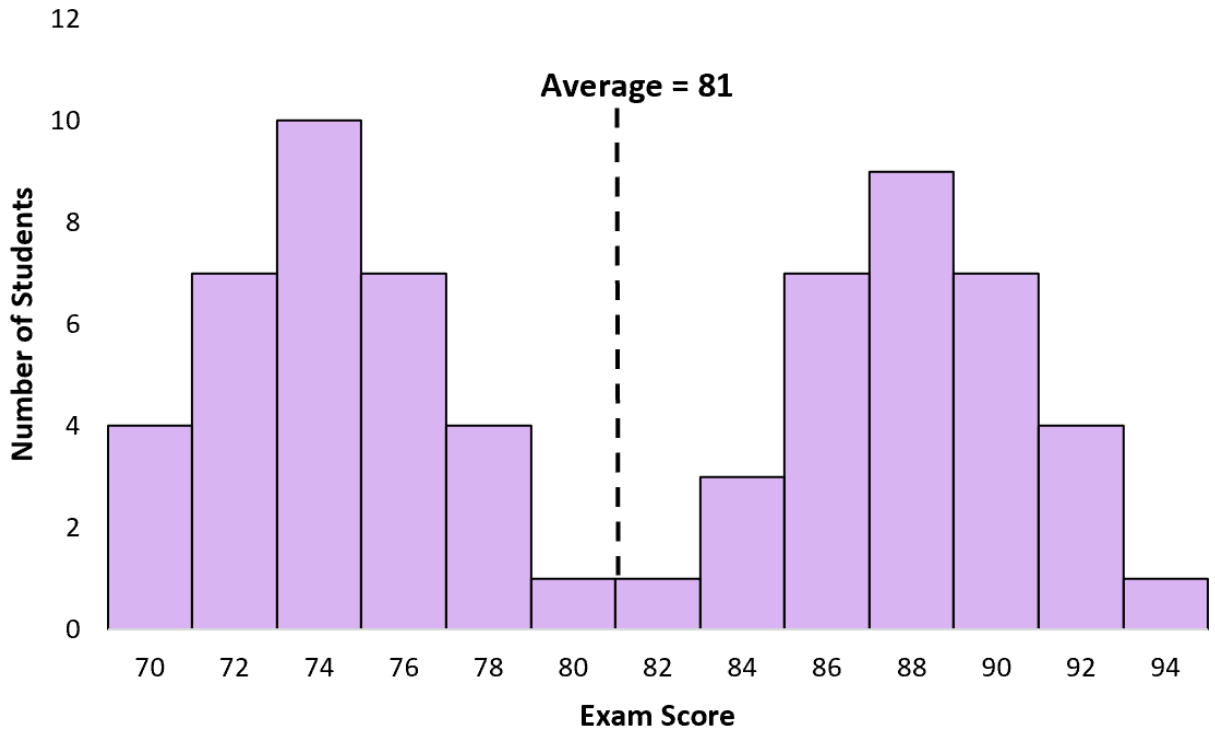


Рис. 3. Приклад, коли усереднене значення усіх мод не є модою

Поміж конкретних практичних прикладів можемо навести задачу визначення хвороби за її симптомами. Деякі хвороби можуть мати дуже схожі симптоми, тому в випадках, коли маємо типові симптоми грипу важко дати однозначну відповідь, оскільки більшість його симптомів збігаються з такими захворюваннями, як COVID, респіраторно-синцитіальною інфекцією та стрептококовою інфекцією горла [7]. Іншим прикладом є процес створення цифрової 3D-моделі з однієї 2D-проекції. Оскільки відразу декілька різних об'єктів можуть мати однакову проекцію, то і коректних кінцевих результатів може бути декілька.

Ймовірнісні моделі успішно застосовуються в галузі взаємодії людини й робота (Human-Robot Interaction, HRI), де здатність інтерпретувати та передбачати людську поведінку є критично важливою для досягнення природної й ефективної співпраці. На відміну від детерміністичних систем, які покладаються на фіксовані правила, ймовірнісні підходи дозволяють роботам опрацьовувати невизначеність, неоднозначність і варіативність – невід’ємні характеристики людських дій і рішень.

У цьому контексті особливо цінними є мультимодальні ймовірнісні моделі. Людська поведінка часто має декілька правдоподібних трактувань або варіантів реагування, навіть за однакового контексту. Наприклад, коли людина тягнеться до об’єкта, вона може мати намір підняти його, пересунути або просто вказати на нього. Навчаючись на прикладах демонстрацій людської поведінки, такі моделі здатні засвоїти розподіл можливих результатів і відобразити весь спектр потенційних намірів.

Ці моделі дозволяють роботам:

- Формувати контекстуально обґрунтовані й адаптивні реакції.
- Розпізнавати та передбачати різноманітні траєкторії людського руху.
- Координувати дії в рамках спільних завдань (наприклад, передача об'єктів, навігація в спільному робочому просторі).
- Забезпечувати безпечну та природну взаємодію з людьми.

Методи на зразок ймовірнісного прогнозування траєкторій [8], мереж змішаних щільностей (MDN) та варіаційного висновку [9] вже продемонстрували свою ефективність у вирішенні таких задач. Це сприяє створенню більш гнучкої, обізнаної про контекст і надійної поведінки роботів – важливого кроку до їх інтеграції в реальні середовища.

Враховуючи проведений аналіз предметної області, можемо сказати, що задача прогнозування даних з мультимодальним розподілом є надзвичайно актуальною у сучасному світі, оскільки такі дані зустрічаються у багатьох реальних сценаріях – від фінансової аналітики та медичної діагностики до обробки природної мови та комп’ютерного зору. Наявність кількох піків у розподілі свідчить про складну структуру даних, що ускладнює використання традиційних методів класифікації та прогнозування.

1.2. Огляд існуючих рішень

Виходячи з проведеного аналізу можемо зробити висновок, що задача прогнозування даних з мультимодальним розподілом потребує врахування

певних особливостей, тому звичайні методи прогнозування даних можуть видавати помилкові результати у таких ситуаціях. Беручи цей факт до уваги можемо сказати, що розглянута задача потребує особливих підходів, які б враховували вищезгадані особливості.

1.2.1. *Ймовірнісні моделі*

Ймовірнісні моделі є центральними для багатьох алгоритмів машинного навчання, оскільки забезпечують математичну основу для побудови висновків на основі вхідних даних. Дані моделі кількісно визначають ймовірність подій, закріплюючи прогнози в математичній системі, яка варіюється від абсолютної впевненості до повної неймовірності. Вони особливо корисні у випадках коли у результуючих даних може існувати певна невизначеність, або коли для одного набору вхідних даних може існувати відразу декілька правильних відповідей.

Однією з визначальних компонент даного підходу є використання розподілу ймовірностей. Це інструменти, які дозволяють ймовірнісним моделям впоратися з невизначеністю вхідних даних, пропонуючи структурований спосіб боротьби з випадковістю. В даному випадку пропонується не лише прогнозоване значення, а і, наприклад, певний коефіцієнт, який відповідає за відображення невизначеності даного прогнозу.

Серед найуживаніших ймовірнісних моделей можемо виділити наступні:

- Байєсівські моделі.
- Гаусові змішані моделі.

Обидва підходи часто застосовуються у задачах кластеризації та класифікації, в тому числі і у випадках, коли дані можуть мати мультимодальний розподіл. Розглянемо більш детально принцип їх роботи.

1.2.2. *Гаусова змішана модель*

Гаусова змішана модель (GMM) – це ймовірнісна модель, яка припускає, що дані походять із суміші кількох Гаусових розподілів. GMM є

гнучкою моделлю, яка добре підходить для кластеризації, оцінки розподілу даних та виявлення аномалій [10]. Дана модель використовує наступну формулу (1) для представлення ймовірнісного розподілу кожної з точок:

$$p(y) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mu_k, \sigma_k^2). \quad (1)$$

Крім цього, на відмінну від багатьох інших ймовірнісних методів, GMM використовує підхід максимізації очікування (Expectation-maximization) замість максимізації оцінки подібності (Maximum likelihood estimator). Даний підхід заснований на обчисленнях матриць ймовірностей приналежності кожного елементу до будь-якого кластеру та послідовному оновленні параметрів моделі на основі розрахованих ймовірностей.

Серед негативних сторін даного методу можемо виділити високу залежність точності кінцевої моделі від початкових значень параметрів. Крім цього за великої кількості Гаусових компонент навчання та робота моделі потребуватиме великої кількості ресурсів.

1.2.3. *Байєсівська модель*

Іншою ймовірнісною моделлю, яку ми розглянемо є Байєсівська модель. Якщо звертатися до визначення, то Байєсівською моделлю можемо описати клас ймовірнісних моделей, які в своїй основі використовують теорему Байєса, яка враховує нові спостереження та на їх основі оновлює значення параметрів розподілу даних. Таким чином, враховуючи попередньо надану інформацію, яку вважаємо апріорною, можемо покращити знання моделі про закономірності в даних, моделюючи невизначеність та створюючи гнучкі прогнози [11].

Власне теорема Байєса, записана у математичній формі, має наступний вигляд (2):

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}. \quad (2)$$

Звернемо більш детальну увагу на компоненти даної формули:

- *P(H|D)* – позначає апостеріорну ймовірність гіпотези *H* після спостереження даних *D*.

- *P(D|H)* – позначає правдоподібність даних *D* за умови, що гіпотеза *H* виявилася істинною.
- *P(H)* – уособлює попередні знання моделі, позначаючи апріорну ймовірність гіпотези *H*.
- *P(D)* – позначає маргінальну ймовірність даних *D*, яка використовується у якості нормувального фактору.

Для побудови Байєсівської моделі за загальним підходом повинні скористатися наступним узагальненим алгоритмом. Для початку опрацьовуються стартова інформація, яка може бути представлена у вигляді попереднього досвіду чи експертних знань, та на її основі формується апріорний розподіл параметрів моделі. На наступних етапах модель починає роботу з новими даними та з кожним новим спостереженням оновлює розподіл параметрів, таким чином покращуючи результати своїх передбачень. Особливістю даної моделі є те, що вона не дає точкові оцінки параметрів, а натомість працює з ймовірнісними розподілами, що дозволяє відображати невизначеність у прогнозі та отримувати інтервал довіри.

Через такий принцип роботи Байєсівські моделі є гнучкими та адаптивними, оскільки вони постійно оновлюються на основі нових спостережень. Важливим є той фактор, що такі моделі можуть працювати із мультимодальними розподілами. Проте такий підхід вимагає великих обчислювальних потужностей, тому під час практичного використання часто доводиться застосовувати апроксимаційні методи. Особливо це відчутно у випадках, коли потрібно враховувати великі набори вхідних даних. Ще одним недоліком Байєсівських моделей є висока залежність ефективності від вибору апріорного розподілу. Тобто, якщо початкова інформація для моделі є неякісною, то точність її прогнозів буде низькою.

1.2.4. *MDN з використанням бінарних інтервалів*

Оригінальна модель MDN, запропонована Крістофером Бішопом [1], поєднує нейронні мережі з сумішшю нормальних (Гаусових) розподілів для

моделювання умовної щільності ймовірностей. Хоча MDN є ефективним інструментом для відображення мультимодальних виходів, вона покладається на функцію втрат у вигляді негативного логарифма ймовірності (Negative Log-Likelihood, NLL). Такий підхід може спричиняти чисельну нестабільність, особливо коли передбачувані ймовірності надто низькі або надто високі. Ця нестабільність ускладнює процес навчання та може негативно впливати на точність передбачень моделі.

У відповідь на ці проблеми в останніх дослідженнях було запропоновано альтернативні функції втрат, які краще підходять для задач мультимодальної регресії. Наприклад, у роботі "Multimodal regression – Beyond L1 and L2 loss" [12] розглянуто використання функцій втрат з багатьма бінарними інтервалами (multi-bin loss), як засіб подолання обмежень класичних L2 втрат. L2 втрата ґрунтується на припущенні про унімодальний (одновершинний) нормальний розподіл, що часто призводить до «розмитих» передбачень у випадках, коли дані мають декілька можливих виходів.

Метод multi-bin полягає в дискретизації простору виходів на кілька бінів (інтервалів) і присвоєнні кожному з них ймовірності. Такий підхід дозволяє моделі краще захоплювати варіативність можливих результатів, що особливо важливо у складних мультимодальних сценаріях. Це також дозволяє уникнути деяких обмежень, притаманних MDN з функцією NLL, надаючи більш стабільне навчання та точніші передбачення, особливо у випадках, коли вихідні значення мають чітко виражену багатовершинну структуру.

Попри численні переваги, підхід, що базується на multi-bin loss або класичній MDN, має низку суттєвих недоліків, які обмежують його ефективність у певних контекстах. Серед основних таких недоліків можемо визначити наступний перелік:

- *Висока обчислювальна складність*. Для досягнення високої точності дискретизація вихідного простору він потребує великої кількості

бінів, що призводить до збільшення розміру вихідного шару та обсягів обчислень.

- *Зниження точності на межах бінів.* Дискретизація простору може призводити до втрати точності, особливо якщо істинне значення лежить між двома бін-сегментами. Це робить передбачення менш гладкими й створює штучні границі між сусідніми значеннями.
- *Погана інтерпретованість.* Виходи multi-bin моделей складно інтерпретувати у зрозумілій формі. Особливо це стосується кількох піків у передбаченнях: модель може передбачити декілька правдоподібних значень, але без чіткої індикації «найбільш доцільного» з них для практичного використання.
- *Потреба у великій кількості даних.* Щоб ефективно навчити multi-bin модель для захоплення мультимодальних розподілів, потрібні значні обсяги репрезентативних даних. У випадку нестачі даних модель часто тяжіє до унімодальних передбачень або некоректної оцінки ймовірностей.
- *Складність оптимізації.* Підбір кількості бінів і правильного масштабу може вимагати трудомісткого гіперпараметричного налаштування.

Ці недоліки вказують на необхідність подальшого вдосконалення підходів до моделювання мультимодальних розподілів.

1.2.5. *Variational Autoencoders*

Варіаційні методи, зокрема варіаційні автокодери (Variational Autoencoders, VAEs), набули широкого застосування для моделювання складних мультимодальних розподілів даних. На відміну від детерміністичних моделей, VAEs вводять ймовірнісний латентний простір, який за допомогою варіаційного висновку дозволяє моделі ефективніше відображати приховану структуру розподілу даних [13]. Це дозволяє

створювати кілька правдоподібних варіантів виходу для одного входу, що є особливо корисним у випадках з великою варіативністю та невизначеністю.

На відміну від моделей MDN, які явно моделюють умовні ймовірнісні розподіли через суміші Гауса, VAEs неявно навчаються структури даних, перетворюючи вхідні дані у нижчовимірний латентний простір із подальшим стохастичним семплюванням. Це надає моделі гнучкості та масштабованості, особливо у випадках роботи з високовимірними або надзвичайно варіативними даними.

Крім того, VAEs природно враховують невизначеність, генеруючи розподіли над виходами, а не лише точкові передбачення. Така властивість є надзвичайно корисною для задач на кшталт синтезу зображень, прогнозування часових рядів або семантичної сегментації в умовах неоднозначності. Розширені варіанти, такі як умовні варіаційні автокодери (Conditional VAEs), дозволяють включати додаткові вхідні умови, забезпечуючи контрольовану генерацію та покращене представлення структурованих мультимодальних вихідних просторів.

Основні недоліки VAEs:

- *Розмитість результатів.* Подібно до моделей із L2-втратою, стандартні VAEs схильні генерувати розмиті передбачення, особливо при моделюванні детальних мультимодальних даних, через мету максимізації нижньої межі доказу (ELBO).
- *Втрати модальностей (mode collapse).* Без додаткових налаштувань архітектури чи регуляризації модель може неефективно відображати всі моди розподілу.
- *Складність навчання.* Процес оптимізації включає балансування між помилкою реконструкції та терміном KL-дивергенції, що може бути складно, особливо для моделей із великою пропускною здатністю.

1.2.6. *Physics-guided Mixture Density Networks*

Значним кроком уперед у ймовірнісному моделюванні мультимодальних даних стало впровадження предметно-орієнтованих знань, зокрема фізичних законів, у процес навчання моделей. Ця ідея лягла в основу мереж змішаних щільностей, керованих фізикою (Physics-guided Mixture Density Networks, PgMDNs) – розширення стандартної MDN-архітектури, що включає фізичні закони, обмеження або емпіричні рівняння безпосередньо в архітектуру нейромережі або у функцію втрат [14].

Основна мотивація PgMDNs полягає у спрямуванні процесу навчання на отримання фізично обґрунтованих передбачень, особливо у випадках, коли суто дата-дрівен моделі можуть генерувати результати, що суперечать відомим фізичним принципам. Наприклад, у галузях структурної інженерії чи кліматичного моделювання вкрай важливо, щоб передбачення відповідали законам збереження (енергії, маси) або співвідношенням напружень і деформацій. PgMDNs досягають цього шляхом включення фізичних обмежень у вигляді регуляризаційних термів, залишків фізичних рівнянь або гібридних функцій втрат.

Інтеграція фізичних знань підвищує не лише інтерпретованість та надійність передбачень, а й покращує здатність до узагальнення, особливо в умовах обмеженої кількості тренувальних даних. Крім того, це забезпечує природну форму регуляризації, зменшуючи ризик перенавчання та підвищуючи стійкість моделі до екстраполяційних сценаріїв – наприклад, прогнозування екстремальних подій чи поведінки в нових умовах.

Основні недоліки PgMDNs:

- Потребують глибоких предметних знань для правильного формалізування фізичних обмежень.
- Важко застосовуються в галузях, де фізичні закони або слабо визначені, або дуже нелінійні.

- Ускладнене проєктування та навчання моделі через необхідність поєднання декількох функцій втрат або врахування фізичних залишків.

У сферах автономного водіння та робототехніки задача прогнозування траєкторій агентів (наприклад, автомобілів чи пішоходів) є мультимодальною за своєю природою, оскільки існує багато потенційно правдоподібних варіантів руху в майбутньому. Традиційні моделі часто орієнтовані на передбачення одного «найбільш ймовірного» шляху, що обмежує їхню здатність враховувати повний спектр можливих варіантів розвитку подій.

Щоб подолати ці обмеження, в новітніх дослідженнях запропоновано стратегії навчання з багатьма функціями втрат, які безпосередньо стимулюють різноманітність, фізичну допустимість та напрямну узгодженість у передбаченнях. Зокрема, у дослідженні Rahimi & Alahi [15] представлено композитну функцію втрат, яка включає:

- Off-road loss (втрата за виїзд за межі дороги): Штрафує траєкторії, що виходять за межі проїзної частини чи допустимого простору, забезпечуючи фізичну достовірність прогнозів.
- Diversity loss (втрата за нестачу варіативності): Стимулює модель до генерації декількох відмінних траєкторій, що покращує відображення мультимодальної невизначеності.
- Direction consistency error (помилка напрямної узгодженості): Забезпечує логічну послідовність між передбаченням і попереднім рухом, зменшуючи кількість нереалістичних змін напрямку.

Комбіноване використання цих втрат створює цілісну навчальну парадигму, яка набагато краще узгоджується з реальними обмеженнями та поведінковими патернами. Це дозволяє створювати надійніші, точніші та більш інтерпретовані моделі прогнозування, що є критично важливим для систем автономного керування або роботів, які взаємодіють з людьми в динамічному середовищі.

1.2.7. *Мультимодальне глибинне навчання*

Окрім моделей MDN, у сучасному штучному інтелекті активно розвивається підхід мультимодального глибинного навчання, який дозволяє обробляти та інтегрувати інформацію з різнорідних джерел даних – таких як текст, зображення чи аудіо. Такі моделі покликані поєднувати й узгоджувати дані з різних модальностей, використовуючи їх взаємодоповнюючі властивості для підвищення точності передбачень, покращення узагальнюваності та стійкості до шумів.

Кожна модальність несе унікальні ознаки: наприклад, зображення – просторову або структурну інформацію, текст – семантичний контекст, аудіо – часові характеристики. Їхнє поєднання дозволяє моделі формувати багатші та контекстно залежні уявлення, що особливо важливо у випадках високої невизначеності чи неоднозначності вхідних даних.

Вражаючий приклад такого підходу – передбачення функцій білків, де необхідно враховувати як послідовність амінокислот (лінійні 1D-дані), так і просторову (3D) структуру білка. Дослідження Qian та ін. [16] показало, що інтеграція послідовнісних ознак і структурних представлень (наприклад, графових описів складок білка) у межах мультимодальної архітектури суттєво покращує результати класифікації. Такі моделі краще враховують взаємозв’язок між формою та складом, що підвищує точність навіть для білків з маловідомою функцією або нових, раніше не досліджених послідовностей.

1.3. Наукова постановка задачі

Правильна постановка наукової задачі є критично важливим етапом будь-якого дослідження. В її рамках формується відповідь на питання про актуальність вирішуваної проблеми в межах різних предметних областей. Вона визначає напрямок роботи, методологію, очікувані результати та можливий внесок у науку. Також здійснюється аналіз можливих ризиків, які можуть впливати на якість та перебіг дослідження, для подальшого

визначення контрзаходів та стратегії контролю ризиками. Неправильне або неконкретне формулювання проблеми може призвести до неефективного використання ресурсів, відсутності значущих результатів або навіть до хибних висновків.

Після чіткого визначення проблематики дослідження проводиться докладний аналіз існуючих рішень, який включає основні принципи роботи, їх переваги та недоліки у використанні, сфери їх застосування. На основі цієї інформації формується висновок про актуальність проведення даного дослідження та розробки власного методу для вирішення визначеної проблеми.

Наступним етапом є власне розробка відповідного методу, який би враховував результати аналізу проведеного на попередніх етапах та запропонував би покращені результати у порівнянні з існуючими рішеннями. Тому ще одним важливим кроком є визначення відповідних метрик, за якими можна було порівняти згадані методи. Також необхідно зібрати необхідні для достатнього покриття проблематики набори даних, на основі яких відбуватиметься навчання та перевірка ефективності методу на основі визначених метрик.

На основі вищезгаданих аспектів постановки наукової задачі сформулюємо основні завдання з реалізації магістерського наукового дослідження:

1. Визначити тему наукового дослідження, провести логіко-структурний аналіз обраної проблематики.
2. Проаналізувати наявні методи та алгоритми, які вирішують розглянуту проблему у різних проявах, визначити їх переваги та недоліки, базуючись на результатах та принципах їх роботи.
3. Розробити власний метод для вирішення обраної проблеми, описати його теоретичні засади.
4. Виконати програмну реалізацію запропонованого методу.
5. Провести тестування розробленої системи.

6. Провести детальний аналіз розробленого методу ефективності у порівнянні з розглянутими існуючими рішеннями з використанням при різних сценаріях.

1.4. Висновки до першого розділу

Спираючись на результати проведеного логіко-структурного аналізу розглянутої проблеми, а саме прогнозування даних з мультимодальним розподілом, можемо зробити висновки про те, що її дослідження, а також розробка нових рішень для її вирішення є актуальними завданнями у контексті сучасних реалій.

Дане твердження базується на основі аналізу, в рамках якого було розглянуто, в загальних рисах, задачу класифікації та, більш детально, задачу прогнозування даних з мультимодальним розподілом як окремий випадок задачі класифікації. Було розглянуто типові приклади наборів даних, що мають мультимодальний розподіл, а також основні причини їх виникнення. Також було розглянуто приклади практичні приклади задач прогнозування даних з мультимодальним розподілом та певні особливості, які можуть виникати під час розв’язання таких задач у порівнянні із випадками, коли набори даних не мають більше однієї моди у своєму розподілі.

Було проведено детальний аналіз існуючих рішень розглянутої проблеми, враховуючи різні сценарії її виникнення. При цьому було розглянуто їх основні принципи роботи, основні переваги та недоліки використання у відповідних ситуаціях. На основі проведеного аналізу можемо зробити висновок про відсутність комплексного та універсального рішення щодо прогнозування даних з мультимодальним розподілом.

За результатами проведеного в рамках першого розділу аналізу було сформовано відповідні висновки, визначено постановку наукової задачі та основних кроків щодо виконання задач для проведення магістерського наукового дослідження.

2. РОЗРОБЛЕНИЙ АЛГОРИТМІЧНО-ПРОГРАМНИЙ МЕТОД ПРОГНОЗУВАННЯ МУЛЬТИМОДАЛЬНИХ ДАНИХ НА ОСНОВІ МОДЕЛІ MDN

MDN – це одна з найпопулярніших моделей ймовірнісних нейронних мереж, яка базується на використанні суміші гаусів для прогнозування ймовірності кожного можливого результату. Цей підхід дає можливість передбачити більше ніж один можливий результат, залежно від кількості Гаусових компонент (гіперпараметр MDN). У цьому дослідженні розглядається використання ймовірнісного підходу замість підходу подібності як модифікації MDN для підвищення точності навчання для прогнозування мультимодальних даних.

2.1. Опис базової моделі MDN

Насамперед розглянемо загальну структуру MDN та основні принципи її роботи. MDN базується на комбінованому використанні нейронної мережі з сумішшю моделей Гауса.

2.1.1. *Суміш гаусів*

Суміш гаусів (MoG, або Gaussian Mixture Model, GMM) – це ймовірнісна модель, яка використовується для представлення складних розподілів і даних. MoG моделює ймовірнісний розподіл даних як зважену суму кількох нормальних (гаусових) розподілів, кожен з яких описує локальну структуру або підгрупу в даних.

Функція щільності ймовірності суміші Гаусів (3) має вигляд:

$$p(y) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mu_k, \sigma_k^2), \quad (3)$$

де:

- K – кількість компонентів у суміші (тобто, кількість гаусових розподілів);
- π_k – ваговий коефіцієнт (ймовірність вибору компонента k), який задовольняє умову нормалізації (4):

$$\sum_{k=1}^{K} \pi_k = 1, \quad \pi_k(x) \geq 0. \tag{4}$$

- $\mathcal{N}(y|\mu_k, \sigma_k)$ – нормальний розподіл із математичним сподіванням μ_k і дисперсією σ_k, описаний формулою (5):

$$\mathcal{N}(y|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(y-\mu_k)^2}{2\sigma_k^2}}. \tag{5}$$

Таким чином, кожне значення y оцінюється як зважена сума ймовірностей, обчислених для кожного гаусового компонента. Принцип побудови ймовірнісного розподілу для MoG можемо спостерігати на рис. 4-6. Приклад функції щільності ймовірності, який складається з більшої кількості компонент зображено на рис. 7.

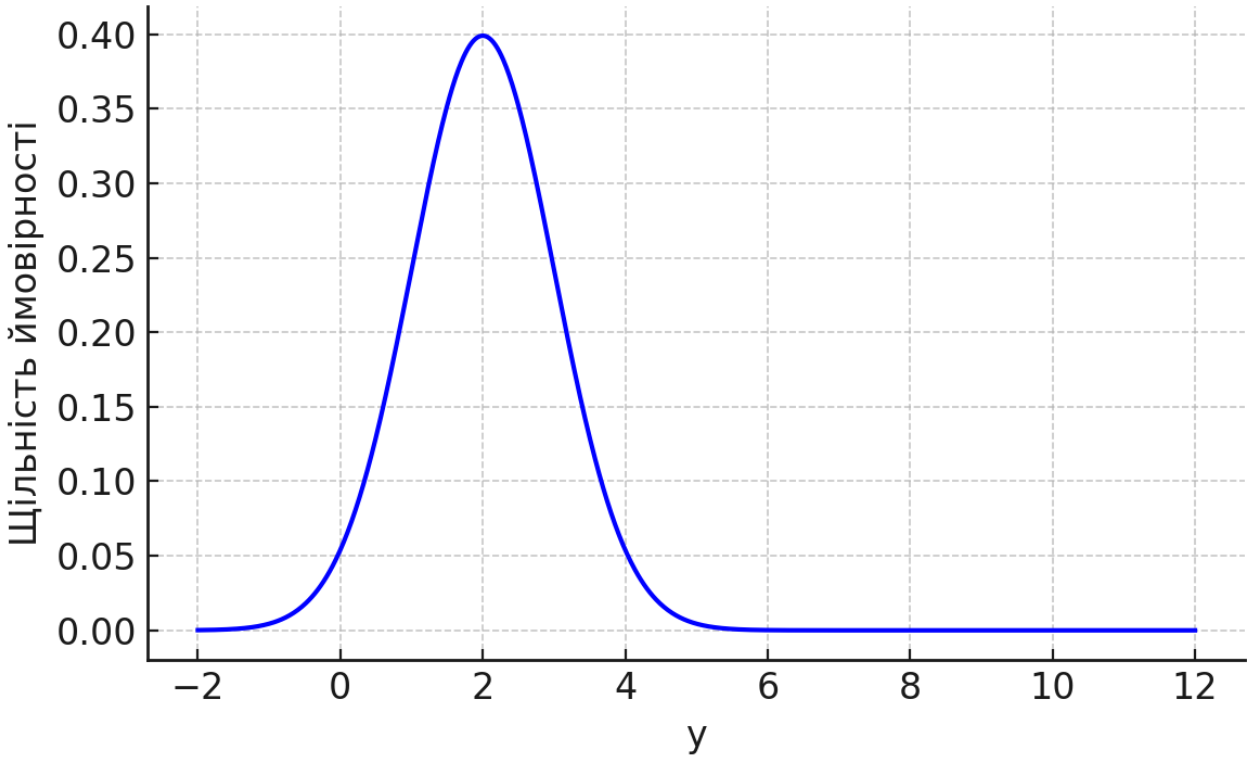


Рис. 4. Приклад побудови ймовірнісного розподілу для суміші гаусового розподілу

Перейдемо до розгляду принципу функціонування даної ймовірнісної моделі. Суміш Гаусів передбачає, що кожна точка даних y була згенерована одним із K гаусових розподілів, але ми не знаємо, який саме компонент її породив.

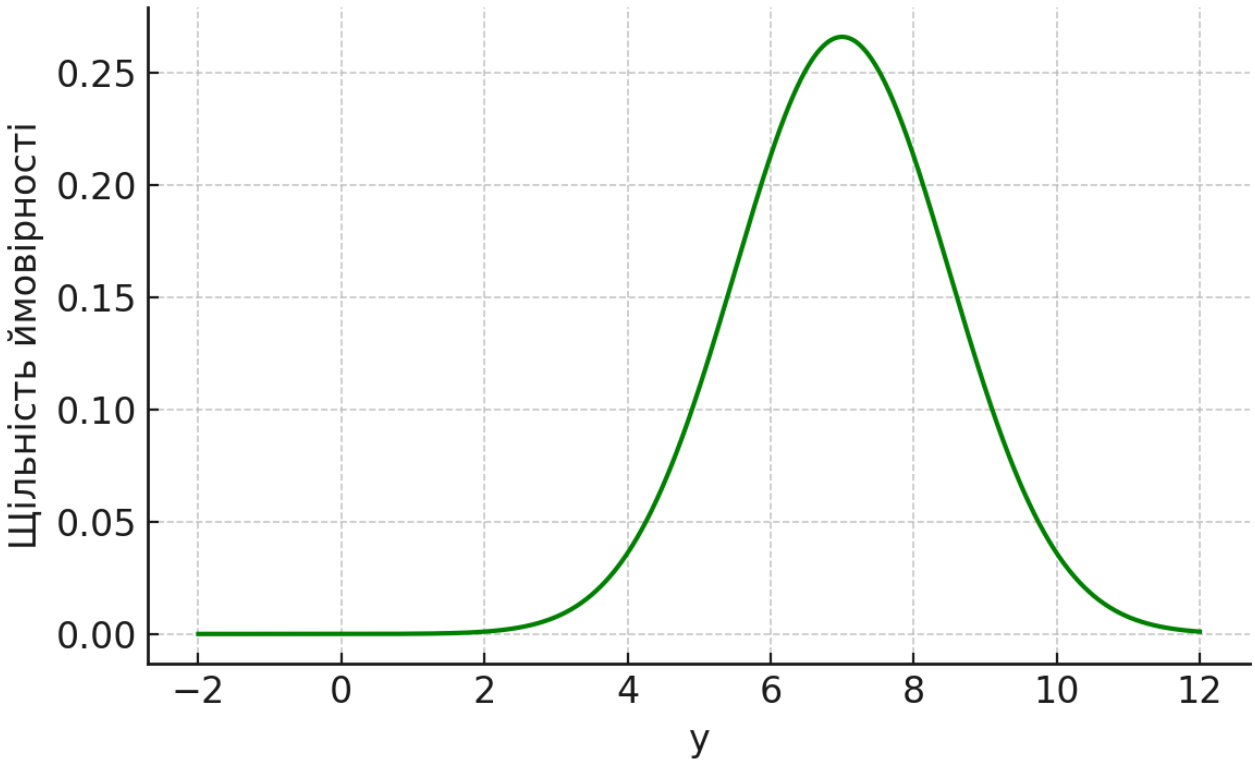


Рис. 5. Приклад побудови ймовірнісного розподілу для суміші гаусового розподілу

MoG моделює це приховане (латентне) групування шляхом введення прихованої змінної z, яка приймає значення від 1 до K та вказує на номер компонента, з якого походить точка.

У процесі навчання параметри π_k, μ_k, σ_k^2 (або коваріаційна матриця у багатовимірному випадку) оцінюються таким чином, щоб максимізувати правдоподібність спостережуваних даних. Це зазвичай реалізується за допомогою EM-алгоритму (алгоритму очікування-максимізації) [17], який поетапно оцінює ймовірність приналежності точок до кожного компонента та оновлює параметри компонентів.

Розглянемо основні властивості даної ймовірнісної моделі, які є особливо корисними у задачах мультимодального прогнозування:

- MoG добре апроксимує багатомодальні розподіли (тобто такі, що мають кілька локальних максимумів).

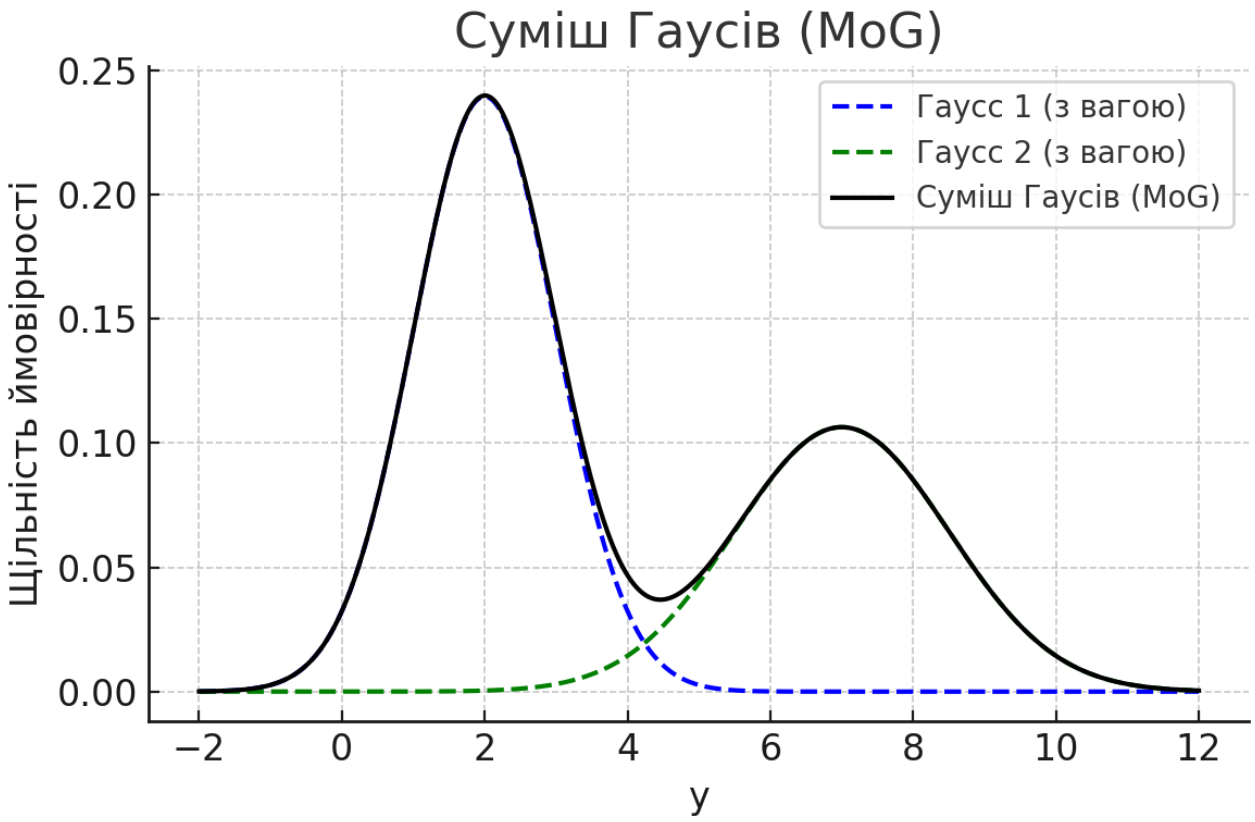


Рис. 6. Приклад побудови ймовірнісного розподілу для суміші гаусового розподілу, утвореного за рахунок зваженої суми двох гаусових компонент

- Завдяки гнучкості моделі вона здатна відображати складну структуру даних у таких завданнях, як кластеризація, відновлення даних, детекція аномалій, сегментація зображень тощо.
- У багатовимірному випадку гаусові розподіли мають вектор середніх та коваріаційну матрицю, що дозволяє враховувати кореляції між ознаками.

Враховуючи умову, виражену у формулі (4) функція щільності $p(y)$ (3) залишається валідним ймовірнісним розподілом, тобто її кумулятивна ймовірність по всій області визначення змінної y дорівнює 1. Це означає, що суміш Гаусів є повноцінною ймовірнісною моделлю, яку можна використовувати для оцінювання ймовірностей та генерації нових зразків.

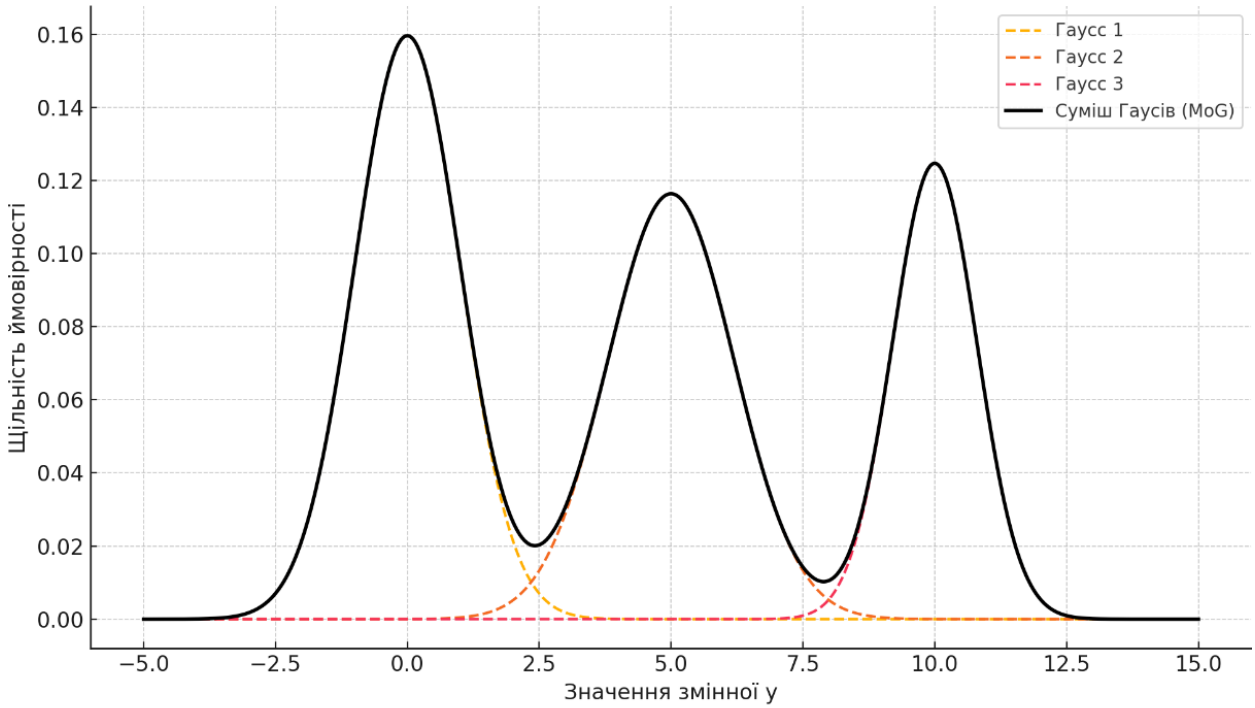


Рис. 7. Приклад графіку функції щільності ймовірності для суміші гаусового розподілу, який містить 3 компоненти

2.1.2. *Опис принципу роботи моделі MDN*

Mixture Density Network – це тип нейронної мережі, що об'єднує в собі потужність класичних нейронних мереж із гнучкістю ймовірнісного моделювання, яке надає суміш Гаусових розподілів. Головна ідея MDN полягає не у передбаченні одного детермінованого вихідного значення, а у моделюванні повного умовного розподілу вихідних даних y залежно від вхідних даних x.

Перейдемо до розгляду архітектури моделі MDN. Вона має загальну структуру, подібну до звичайної нейронної мережі:

- Вхідний шар (input layer) – отримує вхідні дані x, які можуть бути числовими ознаками, пікселями зображення, часовими рядами тощо.
- Приховані шари (hidden layers) – виконують обробку даних за допомогою нелінійних активацій (ReLU, sigmoid, tanh). Ці шари витягують ознаки, необхідні для представлення взаємозв'язку між x і умовним розподілом $p(y \mid x)$.

- Вихідний шар (output layer) – основна відмінність MDN від звичайної нейромережі. Замість одного числового виходу (як у регресії) або ймовірностей класів (як у класифікації), мережа генерує параметри суміші Гаусів.

Схематичне зображення архітектури моделі MDN можемо спостерігати на рис. 8.

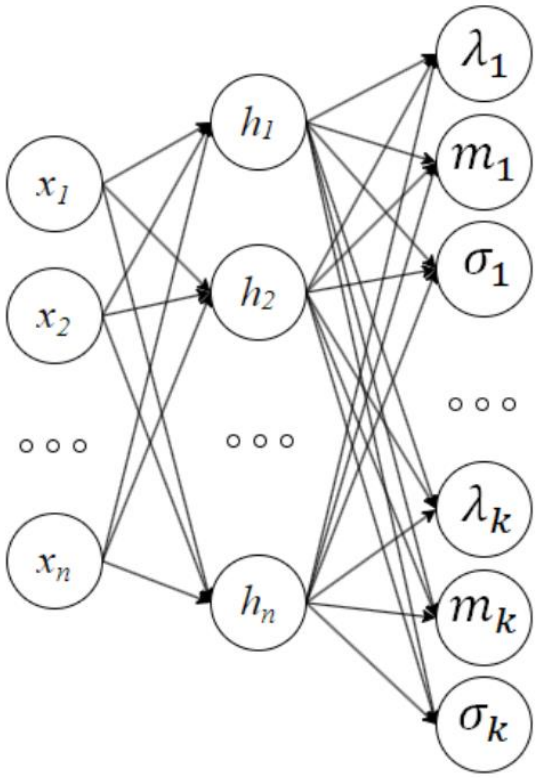


Рис. 8. Схематичне зображення архітектури моделі MDN для випадку мілкої нейронної мережі

Для кожного з K гаусових компонентів мережа прогнозує:

- $\mu_k(x)$ – середнє значення компонента (може бути вектором у багатовимірному випадку);
- $\sigma_k(x)$ – стандартне відхилення (або коваріаційна матриця);
- $\pi_k(x)$ – коефіцієнт змішування (тобто ймовірність вибору цього компонента), що задовольняє умову нормалізації (4).

Таким чином, вихід мережі – це повний набір параметрів, що визначають умовний розподіл $p(y \mid x)$ у вигляді суміші Гаусів (6).

$$p(y|x) = \sum_{k=1}^{K} \pi_k(x)\mathcal{N}(y|\mu_k(x), \sigma_k^2(x)). \quad (6)$$

MDN особливо корисна у випадках, коли залежність між вхідними та вихідними даними є багатозначною або неоднозначною, тобто коли існує кілька можливих результатів y для одного й того ж x.

Наприклад:

- передбачення наступного положення руки в жестовому керуванні;
- синтез мови (один текст – багато варіантів звучання);
- передбачення майбутнього руху транспортного засобу;
- моделювання ймовірності візуальних даних.

Кількість гаусових компонентів K є гіперпараметром, який задається до навчання та визначає гнучкість моделі. Занадто мала кількість може не описати складність розподілу, занадто велика – призводить до перенавчання.

MDN навчається шляхом мінімізації функції втрат, яка є від'ємною логарифмічною правдоподібністю (Negative Log-Likelihood, NLL). Це означає, що модель намагається максимізувати правдоподібність того, що дані y були згенеровані із передбачуваного умовного розподілу $p(y \mid x)$. Математично представити функцію втрат, якою користується модель MDN для навчання, можна у вигляді наступної формули (7):

$$L[\phi] = \sum_{i=1}^{I} -\log\left(\int_{y_i-\varepsilon}^{y_i+\varepsilon} \sum_{k=1}^{K} \pi_k(x_i, \phi)\mathcal{N}\left(y_i \middle| \mu_k(x_i, \phi), \sigma_k^2(x_i, \phi)\right) dy\right). \quad (7)$$

В даній формулі ϕ – параметри мережі, які відповідають за вихідні значення π_k, μ_k, σ_k^2.

Мінімізація цієї втрати спонукає мережу правильно моделювати форми розподілів, з урахуванням як багатомодальності, так і невизначеності у виходах.

Зазначимо певні особливості реалізації вихідного шару для даної моделі:

- Значення π_k формуються через *softmax*, щоб забезпечити нормалізацію (сума = 1).

- Значення σ_k^2 проходять через *exponential* або *softplus*, щоб гарантувати додатність дисперсії.
- Середнє μ_k не обмежується і може приймати довільні значення.

Серед основних переваг даного підходу до прогнозування даних можемо зазначити наступні:

- *Моделювання складних розподілів*. MDN дозволяє моделювати багатомодальні (множинні піки) розподіли, де традиційна регресія дає лише середнє значення.
- *Облік невизначеності*. Модель повертає повний умовний розподіл $p(y \mid x)$, що дозволяє врахувати розмитість, шум та неоднозначність у даних.
- *Гнучкість і виразність*. Комбінація нейронних мереж і ймовірнісних моделей дозволяє MDN апроксимувати майже будь-який тип залежності між вхідними та вихідними змінними.
- *Придатність для регресії з кількома відповідями*. Може працювати у випадках, коли для одного входу можливі кілька правильних виходів (наприклад, різні траєкторії в автономному русі).
- *Можливість генерації нових зразків*. Використовуючи передбачуваний розподіл, можна генерувати ймовірні вихідні значення, що корисно в генеративних задачах.

Щодо основних недоліків моделі MDN, то можемо сформувати наступний їх список:

- *Складність навчання*. Функція втрат (log-likelihood) може бути чутливою до нестабільності: неправильні значення дисперсії можуть викликати числові проблеми (наприклад, $log(0) \rightarrow \infty$).
- *Проблеми з масштабуванням*. У багатовимірному випадку потрібно передбачати не тільки середні й дисперсії, а ще й коваріаційні матриці, що значно ускладнює модель.

- *Інтерпретація результатів*. Аналіз передбачуваних параметрів (наприклад, яка компонента головна) не завжди інтуїтивно зрозумілий.
- *Чутливість до вибору кількості компонентів K*. Занадто мала кількість спричиняє неадекватне моделювання; занадто велика – сприяє перенавчанню та ускладненню.
- *Низька популярність у деяких сферах*. Через складність реалізації MDN іноді витісняються байєсівськими нейромережами чи dropout-емпіричними методами.

Відповідно до таких особливостей модель MDN може застосовуватися у різних сферах, зокрема:

Регресія з неоднозначністю та невизначеністю.

Приклад: прогнозування ринкових цін, розпізнавання жестикуляції, обробка кліматичних або біомедичних даних.

Чому MDN: у випадках, коли для одного й того ж набору вхідних характеристик може існувати кілька допустимих виходів (наприклад, різні ціни на схожі будинки), MDN не навчається на "середньому", як традиційна регресія, а виводить повний розподіл ймовірностей.

Прогнозування траєкторій у робототехніці й автономному русі

Приклад: передбачення наступного кроку пішохода, об’їзд перешкод дронами, рух автомобіля в умовах непередбачуваного середовища.

Чому MDN: модель дозволяє враховувати кілька потенційних траєкторій руху (напр. обігнати зліва чи справа), оцінюючи ймовірність кожного сценарію.

Генерація даних (Generative Modeling)

Приклад: генерація мелодій, синтез зображень, генерація мовлення (в системах TTS – text-to-speech).

Чому MDN: в генеративних завданнях важливо не просто "відтворити" середній шаблон, а створити різноманітні варіації відповідно до розподілу.

MDN дозволяє моделювати багатомодальні структури – тобто створювати різні правдоподібні варіанти одного запиту.

Зворотні задачі у фізиці, біології, техніці

Приклад: реконструкція джерела сигналу (наприклад, в сейсмології), визначення властивостей матеріалів за експериментальними спостереженнями.

Чому MDN: в зворотних задачах існує невизначеність між причинно-наслідковими зв'язками, і відповідь може бути неоднозначною – MDN якраз моделює такий випадок, повертаючи повний умовний розподіл можливих рішень.

Нерівнозначна класифікація / нечітке групування

Приклад: класифікація об’єктів на зображенні при недостатньо чітких даних, оцінка належності до декількох категорій (наприклад, зображення "кіт-собака").

Чому MDN: замість бінарної або softmax класифікації, модель може передбачити розподіл над кількома можливими класами з неоднозначністю (наприклад, коли об'єкт належить одразу до двох груп з певними ймовірностями).

Моделювання часового ряду з кількома сценаріями розвитку

Приклад: передбачення економічних показників, попиту, погоди, поведінки користувачів.

Чому MDN: на відміну від класичних підходів, MDN дозволяє врахувати кілька варіантів майбутнього розвитку з відповідними ймовірностями, що важливо в умовах невизначеності.

Доповнення до Reinforcement Learning (RL) та ймовірнісних агентів

Приклад: моделювання політики агента в середовищі з частковою або шумною інформацією.

Чому MDN: дозволяє агентам приймати рішення на основі ймовірнісної оцінки майбутніх наслідків, а не лише детермінованого значення нагороди.

У якості висновку можемо сказати, що MDN – це потужна модель, яка:

- не просто прогнозує окреме значення, а повертає розподіл ймовірностей для виходу;
- дозволяє враховувати невизначеність і неоднозначність у прогнозах;
- може застосовуватися у задачах регресії, класифікації з невизначеністю, генеративному моделюванні тощо.

2.2. Опис запропонованого методу на основі моделі MDN

У дослідженні акцентується увага на недоліках класичного підходу до моделювання втрат у задачах, де модель прогнозує ймовірність певного значення y.

2.2.1. *Передумови та мотивація для модифікації методу*

Зазвичай втрати оцінюються як від'ємний логарифм передбаченої щільності ймовірності в точці y_i. Проте в класичних реалізаціях щільність ймовірності $p(y_i)$ може бути більше 1, що математично допустимо (для щільностей), але при використанні даного значення у якості показника передбаченої ймовірності, можемо зіткнутися з певними проблемами:

- функція втрат стає від'ємною, що суперечить інтуїтивним уявленням про втрати як позитивну міру помилки;
- інтерпретація результатів навчання, базуючись на показниках функції втрат ускладнюється;
- потенційний критерій зупинки навчання розмивається, бо не існує чіткого «еталонного» нуля.

2.2.2. *Опис математичної моделі запропонованого підходу*

Замість оцінювання щільності ймовірності у точці**,** в даній роботі пропонується оцінювати ймовірність потрапляння у малий інтервал навколо точки y_i (8):

$$p(y_1, y_2) = \int_{y_1}^{y_2} \sum_{k=1}^{K} \pi_k(x_i, \phi) \mathcal{N}(y_i | \mu_k(x_i, \phi), \sigma_k^2(x_i, \phi))\, dy, \qquad (8)$$

де $y_1 = y_i - \varepsilon$, $y_2 = y_i + \varepsilon$.

Відповідно функція втрат модифікується на наступну (9):

$$L[\phi] = \sum_{i=1}^{I} -\log\left(\int_{y_i-\varepsilon}^{y_i+\varepsilon} \sum_{k=1}^{K} \pi_k(x_i,\phi)\mathcal{N}\left(y_i \middle| m_k(x_i,\phi), \sigma_k^2(x_i,\phi)\right) dy\right), \quad (9)$$

де:

- ε – гіперпараметр, що визначає ширину інтервалу навколо кожного спостереження y_i;
- ϕ – параметри моделі (нейронної мережі);
- K – кількість гаусових компонентів.

Коротко описуючи суть зміни у підході до побудови моделі MDN у порівнянні з класичним, можемо вказати основні пункти, спочатку для стандартного підходу:

- Втрата = $-\log\big(p(y_i)\big)$.
- Використовується значення щільності в точці y_i, яке може бути більше 1, що може призвести до від'ємних значень функції втрат.

Тоді як у запропонованому методі:

- Втрата = $-\log\big(p(y_i \in [y_i - \varepsilon, y_i + \varepsilon])\big)$.
- Оцінюється ймовірність попадання у вузький інтервал – завжди значення знаходитиметься в межах від 0 до 1, а тому логарифм завжди менше або рівне 0, а отже значення функції втрат завжди більше 0.

2.2.3. *Наслідки застосування запропонованого методу*

Розглянемо основні аналітичні та практичні наслідки застосування описаного підходу:

1. *Краща інтерпретація втрат*. Логіка втрат стає прозорішою – мінімальна втрата показує високу ймовірність попадання у вузький інтервал, а отже і високу точність передбачення.
2. *Узгодження з ймовірнісною природою MoG*. Гаусова модель завжди має нульову ймовірність у точці (щільність в даному випадку не є

тотожним поняттям до ймовірності), тому інтегрування більш фізично обґрунтоване.

3. *Контроль рівня точності через ε*. Гіперпараметр ε дозволяє гнучко контролювати «допустиму похибку» моделі. Мале ε – жорстка оцінка, велике – м'яка.
4. *Вартість обчислень*. Недолік: для кожного спостереження потрібно обчислювати інтеграл суміші Гаусів у заданому інтервалі, що призводить до більших обчислювальні витрат під час тренування. Перевага: після навчання, модель працює так само швидко, оскільки модифікація стосується лише етапу навчання.

2.2.4. *Висновки щодо запропонованого методу*

Авторський метод:

- Вирішує низку проблем класичних втрат у MDN.
- Робить модель більш інтерпретовною та стабільною.
- Залишає прогнозну потужність незмінною.
- Вимагає більше ресурсів при навчанні, але не у використанні.
- Є загальним підходом, який можна застосувати і до інших ймовірнісних моделей.

2.3. Аналіз засобів наближеного обчислення визначеного інтегралу

Підбір ефективного методу обчислення визначеного інтегралу на проміжку для суміші гаусових розподілів є критично важливим у запропонованій модифікації MDN з кількох причин:

1. *Обчислення втрат залежить від інтегралу*. У модифікованій функції втрат (9) необхідно обчислити ймовірність, що випадкова величина з суміші нормальних розподілів потрапить у вузький інтервал. Це означає, що потрібно інтегрувати зважену суму густин (8).

2. Точність і стабільність. Оскільки цей інтеграл прямо впливає на значення функції втрат, неточність або нестабільність обчислення може призвести до: неправильного оновлення ваг у процесі навчання, поганого узгодження моделі з даними та до складнощів у мінімізації втрат.
3. *Швидкодія*. Навчання MDN вимагає масового обчислення інтегралів для кожного зразка на кожній ітерації. Повільні або надто складні методи інтегрування можуть зробити навчання вкрай неефективним або навіть непридатним для практичного використання.

Отже, ефективне обчислення визначених інтегралів для сумішей нормальних розподілів є ключем до точного, стабільного та швидкого навчання модифікованої MDN-моделі. Перейдемо до аналізу конкретних чисельних та стохастичних методів обчислення визначеного інтегралу для суміші гаусових ймовірностей

2.3.1. *Метод прямокутників*

Метод прямокутників – це найпростіший чисельний метод для наближеного обчислення визначеного інтегралу. Його суть полягає в апроксимації площі під графіком функції за допомогою прямокутників. Запишемо алгоритмічну суть методу.

Нехай необхідно обрахувати визначений інтеграл функції $f(x)$ на проміжку $[a, b]$, формулу для якого можемо записати наступним чином (10):

$$\int_a^b f(x)dx. \tag{10}$$

Для цього мусимо виконати наступні кроки:

1. Розбиваємо відрізок $[a, b]$ на n рівних частин шириною $h = \frac{b-a}{n}$.
2. Для кожного підінтервалу будуємо прямокутник, в залежності від типу методу використовуючи відповідні формули:
   Ліві прямокутники: висота = $f(x_i)$, де $x_i = a + i \cdot h$.
   Праві прямокутники: висота = $f(x_{i+1})$.

Середні прямокутники: висота = $f\left(\frac{x_i+x_{i+1}}{2}\right)$.

3. Площа кожного прямокутника: $h \cdot f(x_i)$.
4. Підсумовуємо площі всіх прямокутників.

Враховуючи особливості даного алгоритму, побудуємо візуальне представлення принципів роботи для кожного типу даного методу (рис. 9-11).

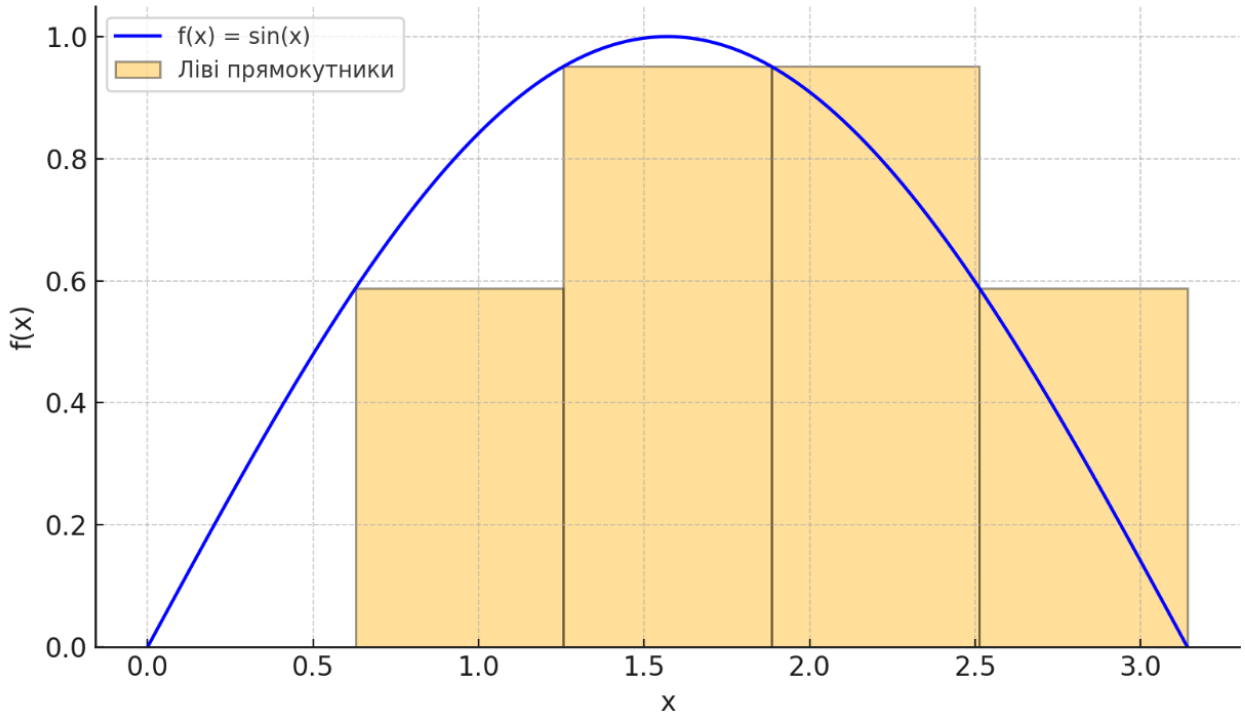


Рис. 9. Графічне представлення результатів обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку від 0 до π методом лівих прямокутників

На графіку (рис. 9) видно, як функція $\sin(x)$ апроксимується за допомогою лівих прямокутників – кожен прямокутник використовує значення функції на початку підінтервалу. Через це кожен прямокутник недооцінює площу під кривою, що характерно для зростаючих функцій. У випадку правих прямокутників (рис. 10) кожен прямокутник використовує значення функції в кінці підінтервалу – для зростаючої функції, як $\sin(x)$, це призводить до переоцінки інтегралу.

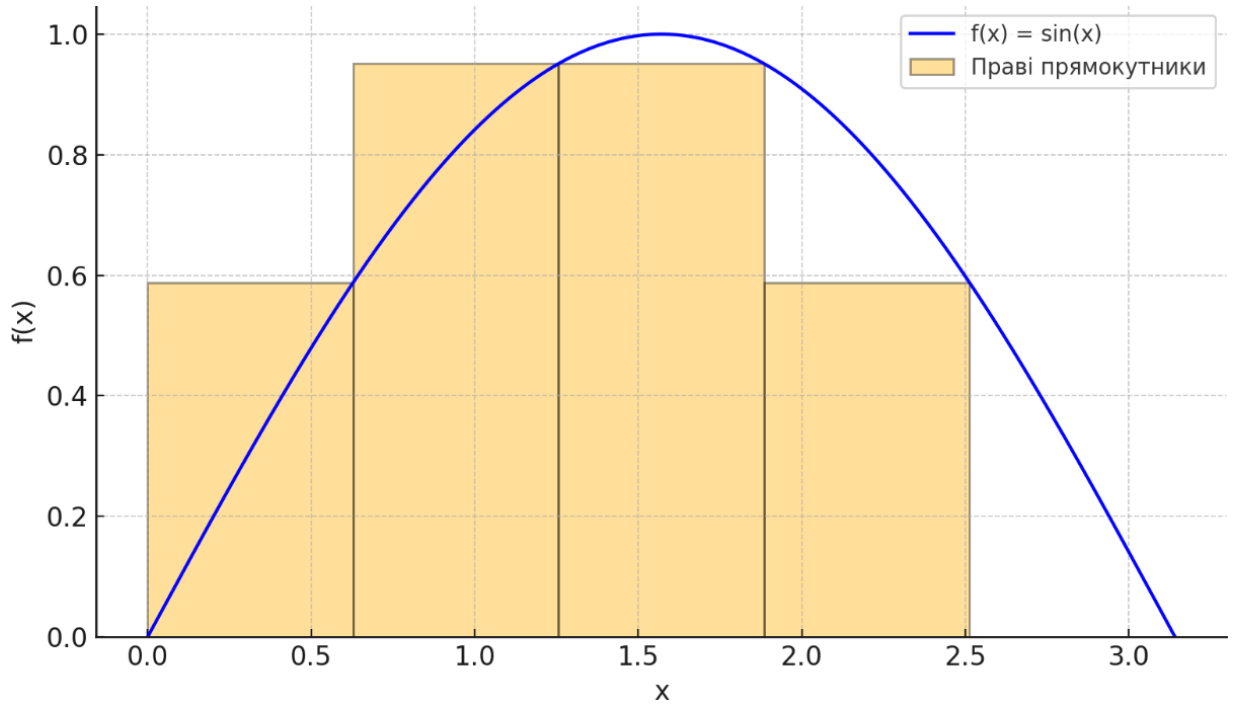


Рис. 10. Графічне представлення обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ методом правих прямокутників

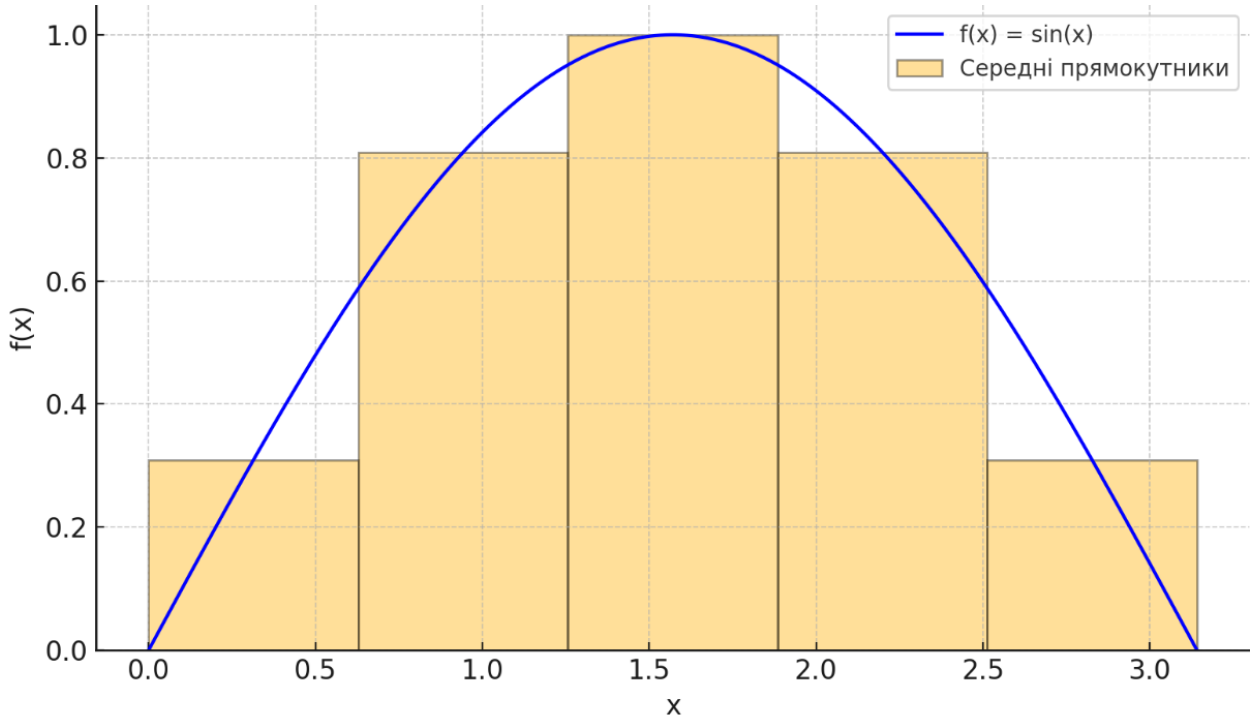


Рис. 11. Графічне представлення обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ методом середніх прямокутників

Для випадку із використанням середніх прямокутників (рис. 11) можемо сказати, що висота кожного прямокутника визначається значенням

функції в центрі підінтервалу та зазвичай дає найточніше наближення серед трьох методів.

Можемо назвати наступні переваги даного методу:

- *Простота реалізації*. Легко програмується навіть вручну.
- *Швидке виконання*. Мінімальні обчислення, особливо при малих n.
- *Гнучкість.* Можливість використання різних варіантів (ліві, праві, середні точки).
- *Початковий підхід*. Зручний для навчання чисельного інтегрування.

В той же час даний метод має досить суттєві недоліки у випадку впровадження у запропонований метод, зокрема:

- *Низька точність*. особливо для функцій зі зміною кривини.
- *Залежність від вибору точки*. ліві й праві прямокутники можуть сильно недооцінювати або переоцінювати значення інтегралу.
- *Погана збіжність*. потрібно багато кроків (n) для прийнятної точності.
- *Грубе наближення*. не враховує зміну функції між точками – лише значення в одній точці.

2.3.2. *Метод трапецій*

Метод трапецій – це чисельний метод обчислення визначеного інтегралу, який апроксимує підінтегральну функцію на кожному підінтервалі відрізка прямою лінією, тобто замість прямокутників будується трапеція, що забезпечує більш точне наближення, ніж метод прямокутників. Запишемо алгоритмічну суть методу.

Нехай маємо задачу, аналогічну до попередньої, тобто необхідно обрахувати визначений інтеграл функції $f(x)$ на проміжку $[a, b]$ (10). Для цього мусимо виконати наступні кроки:

1. Розбиваємо інтервал $[a, b]$ на n рівних частин з кроком $h = \frac{b-a}{n}$.

2. На кожному підінтервалі $[x_i, x_{i+1}]$ функція $f(x)$ апроксимується відрізком прямої, що з'єднує точки $\big(x_i, f(x_i)\big)$ і $\big(x_{i+1}, f(x_{i+1})\big)$.
3. Площа під функцією на цьому підінтервалі оцінюється як площа трапеції (11):

$$\text{площа} = \frac{h}{2} \cdot \big(f(x_i) + f(x_{i+1})\big). \tag{11}$$

4. Підсумовуємо площі всіх трапецій.

Відповідно, для такого методу результат пошуку значення визначеного інтегралу на проміжку $[a, b]$ для функції $f(x)$ можемо представити графічно (рис. 12).

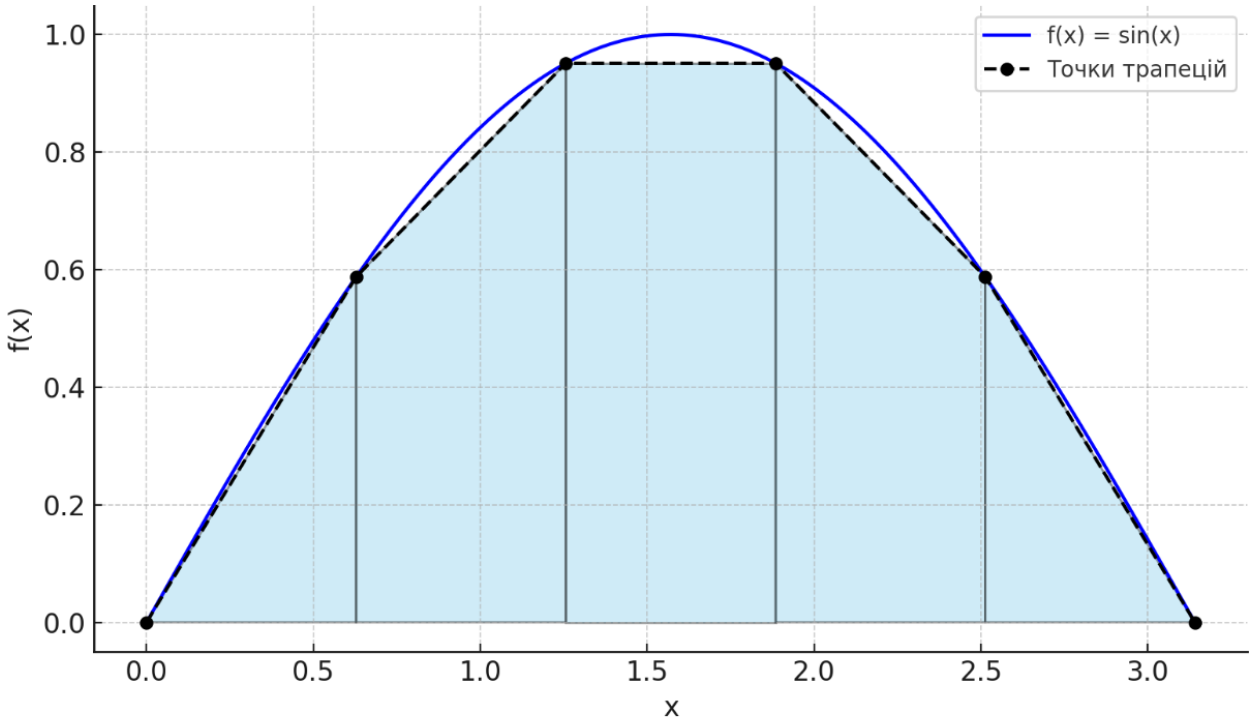


Рис. 12. Графічне представлення результатів обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ методом трапецій

На графіку видно, як метод трапецій апроксимує функцію $\sin(x)$ на кожному підінтервалі прямою лінією, утворюючи фігуру трапеції. Це забезпечує кращу оцінку площі під кривою порівняно з прямокутниками, особливо при гладких функціях.

До основних переваг даного методу можемо віднести наступні:

- Більш точний, ніж метод прямокутників.
- Простий у реалізації.
- Швидко збігається при гладких функціях.

Розглядаючи основні недоліки даного методу, можемо вказати наступний їх перелік:

- Неточний для функцій з високою кривиною на малих інтервалах.
- Може давати помилку порядку $O(h^2)$, тобто точність обмежена.

2.3.3. *Метод Сімпсона (метод парабол)*

Метод Сімпсона – це більш точний чисельний метод обчислення визначених інтегралів, ніж метод прямокутників чи трапецій. Він апроксимує підінтегральну функцію параболою, що проходить через три точки [18]. Запишемо алгоритмічну суть методу.

Нехай маємо задачу, аналогічну до попередньої, тобто необхідно обрахувати визначений інтеграл функції $f(x)$ на проміжку $[a, b]$ (10). Для цього метод Сімпсона розбиває інтервал $[a, b]$ на парну кількість підінтервалів (n), і на кожному парному відрізку $[x_i,\ x_{i+2}]$ будується парабола, що проходить через:

- x_i (ліва точка);
- x_{i+1} (середина);
- x_{i+2} (права точка).

Відповідно, для такого методу результат пошуку значення визначеного інтегралу на проміжку $[a, b]$ для функції $f(x)$ можемо представити графічно (рис. 13).

Переваги даного методу:

- Висока точність: помилка порядку $O(h^4)$, значно точніше за метод трапецій.
- Добре працює для гладких функцій.

- Менша кількість кроків для тієї ж точності, ніж у попередніх методів.

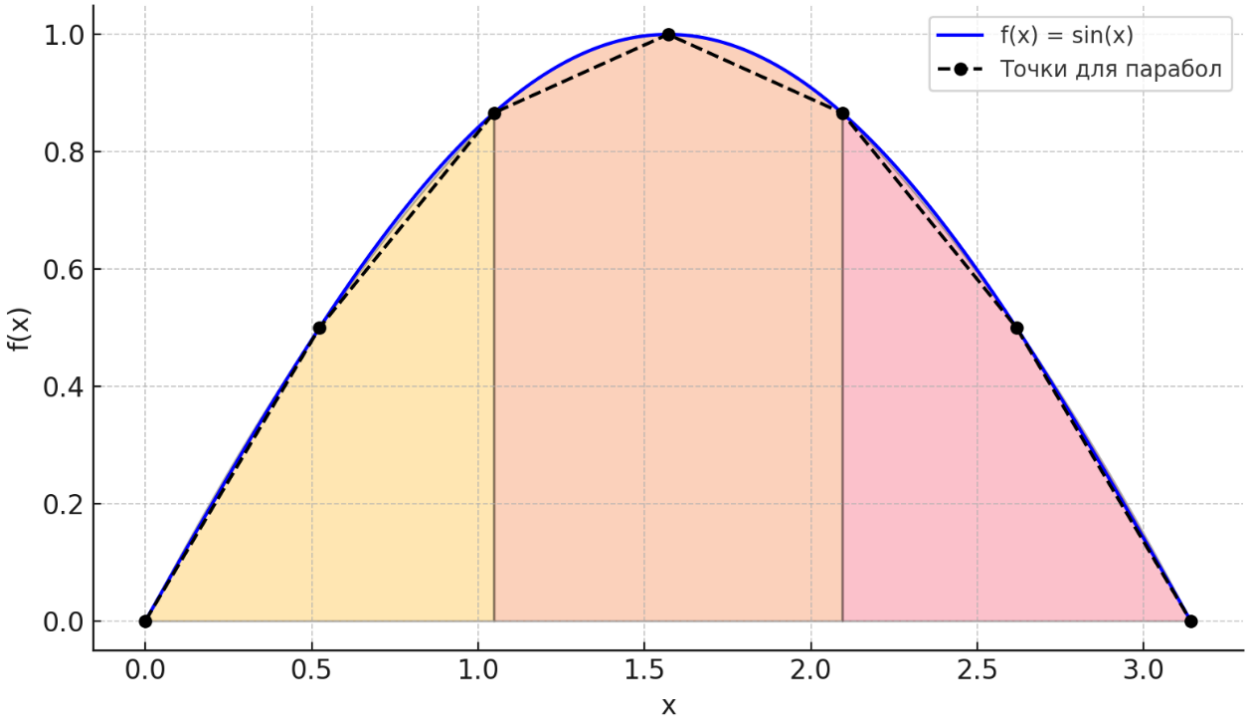


Рис. 13. Графічне представлення результатів обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ методом Сімпсона

До недоліків даного методу можемо віднести:

- Потрібно парне число підінтервалів.
- Складніший в реалізації, ніж прямокутники чи трапеції.
- Не працює добре для функцій з різкими змінами або розривами.

2.3.4. *Метод Монте-Карло (класичний)*

Метод Монте-Карло – це стохастичний (випадковий) метод чисельного інтегрування, який базується на генерації випадкових точок і оцінці середнього значення функції на основі цих точок [19]. Запишемо алгоритмічну суть методу.

Нехай маємо задачу, аналогічну до попередньої, тобто необхідно обрахувати визначений інтеграл функції $f(x)$ на проміжку $[a, b]$ (10).

Для цього мусимо виконати наступні кроки:

1. Генеруємо N випадкових точок $x_i \in [a, b]$, рівномірно розподілених на проміжку.
2. Обчислюємо значення функції в цих точках: $f(x_1, f(x_2), \dots, f(x_N)$.
3. Обчислюємо середнє значення функції (12):

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i). \quad (12)$$

4. Оцінка інтегралу (13):

$$\int_a^b f(x)dx \approx (b-a) \cdot \bar{f}. \quad (13)$$

Відповідно, для такого методу результат пошуку значення визначеного інтегралу на проміжку $[a, b]$ для функції $f(x)$ можемо представити графічно (рис. 14).

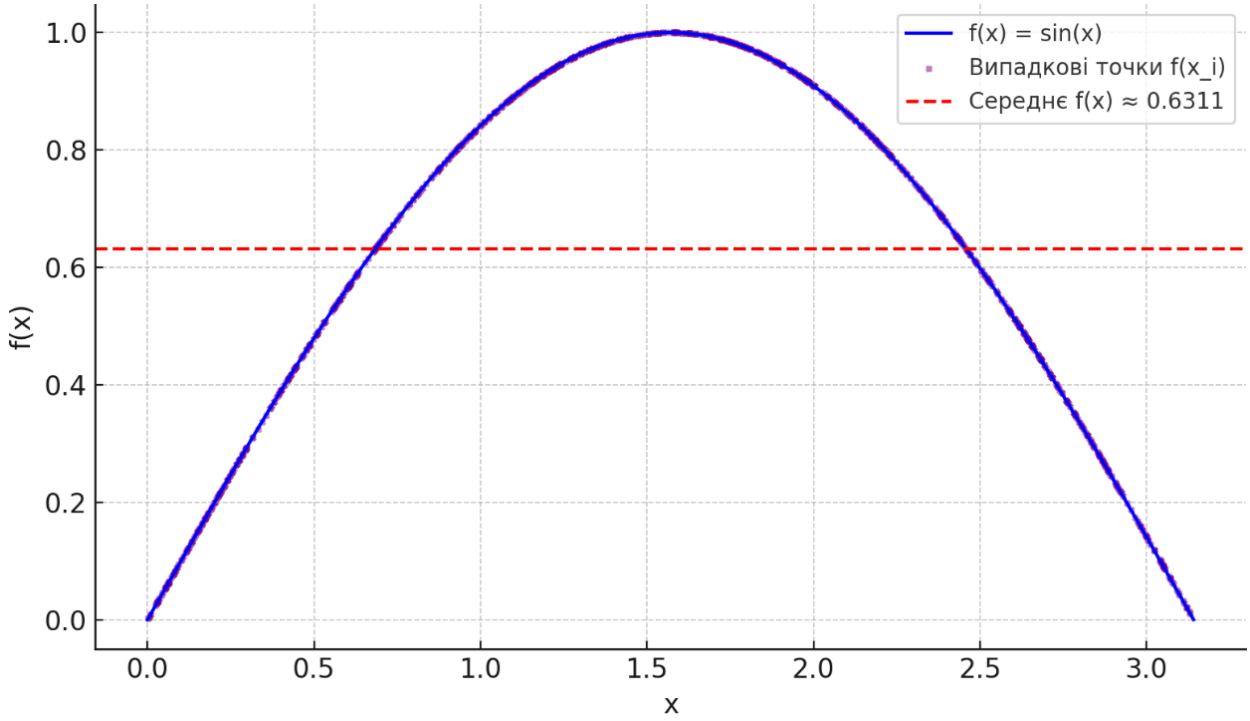


Рис. 14. Графічне представлення результатів обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ класичним методом Монте-Карло

На даному графіку зображено класичний метод Монте-Карло:

- Пурпурові точки – значення функції $\sin(x)$ у випадкових $x_i \in [0, \pi]$.

- Червона горизонтальна лінія – середнє значення $f(x_i)$, яке використовується для оцінки інтегралу.
- Оцінка інтегралу: добуток ширини інтервалу $(\pi - 0)$ на середнє значення функції.

До переваг даного методу можемо зарахувати:

- Добре працює для високовимірних задач (багатовимірні інтеграли).
- Не вимагає гладкості функції або регулярності сітки.
- Проста реалізація.

Проте даний метод має і певні недоліки:

- Повільна збіжність: помилка зменшується як $1/\sqrt{N}$.
- Не дуже ефективний для одновимірних інтегралів, де класичні методи працюють краще.

2.3.5. *Метод Монте-Карло (геометричний)*

Геометричний метод Монте-Карло – це варіант класичного методу, який дозволяє оцінити площу під кривою функції шляхом генерації випадкових точок у прямокутнику, що охоплює функцію [19]. Запишемо алгоритмічну суть методу.

Нехай маємо задачу, аналогічну до попередньої, тобто необхідно обрахувати визначений інтеграл функції $f(x)$ на проміжку $[a, b]$ (10). Для цього мусимо виконати наступні кроки:

1. Оберемо прямокутник з наступними розмірамим: по осі x від a до b; по осі y: від 0 до f_{max} (максимальне значення функції на інтервалі).
2. Згенеруємо N випадкових точок (x_i, y_i).
3. Перевіримо, які з точок потрапили під графік функції: Тобто, для яких $y_i \leq f(x_i)$.
4. Обчислимо частку точок під кривою (14):

$$\text{Частка} = \frac{\text{Кількість точок під кривою}}{N}. \quad (14)$$

5. Оцінимо площу під кривою (15):

$$\int_a^b f(x)dx \approx (b-a) \cdot f_{max} \cdot \text{Частка}. \quad (15)$$

Відповідно, для такого методу результат пошуку значення визначеного інтегралу на проміжку $[a, b]$ для функції $f(x)$ можемо представити графічно (рис. 15).

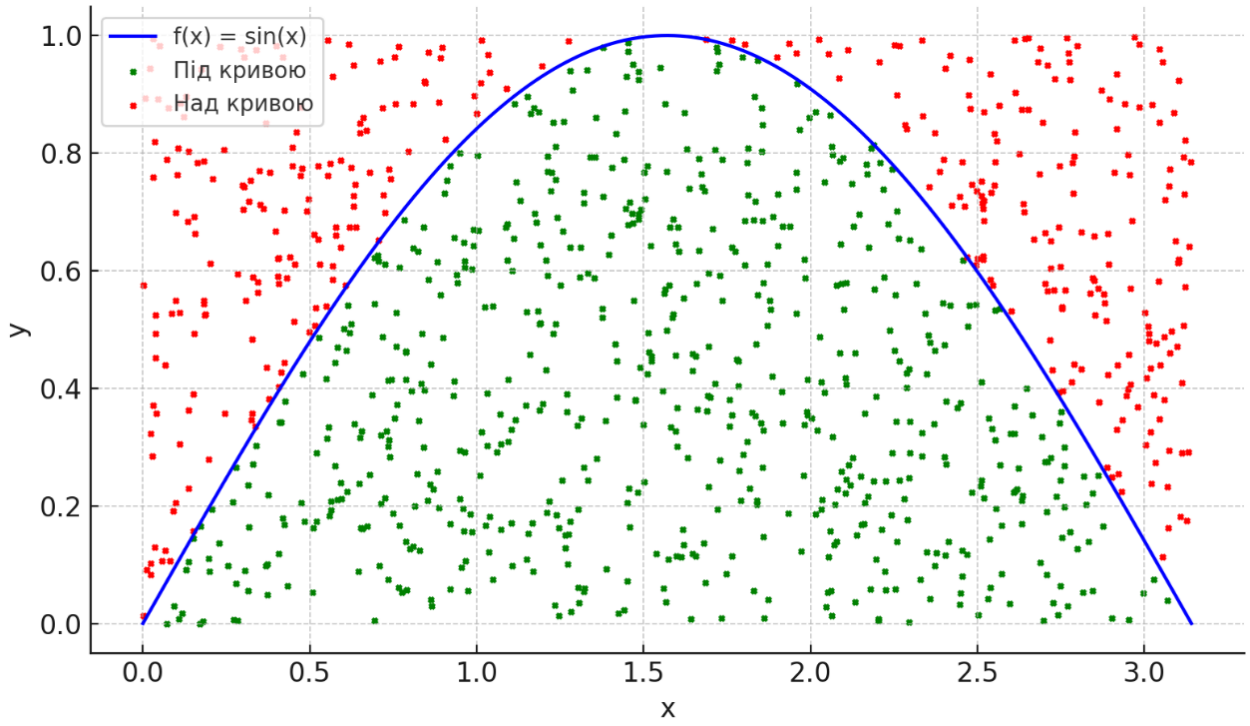


Рис. 15. Графічне представлення результатів обчислення визначеного інтегралу для функції $f(x) = \sin(x)$ на проміжку $[0, \pi]$ геометричним методом Монте-Карло

На графіку видно випадкові точки:

- Зелені – потрапляють під криву $\sin(x)$, а тому враховуються в оцінці інтегралу.
- Червоні – потрапляють над кривою, а тому відкидаються.

Чим більше точок N, тим точніше буде оцінка площі під кривою.

Визначимо основні переваги методу:

- Простий та наочний – особливо зручний для демонстрацій та навчання.

- Працює навіть тоді, коли неможливо аналітично обчислити інтеграл.
- Універсальний – може застосовуватись до будь-якої обмеженої фігури.

Також вкажемо і на певні недоліки даного методу:

- Повільна збіжність: точність зростає як $1/\sqrt{N}$.
- Неекономний у 1D: класичний варіант ефективніший.
- Залежить від вибору f_{max}: якщо обрано занадто великий прямокутник, то точність падає (більше «холостих» точок).

2.3.6. *Аналітичний підхід до обрахунку інтегралу*

Аналітичне обчислення визначеного інтегралу для суміші гаусових розподілів є ефективним та точним методом, який широко використовується у задачах, пов'язаних із MDN, ймовірнісною регресією, а також у статистичному аналізі.

Розглянемо застосування такого підходу безпосередньо у контексті нашого завдання, а саме обрахунку визначеного інтегралу для суміші гаусових розподілів. Тоді, нехай маємо суміш з K одновимірних нормальних розподілі $p(y)$ (3), для яких потрібно обрахувати визначений інтеграл (16).

$$\int_a^b p(y)dy. \tag{16}$$

Оскільки інтеграл від суміші рівний сумі інтегралів окремих компонент, відповідно до властивості лінійності інтегралу, маємо наступну математичну тотожність (17):

$$\int_a^b p(y)dy = \sum_{k=1}^{K} \pi_k \int_a^b \mathcal{N}(y|\mu_k, \sigma_k^2)dy. \tag{17}$$

Враховуючи, що інтеграл від нормального розподілу на інтервалі обчислюється через кумулятивну функцію розподілу (CDF) (18):

$$\int_a^b \mathcal{N}(y|\mu_k, \sigma_k^2)dy = \Phi\left(\frac{b-\mu_k}{\sigma_k}\right) - \Phi\left(\frac{a-\mu_k}{\sigma_k}\right), \tag{18}$$

де $\Phi(z)$ – CDF нормального розподілу, то можемо записати остаточну формулу аналітичного обчислення визначеного інтегралу суміші гаусових розподілів на проміжку (19):

$$\int_a^b p(y)dy = \sum_{k=1}^{K} \pi_k \left[\Phi\left(\frac{b-\mu_k}{\sigma_k}\right) - \Phi\left(\frac{a-\mu_k}{\sigma_k}\right)\right]. \quad (19)$$

Серед основних переваг аналітичного підходу можемо визначити наступні:

- *Точність*: обчислення не потребує апроксимацій.
- *Швидкість*: CDF для нормального розподілу доступна у всіх основних бібліотеках (наприклад, scipy.stats.norm.cdf).
- *Стабільність*: уникнення похибок, пов'язаних з чисельним інтегруванням або дискретизацією.

2.3.7. *Висновки щодо вибору методу для обчислення визначеного інтегралу*

На основі проведеного аналізу можна зробити висновок, що аналітичний метод обчислення визначеного інтегралу на проміжку для суміші гаусових розподілів є найбільш доцільним у контексті модифікованої MDN-моделі. Його використання дозволяє точно, швидко та стабільно обчислювати ймовірність потрапляння значення у вузький інтервал, що безпосередньо впливає на обчислення функції втрат. На відміну від чисельних методів, які потребують дискретизації та можуть створювати похибки, аналітичне інтегрування через кумулятивну функцію розподілу (CDF) стандартного нормального розподілу забезпечує високу обчислювальну ефективність та узгодженість результатів.

Чисельні методи, такі як метод прямокутників, метод трапецій і метод Сімпсона, також можуть бути використані для наближеного обчислення інтегралу, однак вони суттєво поступаються аналітичному підходу в точності та ефективності. Метод прямокутників є найпростішим, проте надає грубу оцінку площі, яка чутлива до вибору точки всередині підінтервалу. Метод трапецій покращує точність за рахунок лінійної апроксимації функції на

кожному кроці, тоді як метод Сімпсона досягає ще вищої точності, використовуючи квадратичну апроксимацію. Попри це, всі ці методи потребують достатньої кількості підінтервалів для забезпечення адекватної точності, що збільшує обчислювальні витрати, особливо у великих моделях.

Стохастичний підхід, представлений методом Монте-Карло, хоч і універсальний, демонструє найгіршу продуктивність у випадку одновимірної суміші нормальних розподілів. Його збіжність є повільною, а оцінки ймовірностей – нестабільними при малих розмірах інтервалу, що критично для модифікованої функції втрат MDN, де інтеграл обчислюється на дуже вузьких ділянках. Через це метод Монте-Карло непридатний для використання в таких задачах як основний засіб обчислення інтегралу.

Отже, враховуючи як точність, так і обчислювальну ефективність, саме аналітичне обчислення інтегралу через функцію розподілу нормального закону є найкращим вибором для використання в задачах моделювання на основі MDN, зокрема при впровадженні модифікованих функцій втрат, орієнтованих на ймовірність потрапляння у вузький інтервал.

2.4. Аналіз методів інтерпретації результуючого розподілу

Підбір ефективного методу для інтерпретації результатів модифікованої MDN-моделі – зокрема для пошуку мод (локальних максимумів) змішаного гаусового розподілу – є ключовим етапом аналізу вихідних даних, особливо в задачах, де важлива невизначеність або множинність можливих відповідей.

Модель MDN не просто передбачає одне значення – вона повертає ймовірнісний розподіл, зазвичай у вигляді суміші Гаусів.

У задачах з мультимодальністю (тобто коли можливі кілька коректних відповідей), знаходження мод (піків розподілу):

- дає інтерпретовану множину найбільш ймовірних відповідей;
- дозволяє розрізнити альтернативні сценарії (наприклад, кілька можливих траєкторій в задачах передбачення);

- покращує пояснюваність моделі – користувач бачить не лише «очікуване значення», а й варіанти.

Змішаний розподіл $p(y)$ є непростою функцією (8): це не одна гуасіана, а суперпозиція кількох, яка не має замкненої аналітичної формули для максимумів. Тому:

- Аналітичний підхід не працює. Неможливо взяти похідну й розв'язати $p'(y) = 0$ в явному вигляді.
- Прості методи можуть пропустити моди. Наприклад, просто брати μ_k, де π_k найбільше – не гарантує, що це буде глобальний максимум (рис. 16). Також часто справжні моди лежать не в центрах компонент (рис. 17).
- Потрібні чисельні або гібридні підходи. Густе семплювання + локальний пошук (наприклад, mean-shift, gradient ascent, або EM-подібні підходи); адаптивні сітки, які дозволяють відокремити моди, навіть якщо вони близькі чи мають перекриття.

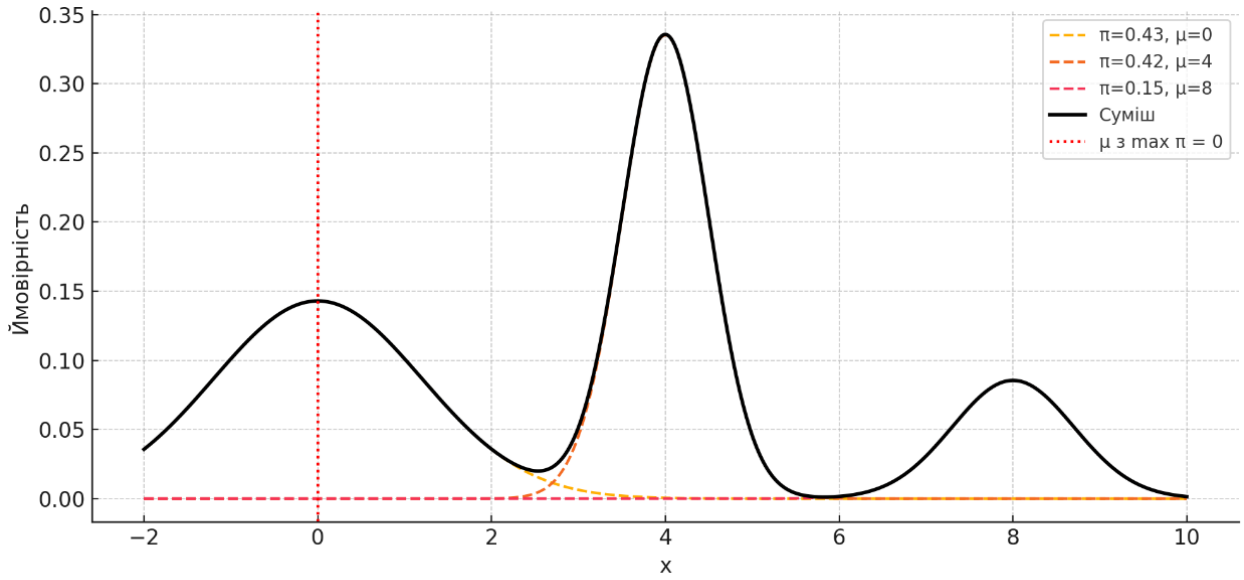


Рис. 16. Приклад, коли значення μ гаусової компоненти з найбільшим значенням не збігається з глобальним максимумом

На графіку (рис. 16) можемо спостерігати ситуацію, коли компоненти з центрами в $\mu = 0$ та $\mu = 4$ мають дуже схожі ваги: $\pi = 0.43$ та $\pi = 0.42$

відповідно однак через те, що друга компонента значно вужча ($\sigma = 0.5$), вона створює вищу локальну щільність. В результаті, глобальний максимум суміші знову зміщується вбік вузької компоненти, хоча вона не має найбільшої ваги. Це ще раз підтверджує: максимальна вага компоненти не гарантує, що її центр є модою розподілу.

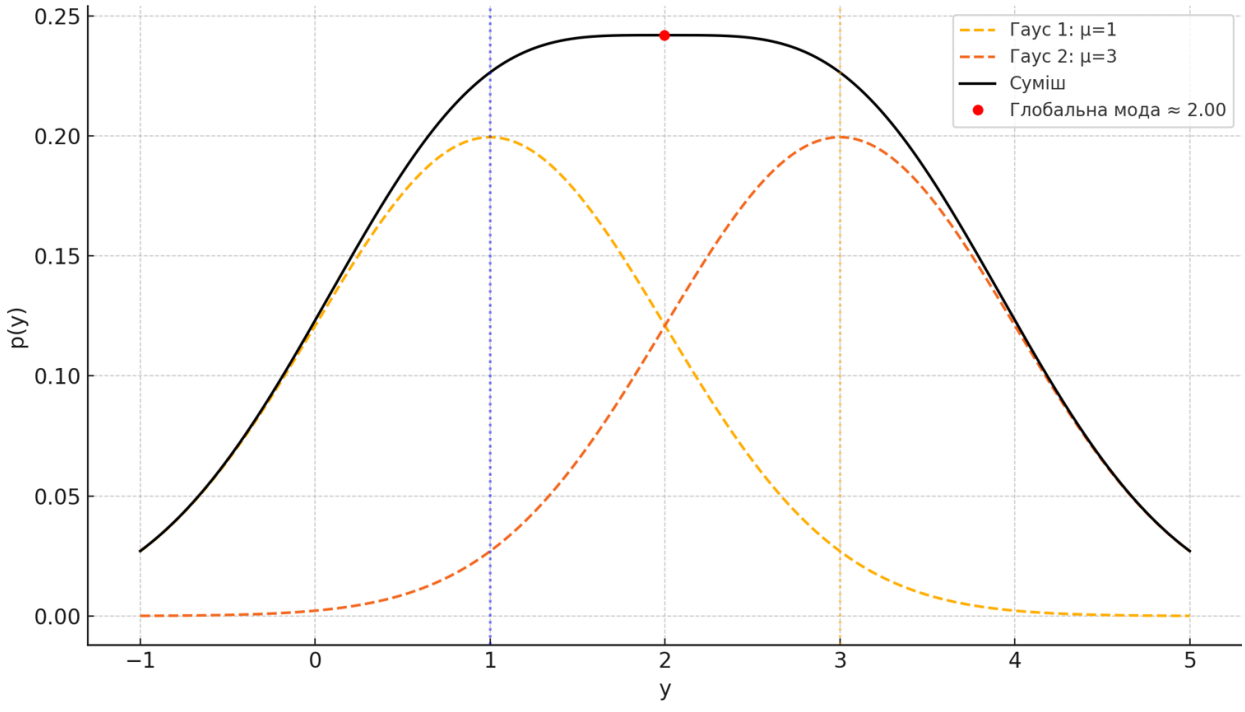


Рис. 17. Приклад, коли значення μ жодної з гаусової компоненти не збігається з глобальним максимумом

На графіку (рис. 17) видно, що у змодельованій ситуації глобальний максимум суміші двох гаусових розподілів (червона точка) не збігається з центром жодної з компонент (сині та помаранчеві вертикальні лінії). Це типовий випадок, коли просте використання μ_k для пошуку мод є хибним – справжній пік може лежати посередині, особливо при симетричних компонентах, або коли середні значення компонент знаходяться на невеликій відстані один від одного.

У модифікованій версії MDN, де втрата обчислюється як логарифм ймовірності потрапляння у вузький інтервал, модель може краще відображати локальні концентрації ймовірності.

У такому випадку:

- Моди стають ще важливішими – вони найкраще представляють основні ймовірні сценарії.
- Інтервальний підхід може підсилювати мультимодальність (наприклад, дві близькі гуасіани можуть породити окремі моди в суміші).

Отже, ефективний пошук мод змішаного гаусового розподілу, який породжує MDN, критично важливий для правильної інтерпретації результатів, особливо коли модель відображає невизначеність, альтернативність або складну структуру даних. Без цього інтерпретація вихідного розподілу буде поверхневою або навіть хибною.

2.4.1. *Метод густого семплювання з кластеризацією*

Метод густого семплювання з кластеризацією – це практичний і гнучкий підхід до виявлення мод (локальних максимумів) в ймовірнісному розподілі, породженому MDN [20].

Для аналізу мод в такому ймовірнісному розподілі ми повинні використовувати наступні його числові характеристики:

- ваги компонент (π_k);
- середні значення (μ_k);
- дисперсії (σ_k).

Для цього генерується велика кількість зразків (семплів) з цієї суміші, користуючись наступними правилами:

- Обираємо компоненту k з ймовірністю π_k.
- Семплюємо $y \sim \mathcal{N}(\mu_k, \sigma_k^2)$.

Виходить вибірка з розподілу $p(y)$. Після отримання семплів застосовуємо кластеризаційний алгоритм, який виявляє скупчення в даних – ці скупчення відповідають модам розподілу.

Найпоширеніші алгоритми для проведення етапу кластеризації [21-23]:

- *Mean-Shift*: автоматично знаходить моди, не потребує вказувати кількість кластерів.
- *DBSCAN*: знаходить скупчення на основі густини.
- *Gaussian Mixture Model + BIC*: повторна апроксимація даних іншою GMM для уточнення.

В результаті кластеризації ми отримаємо значення центрів кластеризації, які будуть рівними значенням мод.

Оцінка методу за ключовими характеристиками наведена у табл. 1.

Таблиця 1

Оцінка ключових характеристик методу семплювання з кластеризацією

Характеристика	Оцінка	Коментар
Швидкодія	Низька	Семплювання та кластеризація потребують значної кількості часу.
Потреба в обчислювальних ресурсах	Висока	Потребує багато пам’яті та обчислень при великій кількості семплів.
Точність	Висока	Дає точні оцінки мод за достатнього покриття простору.
Здатність знаходити глобальні та локальні моди	Висока	Здатен знайти як глобальні, так і локальні моди.
Ефективність при роботі з високовимірними просторами	Низька	У високих розмірностях метод деградує через прокляття розмірності.
Ефективність при наявності великої кількості компонент	Залежить від реалізації	При великій кількості компонент потрібна обережна кластеризація.
Стабільність результату	Висока за достатньої кількості семплів	Результат стійкий за умови якісного вибору параметрів кластеризації.

Серед основних переваг методу можемо зазначити наступні:

- Гнучкий: не залежить від аналітичної форми $p(y)$.
- Добре працює при складних формах суміші, перекриттях, асиметріях.
- Легко масштабувати при достатній обчислювальній потужності.

Проте, також варто зазначити і основні недоліки даного методу, зокрема:

- Чутливість до параметрів кластеризації (наприклад, bandwidth у Mean-Shift або eps/min_samples у DBSCAN).
- Висока обчислювальна вартість при великій кількості семплів, особливо в багатовимірному просторі.
- Існує ймовірність пропуску мод, якщо скупчення недостатньо чітко виражені.

2.4.2. *Метод градієнтного підйому*

Градієнтний підйом – це чисельний метод пошуку максимуму функції [24]. У випадку MDN ми хочемо знайти максимуми функції щільності ймовірності $p(y)$, яку задає суміш Гаусів. Оскільки аналітичний розв'язок рівняння $p'(y) = 0$ зазвичай неможливий, ми застосовуємо чисельний градієнтний підйом, щоб ітеративно підніматися по схилу функції щільності до локального максимуму. Розглянемо алгоритмічну суть методу:

1. Обчислюється значення функції $p(y)$, заданої як суміш, яку можемо охарактеризувати за допомогою формули (3).
2. Ініціалізуються стартові точки (одна або кілька). Найчастіше їх беруть із:
   - центрів компонент μ_k;
   - випадкових значень у діапазоні значущої густини;
   - сітки значень y.
3. Для кожної стартової точки обчислюється похідна $\nabla p(y)$, і виконується крок у напрямку збільшення функції.

4. Процес повторюється до досягнення максимуму (локального), або поки зміни не стануть незначними.

Оцінюючи метод за ключовими характеристиками було складено табл. 2.

Таблиця 2

Оцінка ключових характеристик градієнтного підйому

Характеристика	Оцінка	Коментар
Швидкодія	Висока	Працює швидко при обмеженій кількості стартових точок.
Потреба в обчислювальних ресурсах	Низька	Має низькі вимоги до ресурсів у простому варіанті.
Точність	Обмежена	Знаходить лише локальні моди й може пропустити глобальні.
Здатність знаходити глобальні та локальні моди	Обмежена	Покриває лише ті моди, до яких дійшли стартові точки.
Ефективність при роботі з високовимірними просторами	Низька	Градієнт часто нестабільний у багатовимірному просторі.
Ефективність при наявності великої кількості компонент	Низька	Складний ландшафт функції ускладнює пошук при великій кількості компонент.
Стабільність результату	Сильно залежить від стартових точок	Результат змінюється залежно від ініціалізації та кроку градієнта.

Можемо вказати основні переваги у використанні даного методу:

- Дає точне локальне розташування моди при хорошій ініціалізації.
- Простий у реалізації для одномірного і низьковимірного випадку.

- Не вимагає великих обчислювальних витрат у порівнянні з семплюванням.

Проте використання даного методу може мати певні негативні наслідки, причинами яких є те, що він:

- Залежить від стартових точок – легко пропустити моди, якщо почати не в тій області.
- Може зійтись до неінформативного максимуму, якщо функція має багато плато.
- Потребує чисельного обчислення похідної функції щільності, що може бути нестабільним.

2.4.3. *Метод Mean-Shift*

Mean-Shift – це безпараметричний метод для виявлення локальних максимумів у функції густини [21]. Він працює за принципом послідовного зсуву кожної точки до центру ваги її локального оточення, обчисленого за допомогою ядерної оцінки. Роботу даного методу можемо охарактеризувати наступним чином:

1. Вибирається ядерна функція (зазвичай гаусова) і параметр ширини вікна (bandwidth).
2. Для кожної стартової точки обчислюється зважене середнє її сусідів, значення кого стає новою позицією точки.
3. Процес повторюється, поки зсув не стане меншим за заданий поріг.
4. Точки, що зійшлися до одного положення, утворюють кластер, який відображатиме локальну моду.

Оскільки MDN задає розподіл у вигляді суміші Гаусів, Mean-Shift застосовують до великої вибірки, згенерованої з цієї суміші. Метод дозволяє виявити моди розподілу без потреби у знанні кількості компонент або мод наперед.

Оцінюючи метод за ключовими характеристиками було складено табл. 3.

Таблиця 3

Оцінка ключових характеристик методу Mean-Shift

Характеристика	Оцінка	Коментар
Швидкодія	Середня	Може бути повільним при великій кількості стартових точок.
Потреба в обчислювальних ресурсах	Середня	Потребує помірних ресурсів для зберігання сусідніх точок і повторних обчислень.
Точність	Висока	За правильно обраного bandwidth метод точно локалізує моди.
Здатність знаходити глобальні та локальні моди	Висока	Метод дозволяє виявляти кілька локальних максимумів без потреби у ініціалізації.
Ефективність при роботі з високовимірними просторами	Низька	У високій розмірності пошук сусідів і стабільність знижуються.
Ефективність при наявності великої кількості компонент	Середня	Працює, але потребує ретельного підбору bandwidth, інакше може зливати або ділити моди.
Стабільність результату	Залежить від bandwidth	Нестійкий до вибору параметра ширини ядра, що впливає на кількість і положення мод.

До основних переваг даного методу можемо зарахувати наступні:

- Не потребує задання кількості мод.
- Може виявляти моди складної форми.
- Природно об'єднує точки в області з високою щільністю.

Розглядаючи ж основні недоліки даного методу варто відзначити, що він:

- Чутливий до вибору параметра bandwidth.
- Повільний при великій кількості стартових точок.

- Не масштабується добре у високих розмірностях.

2.4.4. *EM-подібний локальний пошук*

Цей підхід базується на використанні ідей алгоритму очікування–максимізації (Expectation-Maximization), який класично застосовується для навчання параметрів сумішей гаусів. У контексті інтерпретації MDN, EM використовується не для навчання, а для локалізації мод густини [25].

Даний метод працює наступним чином:

1. Ініціалізуються стартові точки – зазвичай центрами компонент (μ_k) або точками на сітці.
2. Для кожної стартової точки виконується ітеративна процедура, аналогічна етапу максимізації у EM:
   - обчислюється очікуване положення максимуму (наприклад, через зважене оновлення положення);
   - повторюється до збіжності.
3. Отримані точки групуються, щоб виділити окремі моди.

На відміну від повного EM, цей метод не оновлює параметри всієї моделі, а виконує локальний уточнювальний пошук, використовуючи структуру вже відомої суміші. Це дозволяє виявити моди розподілу без семплювання. Оцінюючи метод за ключовими характеристиками було складено табл. 4.

Серед основних переваг даного методу можемо виділити наступні:

- Використовує вже доступну інформацію з MDN.
- Ефективний при невеликій кількості компонент.
- Не потребує додаткового навчання або семплювання.

Тоді як перелік основних недоліків даного методу має наступний вигляд:

- Знаходить лише локальні моди поблизу стартових точок.
- Не гарантує знаходження глобального максимуму.
- Нестійкий до поганого вибору початкових точок.

Таблиця 4

Оцінка ключових характеристик ЕМ-подібного локального пошуку

Характеристика	Оцінка	Коментар
Швидкодія	Висока	Локальні обчислення швидкі при малій кількості стартових точок.
Потреба в обчислювальних ресурсах	Низька	Метод працює без семплювання або кластеризації.
Точність	Середня	Дає точні результати в околі стартових точок, але може пропустити моди в інших зонах.
Здатність знаходити глобальні та локальні моди	Обмежена	Лише локальні моди, глобальна мода знаходиться лише при вдалій ініціалізації.
Ефективність при роботі з високовимірними просторами	Середня	Працює краще, ніж семплювання, але залежить від поведінки функції густини.
Ефективність при наявності великої кількості компонент	Низька	Багато компонент ускладнюють визначення, з яких точок починати локальний пошук.
Стабільність результату	Залежить від ініціалізації	Результати змінюються залежно від вибору стартових точок і їхньої близькості до мод.

2.4.5. *Адаптивне дискретизаційне оцінювання*

Цей підхід базується на прямому чисельному оцінюванні функції густини $p(y)$ на сітці значень у просторі вихідних змінних. Після побудови дискретного профілю густини, проводиться локальний аналіз максимумів – наприклад, шляхом пошуку сусідніх точок з меншими значеннями [26].

Розглянемо алгоритмічну суть даного методу:

1. Вибирається початкова сітка точок у в області, де ймовірність може бути значущою.
2. Для кожної точки обчислюється значення p(y), визначене сумішшю Гаусів з MDN.
3. За результатами можна:
   - знаходити глобальні та локальні піки шляхом аналізу сусідів;
   - уточнювати сітку (згущення у зонах високої щільності) – адаптивне уточнення.
4. У багатовимірному випадку застосовують розділення області на підобласті з подальшою деталізацією.

Метод дозволяє уникати стохастичних елементів (як у семплюванні), забезпечує повний контроль над точністю і забезпечує визначеність при виявленні мод. Оцінюючи метод за ключовими характеристиками було складено табл. 5.

Таблиця 5

Оцінка ключових характеристик адаптивного дискретизаційного оцінювання

Характеристика	Оцінка	Коментар
Швидкодія	Середня	Швидкість залежить від щільності сітки та розмірності простору.
Потреба в обчислювальних ресурсах	Висока	При високій точності або в багатовимірному випадку ресурсні витрати зростають швидко.
Точність	Висока	Метод дозволяє точно знаходити моди за рахунок контролю над роздільністю.
Здатність знаходити глобальні та локальні моди	Висока	При належній дискретизації видно всі значущі максимуми.

Продовження табл. 5

Ефективність при роботі з високовимірними просторами	Низька	Кількість точок зростає експоненційно з розмірністю.
Ефективність при наявності великої кількості компонент	Висока	Підходить для складних сумішей, оскільки не залежить від параметрів компонент.
Стабільність результату	Висока	Повністю детермінований метод, стабільність визначається сіткою та порогами.

До ключових переваг даного методу можемо віднести наступні:

- Прямий контроль над щільністю сітки і точністю.
- Дає як глобальні, так і локальні моди.
- Надійний у задачах низької розмірності.

Щодо переліку основних недоліків даного методу можемо віднести наступні:

- Експоненційне зростання кількості точок у високих розмірностях.
- Може стати повільним при використанні набору параметрів, націленому на досягнення високої точності.
- Потребує заздалегідь заданих меж і масштабу простору.

2.4.6. *Інтервальний аналіз*

Інтервальний аналіз ґрунтується на оцінюванні ймовірності потрапляння змінної y у задані вузькі інтервали. Це дозволяє моделі краще відображати локальні скупчення ймовірності. Розглянемо принципи роботи методу:

1. Простір у розбивається на вузькі інтервали (або гіперпрямокутники у вищих розмірностях).
2. Для кожного інтервалу обчислюється ймовірність (20), що y потрапить у нього:

$$P(y \in [a, b]) = \int_a^b p(y)dy. \quad (20)$$

3. Інтервали з найвищою інтегральною ймовірністю вважаються модовими зонами.
4. За потреби інтервали уточнюються, розділяються або зсуваються – утворюючи адаптивний пошук.

Особливо ефективний метод у випадках, коли MDN навчається з інтервальною втратою – тобто логарифмом ймовірності потрапляння у вузький інтервал. Така постановка автоматично стимулює модель створювати локальні підвищення ймовірності, які можна інтерпретувати як моди. Оцінюючи метод за ключовими характеристиками було складено табл. 6.

Таблиця 6

Оцінка ключових характеристик інтервального аналізу

Характеристика	Оцінка	Коментар
Швидкодія	Середня	Залежить від кількості інтервалів і швидкості обчислення інтегралів.
Потреба в обчислювальних ресурсах	Середня	Обчислення інтегралів по кожному інтервалу потребує значних ресурсів.
Точність	Висока	Дає точні оцінки локальної ймовірності за умови правильного вибору інтервалу.
Здатність знаходити глобальні та локальні моди	Висока	Інтервали з високою інтегральною ймовірністю відповідають модам.
Ефективність при роботі з високовимірними просторами	Низька	Інтервали ростуть експоненційно з розмірністю, що робить метод непридатним.
Ефективність при наявності великої кількості компонент	Висока	Метод працює незалежно від числа компонент, оцінюючи лише підсумковий розподіл.
Стабільність результату	Висока	Результати стабільні за фіксованого інтервалу та розбиття простору.

Переваги даного методу:

- Природно узгоджується з інтервальною функцією втрат.
- Може виявляти моди, що не збігаються з центрами компонент.
- Дозволяє побудувати карту з ймовірностями по всьому простору.

Недоліки даного методу:

- Не масштабується на високі розмірності.
- Потребує обчислення багатьох інтегралів (можна спростити аналітичними формулами для гаусових).
- Вимагає точного вибору ширини інтервалу.

2.4.7. *Висновки щодо методів інтерпретації результуючого ймовірнісного розподілу*

Аналіз різних методів інтерпретації результатів моделі MDN показує, що вибір оптимального підходу значною мірою залежить від особливостей задачі, доступних ресурсів і вимог до точності та масштабованості. Жоден із розглянутих методів не є універсальним, кожен має свої переваги й обмеження.

Методи, що базуються на семплюванні, кластеризації або дискретному аналізі, дозволяють ефективно знаходити як глобальні, так і локальні моди, але погано масштабуються на простори високої розмірності та потребують значних обчислювальних ресурсів. Натомість локальні методи, зокрема градієнтний підйом і EM-подібний пошук, забезпечують високу швидкодію та низьке споживання ресурсів, але мають обмежену здатність до глобального охоплення простору та сильно залежать від стартової ініціалізації. Інтервальний аналіз найкраще узгоджується з модифікованими версіями MDN, які оптимізують інтегральну ймовірність у малих інтервалах, і дозволяє точно і стабільно визначати локальні скупчення ймовірності. Адаптивне дискретизаційне оцінювання показує високу точність у просторах малої розмірності, але швидко втрачає ефективність при зростанні кількості вимірів.

Вибір конкретного методу повинен враховувати розмірність простору, кількість гаусових компонент, вимоги до точності та швидкодії, а також те, як саме було навчено MDN-модель.

2.5. Висновки до другого розділу

У другому розділі дисертаційної роботи було детально проаналізовано метод прогнозування мультимодальних даних на основі моделі MDN та запропоновано її модифікацію, спрямовану на покращення точності, інтерпретованості та стабільності навчання. Розглянуто основні принципи побудови MDN, яка дозволяє моделювати умовний розподіл вихідних даних у вигляді суміші гаусових компонент і є особливо корисною у задачах з невизначеністю та багатозначністю.

Було обґрунтовано потребу у зміні класичної функції втрат, яка базується на щільності ймовірності в одній точці, на функцію, що оцінює ймовірність потрапляння у вузький інтервал навколо спостереження. Такий підхід дозволяє уникнути негативних значень втрат, покращити інтерпретацію результатів навчання та надати моделі більшу відповідність ймовірнісній природі задачі.

Було показано, що для реалізації модифікованої функції втрат найдоцільніше використовувати аналітичне обчислення визначеного інтегралу через кумулятивну функцію нормального розподілу, що забезпечує високу точність, швидкодію та обчислювальну стабільність у порівнянні з чисельними та стохастичними методами.

Окрему увагу було приділено аналізу методів інтерпретації результуючого ймовірнісного розподілу, зокрема пошуку локальних максимумів (мод), які представляють найбільш ймовірні сценарії. У цьому контексті розглянуто як чисельні (градієнтний підйом, EM-подібний пошук, адаптивне дискретизаційне оцінювання), так і стохастичні (семплювання з кластеризацією, Mean-Shift) підходи.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ПРОГНОЗУВАННЯ ДАНИХ З МУЛЬТИМОДАЛЬНИМ РОЗПОДІЛОМ

У цьому розділі підсумовуються основні результати виконаної роботи, що охоплює як аналіз технологічних можливостей для реалізації моделі MDN, так і безпосереднє програмне впровадження запропонованого методу. Узагальнення охоплює вибір середовища розробки, обґрунтування змін у підході до формування функції втрат, структуру побудованої моделі та організацію процесу її навчання. Такий комплексний підхід дозволяє оцінити ефективність запропонованого рішення та сформувати основу для подальших експериментальних досліджень.

3.1. Аналіз технологій для програмної реалізації методу

При реалізації та тестуванні нових підходів до побудови моделей нейронних мереж важливу роль відіграє вибір відповідної технологічної основи. Йдеться насамперед про інструменти, які забезпечують можливості для точного математичного моделювання, ефективного навчання, зручного налаштування компонентів моделі, а також подальшого аналізу її роботи. Такий вибір охоплює низку взаємопов’язаних рівнів – від мови програмування до спеціалізованих бібліотек і фреймворків для побудови та навчання моделей. Кожен із цих рівнів має свої особливості, які можуть мати суттєвий вплив на якість, зручність та надійність реалізації запропонованого підходу. У подальшому ми поступово розглянемо ці аспекти, починаючи з аналізу мов програмування, що найчастіше використовуються для розробки моделей машинного навчання.

3.1.1. *Набір технологій на основі мови програмування Python*

Технологічний стек, пов’язаний із Python, є найпоширенішим та найповніше підтримуваним у сфері машинного навчання, що робить його особливо придатним для реалізації нових методів, зокрема модифікованого

підходу до побудови MDN. Стек включає широкий набір інструментів для розробки, експериментів, аналізу та подальшого впровадження моделей.

Основні компоненти цього стеку:

- Мова програмування та середовище виконання: Python 3.x – основна мова розробки; Kaggle Notebooks – хмарне середовище з попередньо налаштованими бібліотеками, GPU/TPU підтримкою, простим спільним доступом і можливістю швидкого тестування ідей без додаткового налаштування локального середовища.
- Науково-обчислювальні бібліотеки: NumPy – для ефективної роботи з масивами та лінійною алгеброю; SciPy – для чисельного інтегрування, оптимізації та статистичних обчислень; Pandas – для структурованої роботи з табличними даними; Matplotlib / Seaborn – для побудови графіків, зокрема щільностей, втрат та результатів навчання.
- Бібліотеки глибокого навчання: TensorFlow + Keras – для декларативного опису архітектур і кастомізації функцій втрат; PyTorch – для гнучкої імперативної побудови моделей, зручною для реалізації нових підходів до обробки ймовірнісних розподілів; TensorFlow Probability / Pyro – для роботи з параметричними розподілами, що необхідно при інтеграції функції втрат на основі інтервальної ймовірності.
- Інструменти навчання, валідації та моніторингу: Optuna / Ray Tune – для гіперпараметричної оптимізації; TensorBoard / Weights & Biases – для візуалізації метрик і збереження перебігу експериментів (доступні також у середовищі Kaggle із мінімальними налаштуваннями).
- Інструменти розгортання моделей (опційно): ONNX – для експорту моделей між фреймворками; Flask / FastAPI – для створення API сервісів, які взаємодіють із моделями; Docker – для контейнеризації середовища виконання при перенесенні на сервери або в хмару.

Таким чином, Python-стек із використанням середовища Kaggle забезпечує не лише простоту реалізації та повторюваність експериментів, а й доступ до сучасних обчислювальних ресурсів, необхідних для тестування нових методів моделювання.

3.1.2. *Набір технологій для мов програмування C/C++*

Технологічний стек, пов'язаний із C/C++, відрізняється від Python-орієнтованих рішень своїм фокусом на максимальну продуктивність, низькорівневий контроль і ефективність використання ресурсів. Цей стек традиційно застосовується там, де критичними є швидкість обчислень, стабільність та оптимізація під конкретне апаратне забезпечення, зокрема у вбудованих системах або при розгортанні моделей у виробничих середовищах із жорсткими обмеженнями.

Основні компоненти цього стеку:

- Мови та середовище виконання: C/C++ (C++17 або новіші версії) – основні мови, які забезпечують контроль над пам'яттю, низькорівневу оптимізацію і компіляцію під конкретну архітектуру; CMake / Make – системи збірки, які дозволяють керувати залежностями та компіляцією великих проєктів; IDE: Visual Studio, CLion, Qt Creator – для зручної роботи з кодом, дебагом та компіляцією.
- Бібліотеки для наукових і чисельних обчислень: Eigen – потужна бібліотека для роботи з лінійною алгеброю, яка активно використовується у багатьох ML-фреймворках; Armadillo / Blaze – альтернативи Eigen з хорошою продуктивністю; Boost – набір шаблонів і математичних утиліт, що полегшують реалізацію складних алгоритмів.
- Фреймворки для машинного навчання / нейронних мереж: dlib – бібліотека C++ для реалізації базових ML-алгоритмів і нейронних мереж; tiny-dnn – легковаговий фреймворк для глибокого навчання,

призначений для простих моделей у вбудованих системах; CNTK (Microsoft Cognitive Toolkit) – хоч і частково застарілий, він використовує C++-ядро для високоефективного навчання; TensorRT (від NVIDIA) – оптимізатор і рантайм для розгортання моделей нейронних мереж з високою швидкістю виконання на GPU.

- Бібліотеки для роботи з розподілами та статистикою: C++ не має таких потужних вбудованих засобів, як Python, однак можна використовувати: Boost.Math, що містить реалізації основних ймовірнісних функцій (PDF, CDF, тощо); Cephes / GSL (GNU Scientific Library) – для реалізації спеціальних математичних функцій та чисельного інтегрування.
- Паралелізація та оптимізація: OpenMP / Intel TBB – для багатопотокових обчислень; CUDA / cuDNN – для реалізації обчислень на GPU з мінімальною затримкою, що важливо при роботі з великими обсягами даних або складними розподілами.

Хоч цей стек не настільки зручний для досліджень і прототипування, як Python, він надає критичні можливості у випадках, коли необхідна максимальна ефективність або модель має бути інтегрована у складне виробниче середовище (наприклад, автономна система, промислове ПЗ, пристрій реального часу).

3.1.3. *Набір технологій для мови програмування JavaScript*

Технологічний стек, пов’язаний із JavaScript, має специфічну нішу у сфері машинного навчання, орієнтовану переважно на використання моделей у вебсередовищі, інтерактивну візуалізацію результатів та легке розгортання клієнтських рішень без потреби в серверній інфраструктурі. Цей стек не є оптимальним для первинної розробки чи навчання складних моделей, однак він може бути корисним для демонстрації, інтеграції з вебінтерфейсами або обробки даних у браузері.

Основні компоненти цього стеку:

- Мова та середовище виконання: JavaScript (ES6+) – основна мова розробки; Node.js – середовище виконання на стороні сервера для обробки ML-моделей або попередньої обробки даних; Web browsers (Chrome, Firefox, etc.) – середовище виконання на клієнтській стороні, де може працювати модель прямо у браузері без додаткових установок.
- Бібліотеки для машинного навчання: TensorFlow.js – основний фреймворк для розробки, тренування та використання моделей нейронних мереж у JavaScript. Підтримує імпорт моделей, створених у Python (через формат SavedModel або tfjs-converter), а також побудову моделей безпосередньо в браузері; Brain.js – простіший фреймворк для базових задач машинного навчання (рекомендації, класифікація), придатний для швидкої інтеграції; ONNX.js – рантайм для запуску моделей у форматі ONNX у браузері; Synaptic / ML5.js – бібліотеки з простим API, орієнтовані на освітні та демонстраційні цілі.
- Бібліотеки для візуалізації та інтерфейсної інтеграції: D3.js / Plotly.js – для побудови складних графіків і візуалізацій результатів роботи моделей; Three.js – для 3D-візуалізацій; React / Vue / Angular – сучасні фреймворки для побудови інтерактивних інтерфейсів користувача, з можливістю прямої інтеграції з моделями, які працюють на клієнтському боці.
- Інструменти для розгортання та обслуговування: Vite / Webpack / Parcel – для збирання клієнтських застосунків; Netlify / Vercel / GitHub Pages – для швидкого хостингу вебдодатків із вбудованими ML-моделями; tfjs-vis – модуль TensorFlow.js для візуалізації процесу навчання та структури моделі.

JavaScript-стек не забезпечує повноцінного наукового або дослідницького середовища, однак він дозволяє швидко реалізувати

прототип користувацького інтерфейсу, інтерактивно візуалізувати результати моделі або розгорнути модель безпосередньо в браузері без серверного коду.

3.1.4. *Висновки до аналізу технологій для програмної реалізації методу*

На основі проведеного аналізу трьох технологічних стеків – Python, C/C++ та JavaScript – можемо сформулювати узагальнені висновки щодо їхньої придатності до реалізації модифікованого підходу до побудови моделі Mixture Density Network із функцією втрат, яка ґрунтується на інтегралі ймовірності в заданому інтервалі. У процесі оцінювання було враховано такі аспекти, як гнучкість, продуктивність, наявність необхідних бібліотек, зручність реалізації, доступ до обчислювальних ресурсів, а також рівень підтримки спільноти.

C/C++ демонструють високу швидкодію, низькорівневий контроль над ресурсами та потенціал для розгортання у продуктивних або вбудованих середовищах. Проте складність реалізації, обмеженість готових бібліотек для роботи з нейронними мережами та чисельними методами, а також відсутність зручного інструментарію для швидкого прототипування ускладнюють їх використання на етапі дослідження та експериментів.

JavaScript-стек виявляється корисним переважно у сфері візуалізації, побудови інтерфейсів користувача та розгортання моделей на клієнтській стороні, зокрема у браузері. Однак він не забезпечує необхідної глибини математичного та алгоритмічного опрацювання для повноцінного навчання складних моделей або реалізації нестандартних функцій втрат. Його застосування є доцільним лише як допоміжний інструмент для демонстрації або взаємодії з уже готовими моделями.

Натомість технологічний стек Python забезпечує найбільш повне та збалансоване середовище для реалізації описаного підходу. Він поєднує у собі наявність зрілих фреймворків для глибокого навчання, таких як TensorFlow та PyTorch, з можливістю гнучкого моделювання та створення

користувацьких функцій втрат. Крім того, наявні спеціалізовані бібліотеки для статистики, чисельного інтегрування та обробки даних, зокрема NumPy, SciPy та TensorFlow Probability, які є ключовими для реалізації інтервальної функції втрат. Середовище виконання, таке як Kaggle, дозволяє використовувати обчислювальні ресурси без додаткових налаштувань, що значно прискорює проведення експериментів. Усі ці чинники, разом із широкою підтримкою спільноти та якісною документацією, створюють оптимальні умови для розробки, тестування й удосконалення моделі.

Отже, вибір зупиняється на технологічному стеку Python як на найбільш доцільному та практичному рішенні для реалізації запропонованого методу.

3.2. Програмна реалізація методу мовою Python

У цьому підрозділі подано детальний опис програмної реалізації моделі Mixture Density Network як у класичному, так і в модифікованому варіанті. Реалізація охоплює побудову архітектури нейронної мережі, визначення функцій втрат відповідно до обраного підходу, а також організацію процесу навчання моделей.

Лістинг 1. Підключення необхідних бібліотек

```
import numpy as np
import torch
import torch.nn as nn
from torch.distributions import Normal
```

Цей блок коду забезпечує підключення основних бібліотек, необхідних для побудови та навчання моделі MDN. Використовується NumPy для роботи з масивами і числовими даними, а також PyTorch як основний фреймворк для побудови нейронної мережі, її навчання й обчислення градієнтів. Модуль torch.nn надає інструменти для опису архітектури мережі, а torch.distributions.Normal використовується для роботи з нормальними розподілами, що є ключовим елементом у формуванні ймовірнісного виходу MDN.

Лістинг 2. Приклад реалізації класу MDN

```
class MDN(nn.Module):
    def __init__(self, n_hidden, n_gaussians):
        super(MDN, self).__init__()
        self.z_h = nn.Sequential(
            nn.Linear(1, n_hidden),
            nn.Tanh()
        )
        self.z_lambda = nn.Linear(n_hidden, n_gaussians)
        self.z_sigma = nn.Linear(n_hidden, n_gaussians)
        self.z_mu = nn.Linear(n_hidden, n_gaussians)

    def forward(self, x):
        z_h = self.z_h(x)
        pi = nn.functional.softmax(self.z_lambda(z_h), -1)
        sigma = torch.exp(self.z_sigma(z_h))
        mu = self.z_mu(z_h)
        return pi, sigma, mu
```

Цей код описує архітектуру класичної моделі MDN у вигляді класу на базі PyTorch. Модель поєднує властивості нейронної мережі та ймовірнісної моделі, формуючи на виході параметри суміші гаусових розподілів: ваги компонент, середні значення та стандартні відхилення.

На вхід модель приймає значення (в даному випадку одномірне), яке проходить через прихований шар з n_hidden нейронами та нелінійною активацією (гіперболічний тангенс). Далі побудовано три окремі лінійні шари, які формують вихідні параметри суміші: z_lambda генерує значення, з яких за допомогою softmax отримуються ваги компонент (pi), z_mu – середні значення (mu), а z_sigma – значення стандартних відхилень, які експонуються для забезпечення додатності (sigma). Повертаються тензори, які містять відповідні параметри для кожної гаусової компоненти у суміші. Така модель дозволяє на кожному кроці формувати не одне передбачене значення, а ймовірнісний розподіл, що описує невизначеність виходу.

Лістинг 3. Реалізація функцій обчислення гаусових ймовірностей

```
oneDivSqrtTwoPI = 1.0 / np.sqrt(2.0*np.pi)
def gaussian_distribution(y, mu, sigma):
    result = (y.expand_as(mu) - mu) * torch.reciprocal(sigma)
    result = -0.5 * (result * result)
    return (torch.exp(result) * torch.reciprocal(sigma)) *
oneDivSqrtTwoPI
```

```
def gaussian_probability(y, mu, sigma, epsilon):
    normal_distribution = Normal(mu, sigma)
    upper = y + epsilon
    lower = y - epsilon
    upper = upper.expand_as(mu)
    lower = lower.expand_as(mu)
    result = normal_distribution.cdf(upper) -
normal_distribution.cdf(lower)
        return result
```

У цьому фрагменті реалізовано дві функції, що відповідають за обчислення ймовірностей у межах нормального розподілу, зокрема в контексті моделі MDN.

Перша функція gaussian_distribution реалізує класичну формулу щільності ймовірності для нормального розподілу. Вона використовується для точкової оцінки щільності в конкретному значенні y, що є основою у класичній реалізації функції втрат у MDN.

Друга функція gaussian_probability реалізує аналітичне обчислення ймовірності потрапляння у інтервал $[y - \varepsilon, y + \varepsilon]$ для кожної компоненти нормального розподілу. На відміну від чисельного інтегрування, тут використовується кумулятивна функція розподілу (CDF), яка доступна у бібліотеці torch.distributions.Normal. Під час обчислення вхідне значення y автоматично поширюється до розмірів тензора mu, що дозволяє ефективно працювати з батчами даних. Ця функція є ключовою частиною реалізації модифікованого підходу до функції втрат, в якому замість щільності в точці використовується ймовірність на інтервалі.

Лістинг 4. Реалізація функцій втрат

```
def modified_mdn_loss_fn(pi, sigma, mu, y, epsilon):
    result = gaussian_probability(y, mu, sigma, epsilon) * pi
    result = torch.sum(result, dim=1)
    result = -torch.log(result)
    return torch.mean(result)

def mdn_loss_fn(pi, sigma, mu, y):
    result = gaussian_distribution(y, mu, sigma) * pi
    result = torch.sum(result, dim=1)
    result = -torch.log(result)
    return torch.mean(result)
```

У цьому коді подано дві реалізації функції втрат для моделі MDN: класичну та модифіковану, які відрізняються підходом до оцінки ймовірності правильного передбачення.

Функція mdn_loss_fn реалізує класичну втрату для MDN. Вона базується на точковому значенні щільності ймовірності (PDF) у точці спостереження y. Для кожної гаусової компоненти обчислюється щільність, вона зважується відповідним коефіцієнтом з pi (ймовірність компоненти), після чого всі значення сумуються як ймовірність для суміші. Потім береться логарифм і мінус для отримання від'ємного логарифма правдоподібності. Результатом є середнє значення цієї величини по всьому батчу.

Натомість modified_mdn_loss_fn реалізує модифіковану функцію втрат, яка використовує не точкове значення щільності, а аналітичну ймовірність попадання в інтервал шириною 2ε навколо значення y. Це дозволяє уникнути проблем із нестабільністю, пов'язаних із тим, що PDF може приймати значення, більші за 1, і, відповідно, призводити до від'ємних значень втрат. Завдяки використанню кумулятивної функції розподілу (через gaussian_probability), тут обчислюється інтегральна ймовірність для кожної компоненти, яка потім так само зважується на pi, сумується, логарифмується зі знаком мінус, і усереднюється по батчу.

Таким чином, обидві функції мають однакову структуру, але використовують різну базову ймовірнісну оцінку. Модифікований варіант є більш узагальненим і потенційно більш стабільним у випадках шумних або неточно виміряних цільових значень.

Лістинг 5. Приклад реалізації функцій навчання моделей

```
modified_MDN_model = MDN(n_hidden=20, n_gaussians=2)
MDN_model = MDN(n_hidden=20, n_gaussians=2)

optimizer = torch.optim.Adam(modified_MDN_model.parameters())
mdn_optimizer = torch.optim.Adam(MDN_model.parameters())

def train():
    for epoch in range(epochs):
        pi_variable, sigma_variable, mu_variable =
modified_MDN_model(x_variable)
```

```
        modified_mdn_loss = modified_mdn_loss_fn(pi_variable,
sigma_variable, mu_variable, y_variable, integral_epsilon)
        optimizer.zero_grad()
        modified_mdn_loss.backward()
        optimizer.step()

def train_mdn():
    for epoch in range(epochs):
        pi_variable, sigma_variable, mu_variable =
MDN_model(x_variable)
        loss = mdn_loss_fn(pi_variable, sigma_variable, mu_variable,
y_variable)
        mdn_optimizer.zero_grad()
        loss.backward()
        mdn_optimizer.step()
```

У цьому фрагменті створюються два окремі екземпляри моделі MDN з однаковою структурою – прихованим шаром на 20 нейронів і двома компонентами у суміші. Для кожної з моделей також створено окремий оптимізатор типу Adam, який забезпечує оновлення вагів під час навчання.

Описано дві функції навчання: train відповідає за процес оптимізації для моделі з модифікованою функцією втрат, яка базується на аналітичному обчисленні ймовірності потрапляння у заданий інтервал навколо цільового значення. У ній на кожному етапі виконується пряме проходження моделі, обчислення втрати, обнулення градієнтів, обчислення градієнтів та оновлення параметрів.

Функція train_mdn виконує ті самі дії, але для класичної MDN-моделі, де функція втрат ґрунтується на точковій щільності ймовірності у спостереженні.

3.3. Висновки до третього розділу

У ході виконаної роботи було реалізовано послідовний підхід до впровадження і дослідження модифікованої моделі MDN, яка передбачає зміну у формулюванні функції втрат – замість оцінки щільності ймовірності в одній точці, пропонується використовувати інтегральну ймовірність на малому інтервалі навколо цільового значення. Такий підхід спрямований на підвищення чисельної стабільності та інтерпретованості моделі, особливо у

випадках, коли розподіл результатів не є точковим або наявні вимірювальні похибки.

Першим етапом стало обґрунтування вибору технологічного стеку для реалізації поставленого завдання. Було проаналізовано три альтернативи: Python, C/C++ та JavaScript. Стек на основі Python виявився найбільш придатним завдяки поєднанню гнучкості, потужних бібліотек для машинного навчання, аналітичної обробки, чисельних методів і зручного середовища виконання. C/C++ продемонстрували потенціал у контексті високопродуктивних або вбудованих рішень, однак виявилися малопридатними для дослідницької реалізації. JavaScript-стек розглядався як допоміжний, зосереджений на візуалізації та клієнтському розгортанні, але не придатний для основного етапу навчання.

На наступному етапі було детально описано програмну реалізацію моделі MDN у середовищі PyTorch, що включало визначення архітектури мережі, реалізацію класичної функції втрат на основі щільності, а також модифікованої функції втрат на основі інтегральної ймовірності, яка обчислюється аналітично через кумулятивну функцію нормального розподілу. Була подана і структура навчання для обох моделей – класичної та модифікованої – з чітким розділенням процесів оптимізації. Такий підхід дозволяє не лише реалізувати обидва варіанти, а й створює умови для їхнього експериментального порівняння в однакових умовах.

Таким чином, у роботі було обґрунтовано та реалізовано альтернативний підхід до побудови ймовірнісної моделі MDN, а також закладено основу для подальшого аналізу результатів, тестування поведінки функції втрат і вивчення впливу запропонованої зміни на якість прогнозування.

4. АНАЛІЗ ТА ОЦІНКА ЗАРОПОНОВАНОГО МЕТОДУ ПРОГНОЗУВАННЯ ДАНИХ З МУЛЬТИМОДАЛЬНИМ РОЗПОДІЛОМ

У цьому розділі представлено результати експериментальної оцінки запропонованого методу на низці задач із мультимодальною природою даних. Проведені дослідження дозволили кількісно підтвердити переваги модифікованої моделі у порівнянні з класичними підходами як на синтетичних, так і на прикладних прикладах. Крім аналізу отриманих результатів, у розділі також окреслено можливі напрямки подальших досліджень, спрямовані на вдосконалення моделі, розширення її застосування та оптимізацію обчислювальних процесів.

4.1. Перевірка точності запропонованого методу за різних умов

З метою перевірки ефективності запропонованого підходу було проведено низку експериментів, спрямованих на порівняння його результатів із класичними методами моделювання. Основну увагу зосереджено на задачах передбачення з мультимодальним розподілом вихідних даних, де традиційні підходи часто виявляються недостатньо точними або стабільними. Експерименти охоплюють як синтетичні, так і прикладні задачі, включаючи моделювання зворотної кінематики роботизованих систем. У кожному з випадків аналізується точність передбачення, динаміка навчання та загальна стійкість моделей. Такий підхід дозволяє всебічно оцінити переваги модифікованої структури нейронної мережі у різних умовах.

4.1.1. *Задача прогнозування синтетичних даних з мультимодальним розподілом*

Метою першого експерименту є оцінка ефективності розробленого методу, що базується на модифікованій версії MDN, у порівнянні з класичною реалізацією MDN. Для цього була сформульована синтетична задача передбачення, у якій для кожного значення вхідного параметра існує два коректних варіанти вихідного значення. Така задача моделює ситуацію з

мультимодальним розподілом вихідних даних, тобто випадок, коли одна вхідна величина може відповідати кільком різним, але однаково ймовірним, виходам.

Цей тип задач є особливо релевантним для перевірки роботи MDN, оскільки ця архітектура дозволяє явно моделювати мультимодальність розподілу, на відміну від класичних регресійних підходів.

Умови проведення експерименту:

- Тип задачі: регресія з мультимодальним розподілом цільової змінної.
- Кількість входів: 1 числове значення.
- Кількість виходів: 2 коректних значення для кожного входу.
- Кількість компонентів у Gaussian Mixture: 2.
- Архітектура нейронної мережі: один прихований шар з 20 нейронами, активаційна функція – tanh.
- Оптимізатор: Adam (повна пакетна обробка, тобто full-batch).
- Кількість епох навчання: 1500.
- Стратегія ініціалізації: початкові ваги в обох моделях однакові, щоб забезпечити рівні умови старту.
- Гіперпараметр ε модифікованої моделі, який відповідає за ширину інтервалу інтегрування встановлено рівним 0,15.

Датасет для навчання та оцінки моделей був згенерований наступним чином:

- Згенеровано 1000 чисел x_i, рівномірно розподілених на інтервалі $[-15, 15]$.
- Для кожного значення x обчислюються два відповідні значення y_1 і y_2 згідно формул (21,22):

$$y_1 = \sin(\phi_0 + \phi_1 \cdot x) \cdot e^{-\frac{(\phi_0+\phi_1 \cdot x)^2}{32}} + 0.01 \cdot \epsilon_1, \quad (21)$$

$$y_2 = \log(x + 15) + 0.01 \cdot \epsilon_2, \quad (22)$$

де параметри ϕ_0 та ϕ_1 були встановлені на 3 та 0,9 відповідно. ϵ_1 та ϵ_2 – шумові складові, вибрані з нормального розподілу (для введення стохастичності у дані). Дані були поділені на навчальні та тестові датасети з частками 80% та 20. Візуалізацію датасету можна спостерігати на рис. 18.

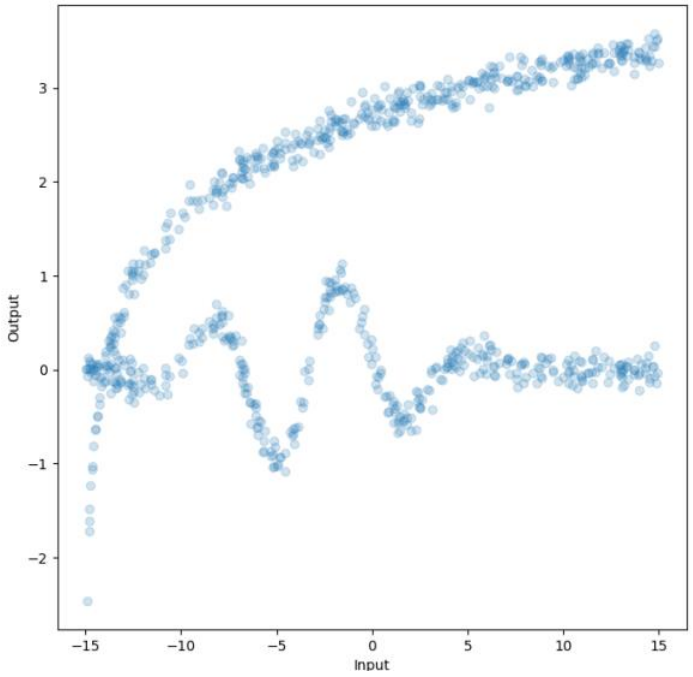


Рис. 18. Візуалізація згенерованих даних

Після навчання обох моделей – класичної MDN та запропонованої модифікованої MDN – протягом 1500 епох, було оцінено їхню ефективність за кількома метриками та сформовано відповідну таблицю (табл. 7).

Таблиця 7

Результати для задачі прогнозування синтетичних даних

Метрика	Класична MDN	Модифікована MDN	Покращення модифікованої моделі
Остаточна втрата (loss) на тренувальному наборі	1.208	1.152	+4.559%
Остаточна втрата на тестовому наборі	1.173	1.120	+4.561%

Продовження табл. 7

Кумулятивна втрата (епохи 500–1500) – тренувальний набір	1337.074	1304.965	+2.401%
Кумулятивна втрата – тестовий набір	1318.767	1291.521	+2.066%

Кумулятивна втрата – це сумарна помилка за всі епохи починаючи з 500-ї (для конкретно цього випадку), що дозволяє оцінити стабільність моделі на етапі конвергенції.

Порівняння динаміки навчання обох моделей можемо спостерігати на рис. 19.

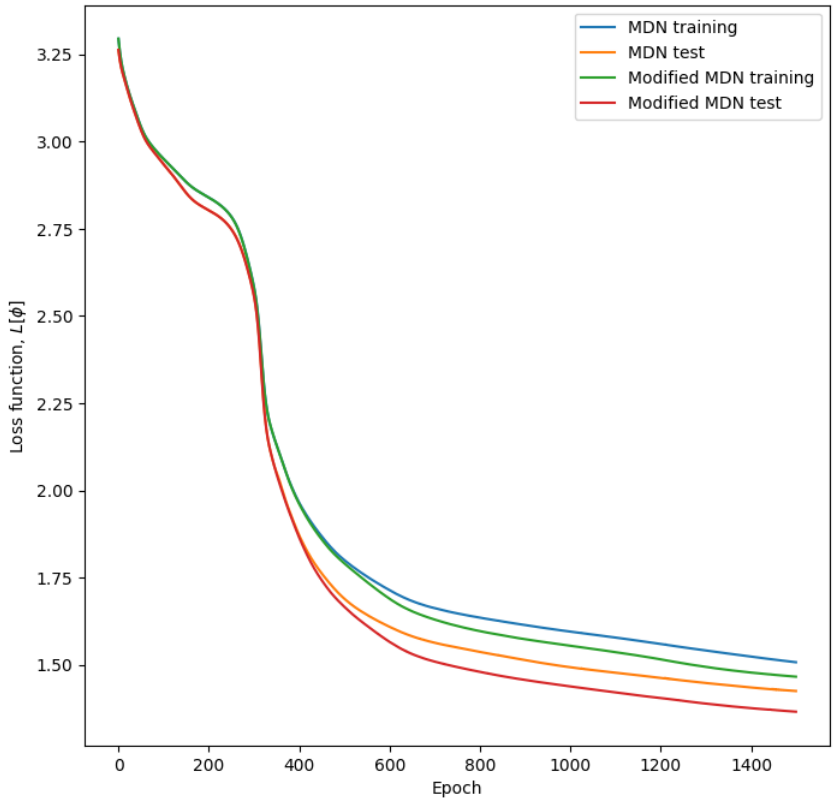


Рис. 19. Порівняння динаміки навчання для оригінальної та модифікованої моделей MDN з точки зору покращення значень втрат для навчальних та тестових даних

У результаті проведеного експерименту із синтетичними даними модифікована версія мережі з ймовірнісним виходом (MDN)

продемонструвала стабільне покращення показників у порівнянні з класичною реалізацією MDN. Модель, що включала запропоновані зміни, досягла нижчих значень функції втрат як на тренувальній, так і на тестовій вибірках. Зокрема, остаточні значення втрат зменшилися приблизно на 4.5%, що свідчить про вищу точність передбачення. Також спостерігається зменшення кумулятивної помилки за період стабільного навчання (починаючи з 500-ї епохи): на 2.4% для тренувальних даних і на 2.1% для тестових. Це вказує на те, що модифікована модель не лише досягає кращих результатів, але й робить це більш стабільно впродовж навчання.

Особливо важливо відзначити, що обидві моделі були ініціалізовані однаковими параметрами та навчалися на однаковому датасеті за ідентичних умов, що забезпечує об’єктивність порівняння. Візуальний аналіз динаміки навчання (рис. 19) також підтверджує перевагу модифікованої архітектури – вона демонструє швидшу конвергенцію та меншу флуктуацію значень функції втрат. Отже, результати експерименту підтверджують ефективність запропонованого методу для задач, де вихідні дані мають мультимодальний характер. Модифікована MDN-модель здатна краще адаптуватися до такої структури даних, забезпечуючи вищу якість моделювання та передбачення.

4.1.2. *Задача зворотної кінематики для двосегментного маніпулятора*

Далі буде проведено порівняння точності передбачення моделей у більш прикладних сценаріях. Як приклад застосування моделі прогнозування до даних з мультимодальним розподілом, розглядається задача, пов’язана з кінематикою роботизованої руки.

Опишемо цю задачу більш детально. Уявімо собі роботизовану руку, яка складається з двох сегментів, з’єднаних між собою та діючих в межах однієї площини. Один кінець першого сегмента закріплений у просторі, а інший з’єднаний з одним кінцем другого сегмента. Обидва сегменти можуть обертатися в межах заданих обмежень. Завдання полягає у тому, щоб

передбачити кути обертання для кожного з сегментів за заданими координатами цільової точки так, щоб вільний кінець руки максимально наблизився до цієї точки.

З математичної точки зору, координати x_1 та x_2 вільного кінця роботизованої руки з довжинами сегментів L_1 та L_2, і кутами обертання θ_1 та θ_2, задаються наступними формулами (23, 24):

$$x_1 = L_1 \cos(\theta_1) - L_2 \cos(\theta_1 + \theta_2), \quad (23)$$

$$x_2 = L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2). \quad (24)$$

Візуальне зображення цієї конфігурації наведено на рис. 20.

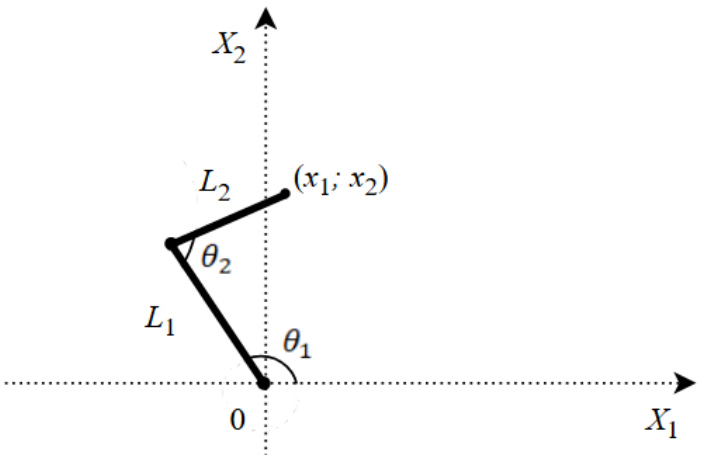


Рис. 20. Схематичне зображення двовимірної роботизованої руки, що складається з двох сегментів довжиною L_1 та L_2

Ця задача передбачає роботу з даними, які мають мультимодальний розподіл, оскільки для певних координат цільової точки можуть існувати дві різні комбінації кутів, що забезпечують досягнення одного й того самого положення кінця руки. У межах цього дослідження було проведено експеримент для порівняння точності передбачення цієї задачі з використанням класичної та модифікованої моделей MDN, а також нейронної мережі, навченої за методом найменших квадратів.

Зазначимо умови експерименту:

- Тип задачі: регресія з мультимодальністю виходу.
- Вхід: 2 значення – координати цільової точки (x_1, x_2).

- Вихід: 2 значення – кути обертання сегментів (θ_1, θ_2).
- Характер задачі: одні й ті ж координати можуть досягатися різними комбінаціями кутів, що зумовлює неоднозначність рішення.

Зазначимо обрані параметри моделей, роз починаючи з MDN-моделей:

- 2 гаусові компоненти.
- Вхідний шар: 2 нейрони.
- Прихований шар: 20 нейронів.
- Вихідний шар: 12 нейронів (для параметрів гаусових сумішей).

Least squares модель має ту саму архітектуру, але вихідний шар містить лише 2 нейрони (без моделювання розподілу). Для усіх моделей:

- Активаційна функція: tanh.
- Оптимізатор: Adam (повна пакетна обробка).
- Кількість епох: 1000.

Для симуляції даних було змодельовано рух двосегментного маніпулятора в межах допустимих значень кутів. Було задано фіксовані довжини обох сегментів та визначено допустимі діапазони обертання для кожного з кутів. Було згенеровано 1000 випадкових комбінацій кутів у межах заданих обмежень та для кожної пари кутів обчислено координати кінцевої точки маніпулятора за формулами (23, 24). В результаті отримали датасет у форматі: вхідні координати → вихідні кути.

Візуалізацію результуючого датасету можемо спостерігати на рис. 21.

Порівняння точності передбачення моделей було проведено за допомогою метрики помилки відстані (distance error), яка відображає середнє відхилення координат досягнутої точки від заданої цільової. У таблиці (табл. 8) наведено результати трьох протестованих моделей: нейронної мережі, навченої за методом найменших квадратів, класичної MDN та модифікованої MDN.

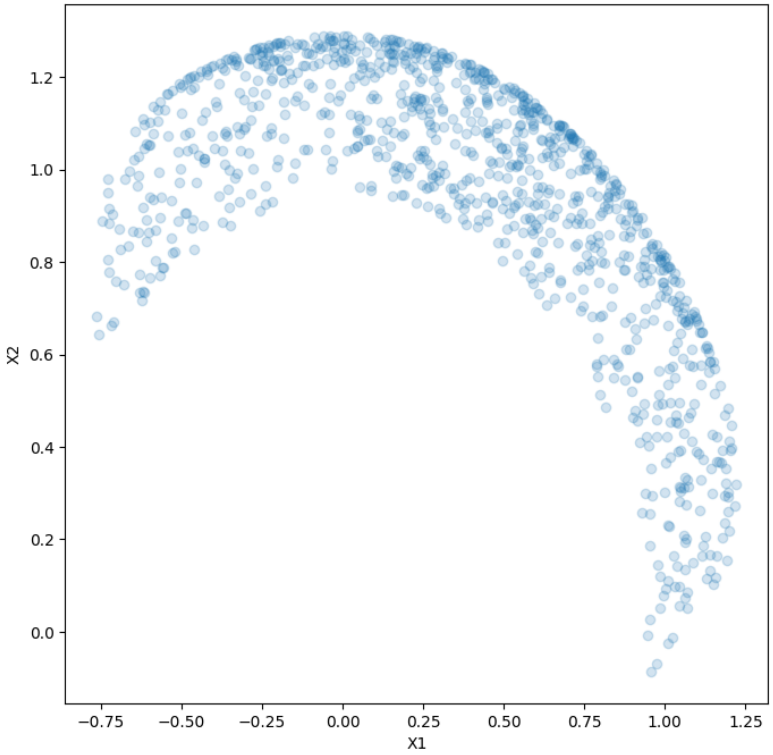


Рис. 21. Візуалізація згенерованих координат цільових точок для досягнення вершиною роботизованої руки

Таблиця 8

Результати для задачі зворотної кінематики двосегментного маніпулятора

Метрика	Least Squares	Класична MDN	Модифікована MDN	Покращення модифікованої моделі (%)
Середня помилка відстані (на тестових даних)	0.154	0.121	0.107	+30.584 (проти LS), +11.573 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	182.645	115.610	107.909	+40.919 (LS), +6.661 (MDN)

Порівняння динаміки зміни значень похибки відстані під час навчання усіх моделей можемо спостерігати на рис. 22.

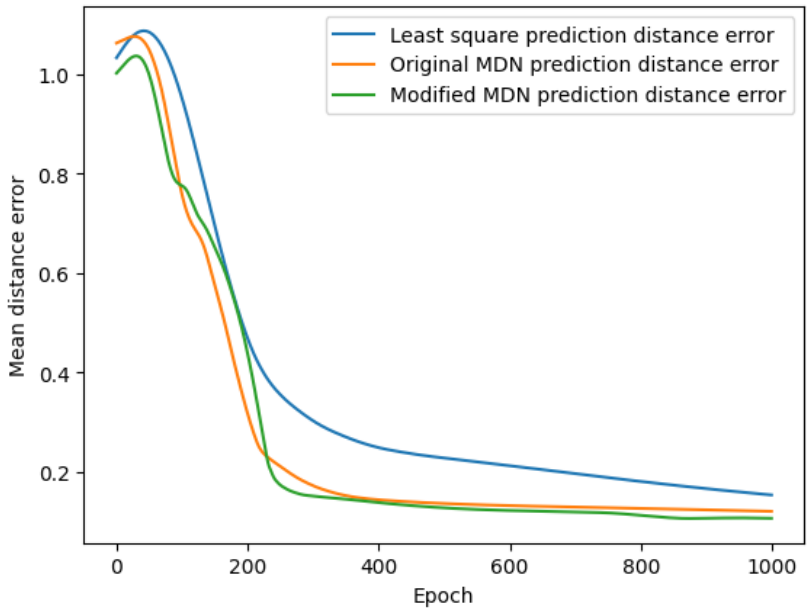


Рис. 22. Динаміка значень похибки відстані для усіх моделей

Модифікована версія MDN показала найкращі результати за всіма метриками. Зокрема, вона знизила середню помилку передбачення на понад 30% у порівнянні з класичним методом найменших квадратів та на понад 11% у порівнянні з класичною MDN. Результати другого експерименту підтверджують, що модифікована MDN-модель є більш ефективною для задач зворотної кінематики, де вихід має мультимодальний характер. У порівнянні як з класичними MDN, так і з простими регресійними підходами, запропонована модель забезпечує вищу точність**,** кращу узгодженість прогнозів і стійкість під час навчання. Це робить її перспективним інструментом для широкого кола задач, що передбачають неоднозначні залежності між входом і виходом, зокрема в робототехніці, системах керування та комп’ютерному моделюванні.

4.1.3. *Експеримент з двома роботизованими маніпуляторами*

У третьому експерименті поставлено більш складну задачу, що є розширенням попереднього випадку з одним маніпулятором. Тут розглядається система з двома однаковими роботизованими руками, кожна з яких складається з двох сегментів.

Обидві руки працюють в одній площині, але мають різні базові положення:

- Перша рука починається в точці (0,0).
- Друга має зсув (offset) на одиницю вздовж осі X_2, тобто її база – (1,0).
- Крім того, кути обертання другої руки мають протилежний напрям обертання.

Таким чином, мета полягає у передбаченні чотирьох кутів обертання (по два для кожної руки) за заданими координатами двох цільових точок, до яких мають дотягнутися кінці обох маніпуляторів. Це значно підвищує складність задачі, оскільки модель має навчитися одночасно моделювати два незалежні мультимодальні зв’язки.

Усі моделі, що брали участь в експерименті, мали подібну архітектуру: один прихований шар із 20 нейронами та функцією активації tanh, оптимізатор Adam і повну обробку пакета даних. MDN-моделі використовували по дві гаусові компоненти в змішаній моделі, з відповідно 24 вихідними нейронами (для параметризації змішання). Модель, навчена методом найменших квадратів, мала 4 вихідні нейрони. Усі моделі навчались протягом 1000 епох. Згенерований набір даних складався з 1000 прикладів, які було поділено на тренувальну (80%) та тестову (20%) вибірки.

Процес генерації даних передбачав створення випадкових комбінацій кутів обертання для обох маніпуляторів у межах допустимих значень. Для кожної комбінації обчислювались координати кінцевих точок обох рук згідно з відповідними тригонометричними формулами. Зокрема, координати другої руки враховували додатковий зсув по осі X_2 та протилежний напрямок обертання. Таким чином, формувався набір у форматі: вхід – координати двох точок (4 значення), вихід – кути обертання чотирьох суглобів (4 значення). Через наявність двох незалежних мультимодальних залежностей, ця задача створює значне навантаження на здатність моделі до узагальнення

та точного моделювання розподілу. Візуалізацію результуючого датасету можемо спостерігати на рис. 23.

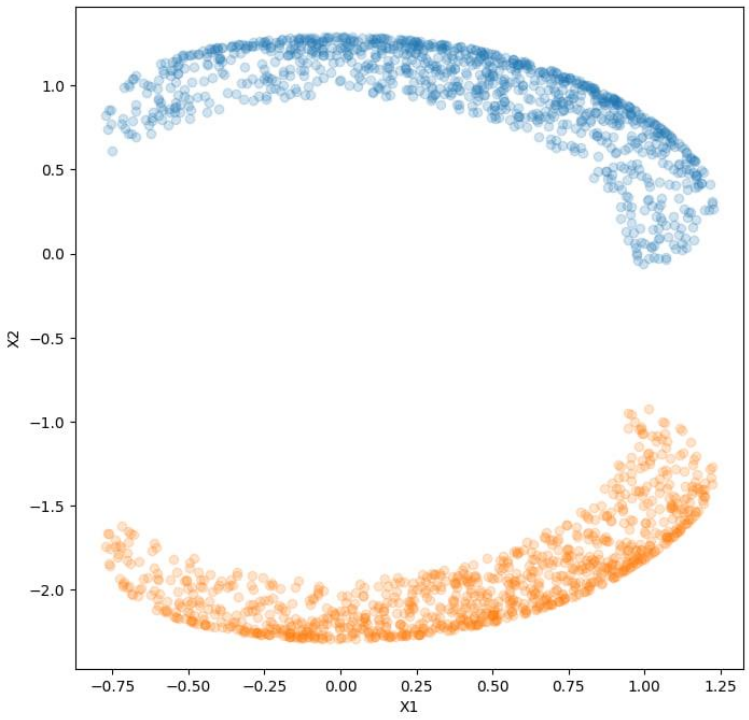


Рис. 23. Візуалізація згенерованих координат цільових точок для досягнення вершиною роботизованої руки

У цьому експерименті точність роботи моделей оцінювалась за тією ж метрикою, що й у попередньому – помилкою відстані (distance error), яка вимірює середнє відхилення досягнутих координат від цільових. Результати представлені в таблиці нижче (табл. 9).

В результаті модифікована MDN-модель знову показала кращі результати, проте покращення у порівнянні з класичною MDN є менш вираженим, ніж у попередньому експерименті. Середня помилка знизилася лише на 2.5%, що свідчить про високу складність задачі та близький потенціал обох варіантів MDN у таких умовах.

Третій експеримент засвідчив, що навіть за умов значного ускладнення задачі – необхідності одночасного передбачення мультимодальних залежностей для двох маніпуляторів – модифікована MDN-модель зберігає свою ефективність.

Таблиця 9

Результати для задачі з двома роботизованими маніпуляторами

Метрика	Least Squares	Класична MDN	Модифікована MDN	Покращення модифікованої моделі (%)
Середня помилка відстані (на тесті)	1.113	1.046	1.020	+8.392 (проти LS), +2.467 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	944.796	847.868	835.355	+11.584 (LS), +1.476 (MDN)

Хоч покращення в порівнянні з класичною MDN є менш вираженим, вона все ж демонструє стабільнішу динаміку навчання**,** меншу помилку передбачення та вищу узагальнюючу здатність**.** Найбільш суттєва перевага моделі спостерігається у порівнянні з класичним регресійним підходом, який не здатен повноцінно враховувати неоднозначність вихідних залежностей. Отже, результати експерименту підтверджують доцільність застосування модифікованої MDN-моделі в задачах підвищеної складності, що включають кілька взаємопов’язаних мультимодальних підзадач.

4.2. Аналіз результатів та подальші напрямки дослідження

Аналіз результатів трьох проведених експериментів підтвердив ефективність запропонованої модифікації MDN-моделі для задач з мультимодальним характером вихідних даних. У всіх розглянутих сценаріях – як у штучно згенерованій задачі, так і в прикладних задачах зворотної кінематики – модифікована модель демонструвала стабільне зниження помилки передбачення порівняно з класичними підходами, особливо помітне у фазі стабільного навчання. Найбільше покращення спостерігалося у задачах середньої складності, де мультимодальність виражена, але структура моделі здатна її повноцінно охопити. У найбільш

складному випадку (передбачення для двох маніпуляторів) переваги модифікації залишалися, хоча й зменшились, що свідчить про потенційну необхідність подальшого вдосконалення архітектури або збільшення кількості компонент у суміші.

На основі отриманих результатів можна окреслити кілька перспективних напрямів подальшого дослідження:

- Адаптація моделі до задач з більш високою розмірністю та складною структурою мультимодальних розподілів шляхом збільшення кількості гаусових компонент або використання ієрархічних MDN.
- Оптимізація гіперпараметрів, зокрема регуляризаційного коефіцієнта, із застосуванням методів автоматичного налаштування.
- Розширення типів задач, зокрема розгляд послідовностей (серії координат або часових рядів) для прогнозування динаміки систем із неоднозначними станами.
- Інтеграція з іншими типами нейронних архітектур, такими як рекурентні чи трансформери, для моделювання мультимодальних залежностей у контексті часових або просторових даних.

Таким чином, запропонована модель має значний потенціал для подальшого розвитку та застосування в широкому спектрі задач, пов’язаних із неоднозначністю та варіативністю вихідної інформації.

4.3. Висновки до четвертого розділу

У межах цього розділу було проведено серію експериментів, спрямованих на оцінку ефективності запропонованого модифікованого підходу на базі мереж з ймовірнісним виходом (MDN) у задачах з мультимодальною природою даних. Результати трьох незалежних експериментів – від синтетичної задачі до прикладних сценаріїв зворотної кінематики – продемонстрували переваги модифікованої моделі у порівнянні

як із класичною MDN, так і з традиційною регресією на основі методу найменших квадратів.

У всіх розглянутих випадках запропонований підхід забезпечив нижчі значення помилок передбачення та кращу стабільність навчання. Найбільш суттєві переваги спостерігалися у задачах середньої складності, де модель мала змогу повноцінно виразити мультимодальну структуру даних. Навіть у складних багатовимірних задачах модифікована модель зберігала свою ефективність, що свідчить про її універсальність та добру здатність до узагальнення.

Таким чином, отримані результати підтверджують доцільність використання модифікованої MDN-моделі як ефективного інструменту для задач, де присутня неоднозначність зв’язку між вхідними та вихідними параметрами. Це відкриває перспективи для подальшого застосування моделі в галузях, що вимагають роботи з невизначеністю, таких як робототехніка, прогнозування, інженерія керування та інші суміжні сфери.

ВИСНОВКИ

У даній магістерській дисертації розглянуто актуальну задачу прогнозування даних із мультимодальним розподілом, яка є важливою для сучасних застосувань у галузях комп'ютерного зору, обробки природної мови, робототехніки та інших сфер, де часто виникає неоднозначність або кілька допустимих варіантів відповіді. Проведено всебічний аналіз предметної області, наведено численні приклади реальних задач з мультимодальною природою, а також критично розглянуто існуючі методи, серед яких особливу увагу приділено ймовірнісним моделям, зокрема GMM, MDN, VAE та їхнім модифікаціям.

Основним науковим внеском роботи є розробка та обґрунтування нового підходу до прогнозування на основі мережі змішаних щільностей (MDN), який полягає у модифікації функції втрат. Замість класичної від'ємної логарифмічної правдоподібності запропоновано оцінювати ймовірність потрапляння у вузький інтервал навколо точки спостереження. Це дозволило досягти кращої інтерпретованості функції втрат, стабільності процесу навчання та адаптивного контролю точності за допомогою гіперпараметра ε.

Проведено реалізацію методу за допомогою сучасних засобів програмування на Python із використанням відповідних бібліотек глибокого навчання. Експериментальне тестування продемонструвало доцільність запропонованого підходу: він забезпечує більш точне моделювання мультимодального розподілу, покращену якість передбачень у порівнянні з класичним MDN та кращу узгодженість з ймовірнісною природою задачі.

Результати дослідження підтверджують, що запропонована модифікація MDN є перспективною для широкого спектру задач, де вихідна змінна має невизначену або багатозначну природу. Окрім підвищення точності та стійкості, метод зберігає сумісність із класичною архітектурою MDN, що дозволяє легко інтегрувати його в існуючі системи. Отримані

результати відкривають подальші можливості для вдосконалення ймовірнісних моделей, зокрема за рахунок адаптивної оцінки інтервалів, використання фізичних обмежень або застосування у високовимірних просторах.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Bishop C. M. Mixture Density Networks. Technical Report NCRG/94/004, Neural Computing Research Group, Aston University, 1994. URL: https://publications.aston.ac.uk/373/1/NCRG_94_004.pdf (дата звернення: 05.05.2025).
2. Alnuaimi A.F.A.H., Albaldawi T.H.K. An overview of machine learning classification techniques. BIO Web of Conferences, т. 97, 00133, 2024. URL: https://doi.org/10.1051/bioconf/20249700133 (дата звернення: 05.05.2025).
3. Sarvandani M.H. Top Applications of Classification in Machine Learning [Електронний ресурс] — Режим доступу до ресурсу: https://medium.com/@mohamadhasan.sarvandani/top-applications-of-classification-in-machine-learning-e7b4351f64eb (дата звернення: 05.05.2025).
4. Statistics How To. Multimodal Distribution Definition and Examples [Електронний ресурс] — Режим доступу до ресурсу: https://www.statisticshowto.com/multimodal-distribution/ (дата звернення: 05.05.2025).
5. StatisticalPoint. What is a Multimodal Distribution? [Електронний ресурс] — Режим доступу до ресурсу: https://statisticalpoint.com/multimodal-distribution/ (дата звернення: 05.05.2025).
6. Nolen J., Pavliotis G.A., Stuart A.M. Multiscale Modelling and Inverse Problems. Lecture Notes in Computational Science and Engineering, Vol. 83. Springer, 2012. URL: https://www.ma.imperial.ac.uk/~pavl/NolenPavliotisStuart.pdf (дата звернення: 05.05.2025).
7. American Heart Association. The differences between flu, COVID, strep throat and RSV [Електронний ресурс] — Режим доступу до ресурсу:

https://www.heart.org/en/health-topics/house-calls/differences-between-flu-covid-strep-throat-and-rsv (дата звернення: 05.05.2025).

8. Xu J., Chen X., Lan X., Zheng N. Probabilistic Human Motion Prediction via A Bayesian Neural Network. arXiv preprint arXiv:2107.06564, 2021. URL: https://arxiv.org/abs/2107.06564 (дата звернення: 05.05.2025).
9. Prasad V., Kshirsagar A., Koert D., Stock-Homburg R., Peters J., Chalvatzaki G. MoVEInt: Mixture of Variational Experts for Learning Human-Robot Interactions from Demonstrations. arXiv preprint arXiv:2407.07636, 2024. URL: https://arxiv.org/abs/2407.07636 (дата звернення: 05.05.2025).
10. Moitra A. Gaussian Mixture Models. Lecture Notes for 18.409 Algorithmic Aspects of Machine Learning, Massachusetts Institute of Technology, Spring 2015. URL: https://ocw.mit.edu/courses/18-409-algorithmic-aspects-of-machine-learning-spring-2015/e339520c4069ca5e785b29a3c604470e_MIT18_409S15_chapp6.pdf (дата звернення: 05.05.2025).
11. Ghahramani Z. Bayesian Modelling. Machine Learning Summer School (MLSS), м. Ла-Пальма, Іспанія, 2012. URL: https://mlg.eng.cam.ac.uk/zoubin/talks/lect1bayes.pdf (дата звернення: 05.05.2025).
12. Liu P. Multimodal Regression — Beyond L1 and L2 Loss [Електронний ресурс] — Режим доступу до ресурсу: https://medium.com/data-science/anchors-and-multi-bin-loss-for-multi-modal-target-regression-647ea1974617 (дата звернення: 05.05.2025).
13. Cohen Kalafut N., Huang X., Wang D. Joint Variational Autoencoders for Multimodal Imputation and Embedding. Nature Machine Intelligence, 2023, т. 5, с. 631–642. URL: https://www.nature.com/articles/s42256-023-00663-z (дата звернення: 05.05.2025).
14. Chen J., Yu Y., Liu Y. Physics-guided mixture density networks for uncertainty quantification. Reliability Engineering and System Safety. 2022.

Vol. 228. Article 108823. URL: https://doi.org/10.1016/j.ress.2022.108823 (дата звернення: 05.05.2025).

15. Rahimi A., Alahi A. A Multi-Loss Strategy for Vehicle Trajectory Prediction: Combining Off-Road, Diversity, and Directional Consistency Losses. arXiv preprint arXiv:2411.19747, 2024. URL: https://arxiv.org/abs/2411.19747 (дата звернення: 05.05.2025).
16. Qian Z., et al. Bridging the Gap between Sequence and Structure Classifications of Proteins. bioRxiv, 2023. URL: https://pubmed.ncbi.nlm.nih.gov/39197652/ (дата звернення: 05.05.2025).
17. Dempster A.P., Laird N.M., Rubin D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 1977, т. 39, № 1, с. 1–38. URL: https://www.jstor.org/stable/2984875 (дата звернення: 05.05.2025).
18. Burden R.L., Faires J.D. Numerical Analysis. 9th ed. Boston: Brooks/Cole, Cengage Learning, 2011. Розділ 4.4 — The Composite Simpson’s Rule. URL: https://archive.org/details/NumericalAnalysis9thEdition (дата звернення: 05.05.2025).
19. Kroese D.P., Taimre T., Botev Z.I. Handbook of Monte Carlo Methods. Wiley Series in Probability and Statistics. John Wiley & Sons, 2011. URL: https://people.smp.uq.edu.au/DirkKroese/ps/montecarlo.pdf (дата звернення: 05.05.2025).
20. Makansi O., Ilg E., Çiçek Ö., Brox T. Overcoming Limitations of Mixture Density Networks: A Sampling and Fitting Framework for Multimodal Future Prediction. arXiv preprint arXiv:1906.03631, 2019. URL: https://arxiv.org/abs/1906.03631 (дата звернення: 05.05.2025).
21. Comaniciu D., Meer P. Mean Shift: A Robust Approach Toward Feature Space Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, т. 24, № 5, с. 603–619. URL: https://doi.org/10.1109/34.1000236 (дата звернення: 05.05.2025).

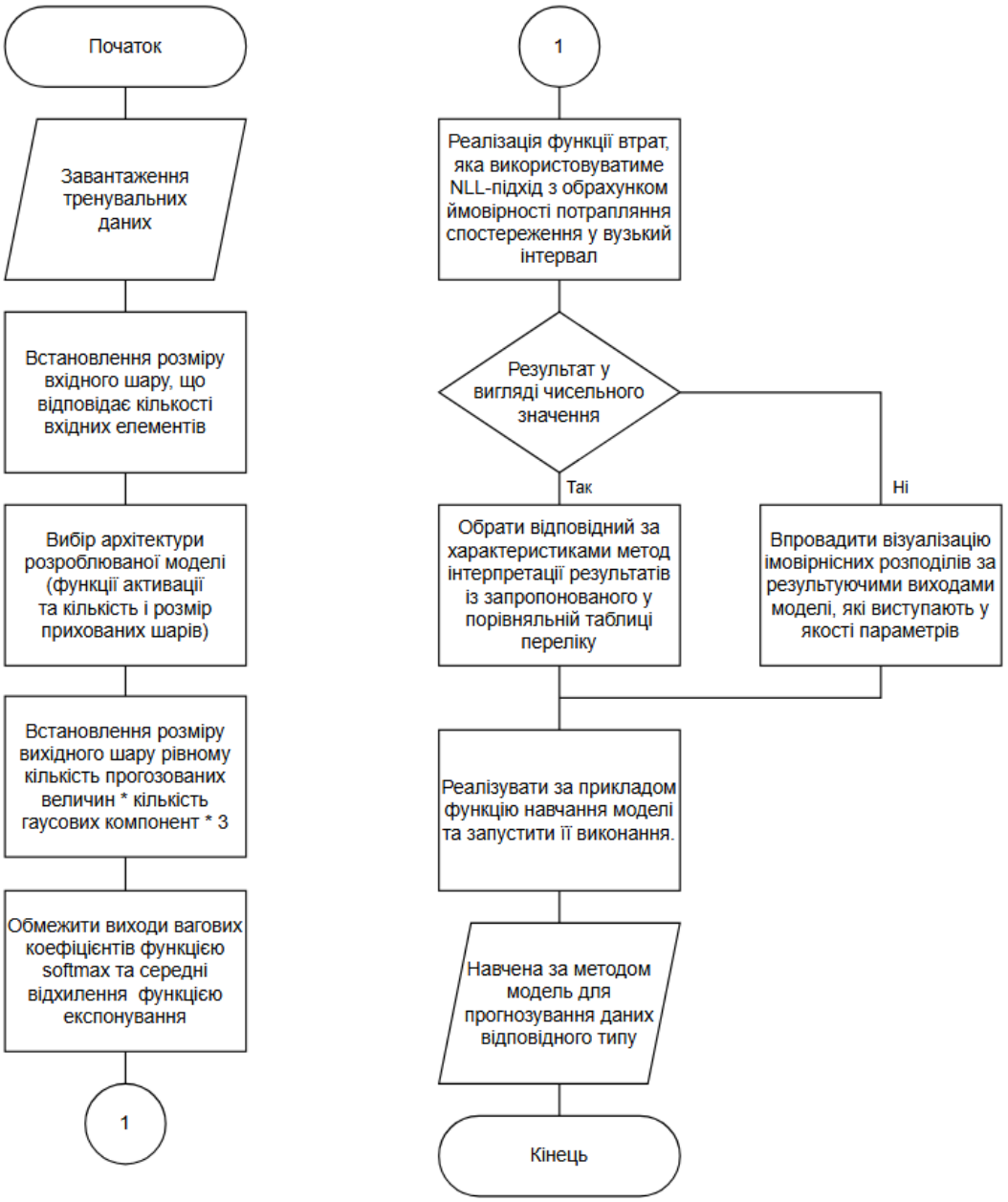
22. Ester M., Kriegel H.-P., Sander J., Xu X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), м. Portland, OR, USA, 2–4 серп. 1996 р. 1996. URL: https://www.dbs.ifi.lmu.de/Publikationen/Papers/KDD-96.final.frame.pdf (дата звернення: 05.05.2025).
23. Fraley C., Raftery A.E. Model-Based Clustering, Discriminant Analysis, and Density Estimation. Journal of the American Statistical Association, 2002, т. 97, № 458, с. 611–631. URL: https://doi.org/10.1198/016214502760047131 (дата звернення: 05.05.2025).
24. Stanford University. Gradient Ascent [Електронний ресурс] — Режим доступу до ресурсу: https://web.stanford.edu/class/archive/cs/cs109/cs109.1192/lectures/22%20-%20GradientAscent.pdf (дата звернення: 05.05.2025).
25. Scrucca L. A Fast and Efficient Modal EM Algorithm for Gaussian Mixtures. arXiv preprint arXiv:2002.03600, 2020. URL: https://arxiv.org/abs/2002.03600 (дата звернення: 05.05.2025).
26. Li Q., Jain A., Abbeel P. AdaCat: Adaptive Categorical Discretization for Autoregressive Models. Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI 2022), м. Ейндговен, Нідерланди, 1–5 серп. 2022 р. 2022. URL: https://proceedings.mlr.press/v180/li22h.html (дата звернення: 05.05.2025).

ДОДАТКИ

Додаток 1

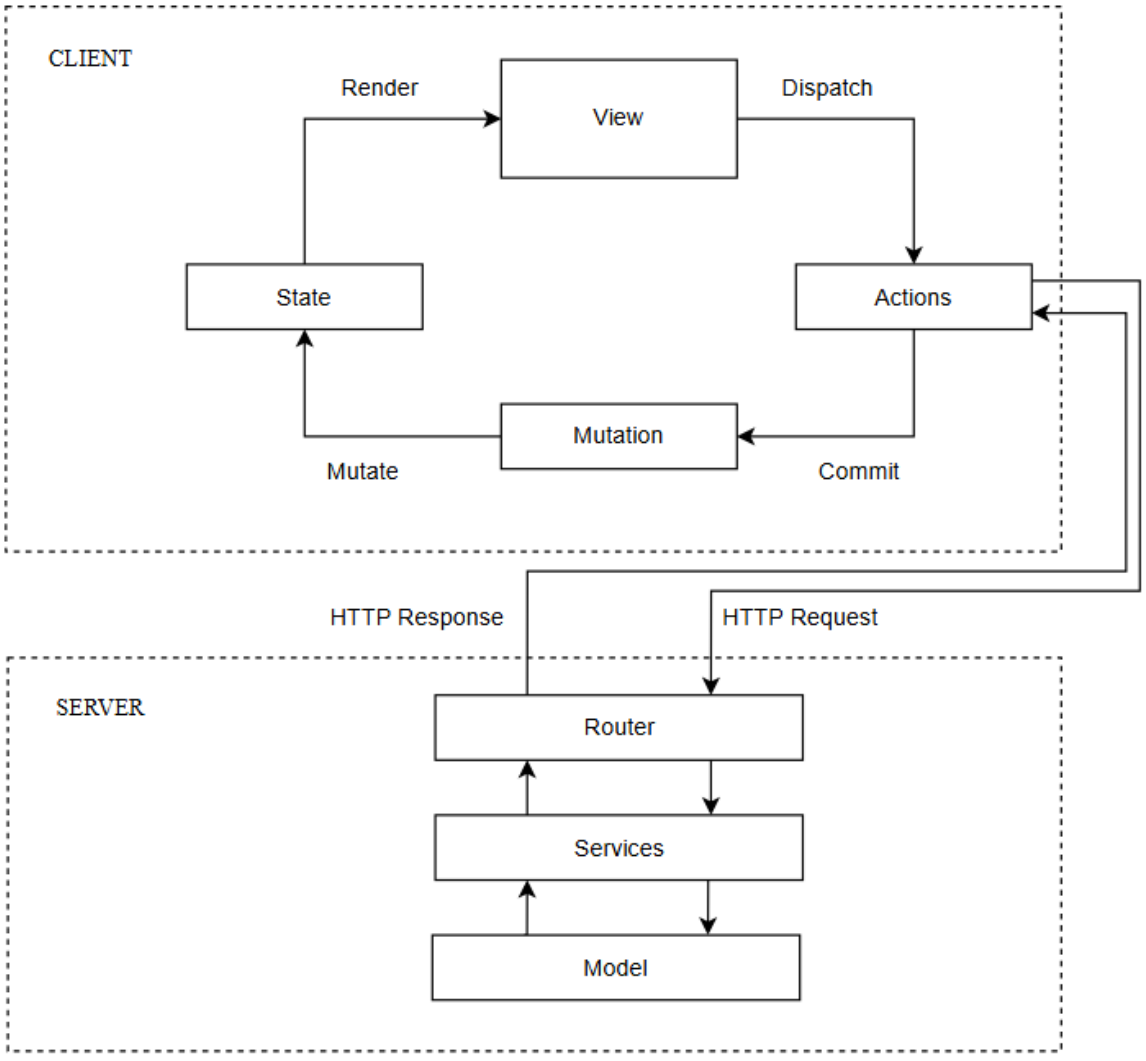
Копії графічних матеріалів

Блок-схема алгоритму побудови моделі нейронної за запропонованим методом



Федорчук Іван, КП-31мн

Архітектура запропонованого програмного забезпечення



Федорчук Іван, КП-31мн

Порівняльна таблиця методів наближеного обчислення визначеного інтегралу для суміші Гаусових розподілів

Метод	Точність	Швидкодія	Складність реалізації	Придатність для вузьких інтервалів	Обчислювальні витрати
Метод прямокутників	Низька	Висока	Низька	Низька	Низькі
Метод трапецій	Середня	Середня	Низька	Середня	Низькі
Метод Сімпсона	Висока	Середня	Середня	Середня	Середні
Монте-Карло (класичний)	Низька	Низька	Низька	Низька	Високі
Монте-Карло (геометричний)	Низька	Низька	Низька	Низька	Високі
Аналітичний метод	Дуже висока	Дуже висока	Низька (за наявності бібліотек)	Висока	Низькі

Федорчук Іван, КП-31мн

Порівняльна таблиця методів інтерпретації результатів запропонованого методу за ключовими критеріями

Метод	Швидкодія	Обчисл. ресурси	Точність	Здатність знаходити усі моди	Ефективність у високій розмірності	Робота з багатьма компонентами	Стабільність
Семплювання з кластеризацією	Низька	Висока	Висока	Висока	Низька	Залежить від реалізації	Висока
Градієнтний підйом	Висока	Низька	Обмежена	Обмежена	Низька	Низька	Залежить
Mean-Shift	Середня	Середня	Висока	Висока	Низька	Середня	Залежить
EM-подібний пошук	Висока	Низька	Середня	Обмежена	Середня	Низька	Залежить
Адаптивна дискретизація	Середня	Висока	Висока	Висока	Низька	Висока	Висока
Інтервальний аналіз	Середня	Середня	Висока	Висока	Низька	Висока	Висока

Федорчук Іван, КП-31мн

Порівняльна таблиця результатів оцінки запропонованого методу при розв’язанні різних задач

Результати для задачі прогнозування синтетичних даних

Метрика	Класична MDN	Модифікована MDN	Покращення модифікованої моделі
Остаточна втрата (loss) на тренувальному наборі	1.208	1.152	+4.559%
Остаточна втрата на тестовому наборі	1.173	1.120	+4.561%
Кумулятивна втрата (епохи 500–1500) – тренувальний набір	1337.074	1304.965	+2.401%
Кумулятивна втрата – тестовий набір	1318.767	1291.521	+2.066%

Результати для задачі зворотної кінематики двосегментного маніпулятора

Метрика	Least Squares	Класична MDN	Модифікована MDN	Покращення модифікованої моделі (%)
Середня помилка відстані (на тестових даних)	0.154	0.121	0.107	+30.584 (проти LS), +11.573 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	182.645	115.610	107.909	+40.919 (LS), +6.661 (MDN)

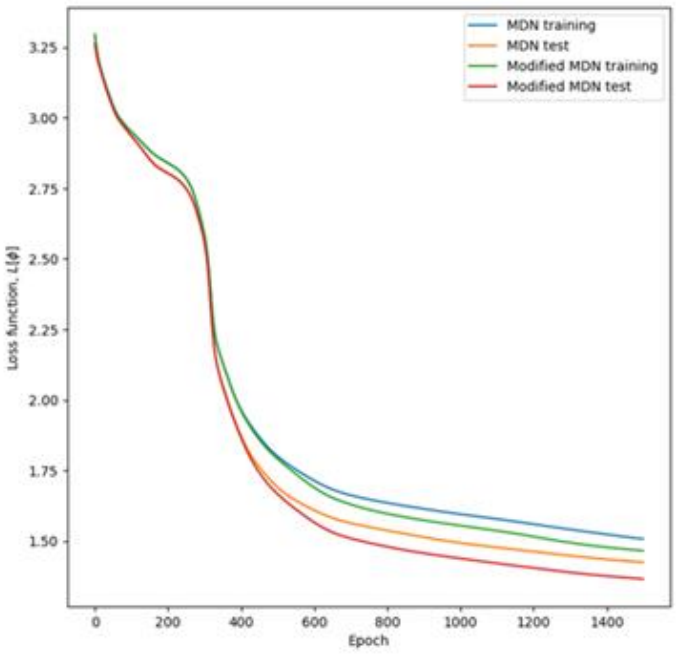
Результати для задачі з двома роботизованими маніпуляторами

Метрика	Least Squares	Класична MDN	Модифікована MDN	Покращення модифікованої моделі (%)
Середня помилка відстані (на тесті)	1.113	1.046	1.020	+8.392 (проти LS), +2.467 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	944.796	847.868	835.355	+11.584 (LS), +1.476 (MDN)

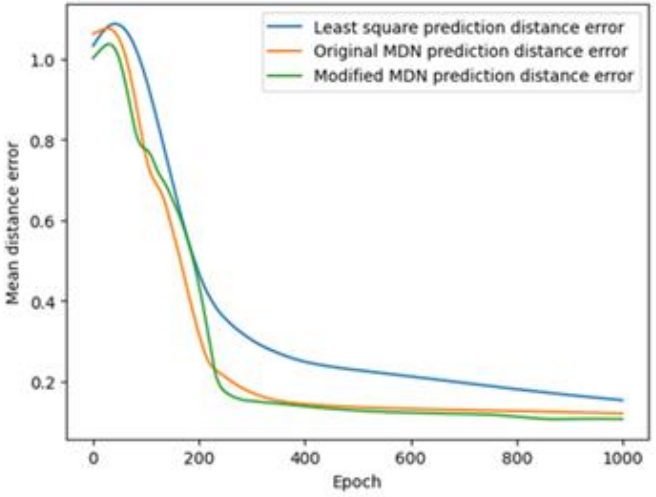
Федорчук Іван, КП-31мн

Графіки динамік змін метрик під час навчання моделей для вирішення різних задач

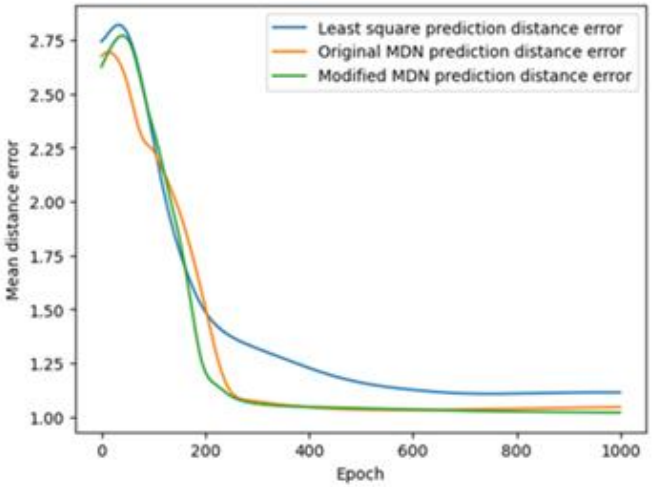
Порівняння динаміки навчання для оригінальної та модифікованої моделей MDN з точки зору покращення значень втрат для навчальних та тестових даних



Порівняння динаміки навчання для розглянутих моделей з точки зору покращення значень похибки відстані для задачі зворотної кінематики двосегментного маніпулятора



Порівняння динаміки навчання для розглянутих моделей з точки зору покращення значень похибки відстані для задачі зворотної кінематики двох двосегментних маніпуляторів



Федорчук Іван КП-31мн

Додаток 2

Лістинг програми

```
import matplotlib.pyplot as plt # creating visualizations
import numpy as np # basic math and random numbers
import torch # package for building functions with learnable parameters
import torch.nn as nn # prebuilt functions specific to neural networks
from torch.autograd import Variable # storing data while learning
from torch.distributions import Normal
import math
def generate_data(n_samples):
    samples_variance = int(n_samples/2)
    epsilon1 = np.random.normal(size=(samples_variance))
    epsilon2 = np.random.normal(size=(samples_variance))
    x_data = np.random.uniform(-15, 15, samples_variance)
    phi_0, phi_1 = 3, 15
    y_data1 = np.sin(phi_0+0.06*phi_1*x_data) * np.exp(-
np.square(phi_0+0.06*phi_1*x_data)/32) + epsilon1*0.1
    y_data2 = np.log(x_data+15) + epsilon2*0.1
    y_data = np.concatenate((y_data1, y_data2), axis=0)
    x_data = np.concatenate((x_data, x_data), axis=0)
    return x_data, y_data

n_samples = 1000
n_training_samples = int(n_samples*0.8)
n_test_samples = n_samples-int(n_samples*0.8)
integral_epsilon = 0.15
x_data, y_data = generate_data(n_samples)

indices = np.random.permutation(x_data.shape[0])
training_idx, test_idx = indices[:n_training_samples],
indices[n_training_samples:]
x_training, x_test = x_data[training_idx], x_data[test_idx]
y_training, y_test = y_data[training_idx], y_data[test_idx]

plt.figure(figsize=(8, 8))
plt.scatter(x_training, y_training, alpha=0.2)
plt.xlabel('Input')
plt.ylabel('Output')
plt.show()
plt.figure(figsize=(8, 8))
plt.scatter(x_test, y_test, alpha=0.2)
plt.xlabel('Input')
plt.ylabel('Output')
plt.show()

class MixOfTwoGaussian(nn.Module):
    def __init__(self, n_hidden, n_gaussians):
        super(MixOfTwoGaussian, self).__init__()
        self.z_h = nn.Sequential(
            nn.Linear(1, n_hidden),
            nn.Tanh()
        )
        self.z_lambda = nn.Linear(n_hidden, n_gaussians)
        self.z_sigma = nn.Linear(n_hidden, n_gaussians)
        self.z_mu = nn.Linear(n_hidden, n_gaussians)

    def forward(self, x):
        z_h = self.z_h(x)
        _lambda = nn.functional.softmax(self.z_lambda(z_h), -1)
        _sigma = torch.exp(self.z_sigma(z_h))
        _mu = self.z_mu(z_h)
        return _lambda, _sigma, _mu

class MDN(nn.Module):
    def __init__(self, n_hidden, n_gaussians):
        super(MDN, self).__init__()
```

```
        self.z_h = nn.Sequential(
            nn.Linear(1, n_hidden),
            nn.Tanh()
        )
        self.z_lambda = nn.Linear(n_hidden, n_gaussians)
        self.z_sigma = nn.Linear(n_hidden, n_gaussians)
        self.z_mu = nn.Linear(n_hidden, n_gaussians)

    def forward(self, x):
        z_h = self.z_h(x)
        pi = nn.functional.softmax(self.z_lambda(z_h), -1)
        sigma = torch.exp(self.z_sigma(z_h))
        mu = self.z_mu(z_h)
        return pi, sigma, mu

oneDivSqrtTwoPI = 1.0 / np.sqrt(2.0*np.pi)
def gaussian_distribution(y, mu, sigma):
    result = (y.expand_as(mu) - mu) * torch.reciprocal(sigma)
    result = -0.5 * (result * result)
    return (torch.exp(result) * torch.reciprocal(sigma)) * oneDivSqrtTwoPI

def gaussian_probability(y, mu, sigma, epsilon):
    normal_distribution = Normal(mu, sigma)

    upper = y + epsilon
    lower = y - epsilon

    upper = upper.expand_as(mu)
    lower = lower.expand_as(mu)

    result = normal_distribution.cdf(upper) - normal_distribution.cdf(lower)

    return result

def modified_mdn_loss_fn(pi, sigma, mu, y, epsilon):
    result = gaussian_probability(y, mu, sigma, epsilon) * pi
    result = torch.sum(result, dim=1)
    result = -torch.log(result)
    return torch.mean(result)

def mdn_loss_fn(pi, sigma, mu, y):
    result = gaussian_distribution(y, mu, sigma) * pi
    result = torch.sum(result, dim=1)
    result = -torch.log(result)
    return torch.mean(result)

modified_MDN_model = MDN(n_hidden=20, n_gaussians=2)
MDN_model = MDN(n_hidden=20, n_gaussians=2)

modified_MDN_model.load_state_dict(MDN_model.state_dict())
epochs = 1500

optimizer = torch.optim.Adam(modified_MDN_model.parameters())
mdn_optimizer = torch.optim.Adam(MDN_model.parameters())

x_traning_tensor =
torch.from_numpy(np.float32(x_training).reshape(n_training_samples, 1))
y_traning_tensor =
torch.from_numpy(np.float32(y_training).reshape(n_training_samples, 1))
x_variable = Variable(x_traning_tensor, requires_grad=True)
y_variable = Variable(y_traning_tensor, requires_grad=True)

x_test_tensor = torch.from_numpy(np.float32(x_test).reshape(n_test_samples,
1))
```

```
y_test_tensor = torch.from_numpy(np.float32(y_test).reshape(n_test_samples,
1))
x_test_variable = Variable(x_test_tensor, requires_grad=False)
y_test_variable = Variable(y_test_tensor, requires_grad=False)

loss_story_training = []
loss_story_test = []

mdn_loss_story_training = []
mdn_loss_story_test = []

def train():
    for epoch in range(epochs):
        pi_variable, sigma_variable, mu_variable =
modified_MDN_model(x_variable)
        loss = modified_mdn_loss_fn(pi_variable, sigma_variable, mu_variable,
y_variable, integral_epsilon)
        pi_variable, sigma_variable, mu_variable =
modified_MDN_model(x_test_variable)
        loss_test = modified_mdn_loss_fn(pi_variable, sigma_variable,
mu_variable, y_test_variable, integral_epsilon)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        loss_story_training.append(loss.data.item())
        loss_story_test.append(loss_test.data.item())
        if epoch % 500 == 0:
            print(epoch, loss_test.data.item())

train()

def train_mdn():
    for epoch in range(epochs):
        pi_variable, sigma_variable, mu_variable = MDN_model(x_variable)
        loss = mdn_loss_fn(pi_variable, sigma_variable, mu_variable,
y_variable)
        loss_training = modified_mdn_loss_fn(pi_variable, sigma_variable,
mu_variable, y_variable, integral_epsilon)
        pi_variable, sigma_variable, mu_variable = MDN_model(x_test_variable)
        loss_test = modified_mdn_loss_fn(pi_variable, sigma_variable,
mu_variable, y_test_variable, integral_epsilon)
        mdn_optimizer.zero_grad()
        loss.backward()
        mdn_optimizer.step()

        mdn_loss_story_training.append(loss_training.data.item())
        mdn_loss_story_test.append(loss_test.data.item())
        if epoch % 500 == 0:
            print(epoch, loss_test.data.item())

train_mdn()

plt.figure(figsize=(8, 8))
plt.plot(range(epochs), np.array(mdn_loss_story_training), label='MDN
training')
plt.plot(range(epochs), np.array(mdn_loss_story_test), label='MDN test')
plt.plot(range(epochs), np.array(loss_story_training), label='Modified MDN
training')
plt.plot(range(epochs), np.array(loss_story_test), label='Modified MDN test')
plt.legend(loc='upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss function, $L[\\phi]$')
# plt.ylim([0,5])
```

```
plt.show()

def percentageLossGain(original, modified):
    return (original - modified) / original * 100

plt.figure(figsize=(8, 8))
plt.plot(range(epochs), list(map(percentageLossGain, mdn_loss_story_training,
loss_story_training)), label='Training')
plt.plot(range(epochs),  list(map(percentageLossGain, mdn_loss_story_test,
loss_story_test)), label='Test')
plt.legend(loc='upper left')
plt.xlabel('Epoch')
plt.ylabel('Modified method accuracy gain (%)')
# plt.ylim([0,5])
plt.show()

m_tr = np.array(loss_story_training)[epochs-1]
m_ts = np.array(loss_story_test)[epochs-1]
mdn_tr = np.array(mdn_loss_story_training)[epochs-1]
mdn_ts = np.array(mdn_loss_story_test)[epochs-1]
m_tr_acc = sum(loss_story_training)
m_ts_acc = sum(loss_story_test)
mdn_tr_acc = sum(mdn_loss_story_training)
mdn_ts_acc = sum(mdn_loss_story_test)

print("Final original taining loss     {:.3f}".format(mdn_tr))
print("Final modified taining loss     {:.3f}".format(m_tr))
print("Loss modified training gain (%) {:.3f}".format(100*(mdn_tr -
m_tr)/mdn_tr))
print("")
print("Final original testing loss    {:.3f}".format(mdn_ts))
print("Final modified testing loss    {:.3f}".format(m_ts))
print("Loss modified testing gain (%) {:.3f}".format(100*(mdn_ts -
m_ts)/mdn_ts))
print("")
print("Total accumulative original training loss {:.3f}".format(mdn_tr_acc))
print("Total accumulative modified training loss {:.3f}".format(m_tr_acc))
print("Loss modified training gain (%)
{:.3f}".format(100*(mdn_tr_acc - m_tr_acc)/mdn_tr_acc))
print("")
print("Total accumulative original testing loss {:.3f}".format(mdn_ts_acc))
print("Total accumulative modified testing loss {:.3f}".format(m_ts_acc))
print("Loss modified testing gain (%)
{:.3f}".format(100*(mdn_ts_acc - m_ts_acc)/mdn_ts_acc))

def getAccumulatedSlice(arr, end):
    return sum(arr[0:end])

step = 500
for index in range(step-1, epochs-1, step):
    mdn_tr_acc = getAccumulatedSlice(mdn_loss_story_training, index)
    m_tr_acc = getAccumulatedSlice(loss_story_training, index)
    mdn_ts_acc = getAccumulatedSlice(mdn_loss_story_test, index)
    m_ts_acc = getAccumulatedSlice(loss_story_test, index)
    print("")
    print("Accumulative original training loss for first {} epochs
{:.3f}".format(index+1, mdn_tr_acc))
    print("Accumulative modified training loss for first {} epochs
{:.3f}".format(index+1, m_tr_acc))
    print("Loss modified training gain (%) {:.3f}".format(100*(mdn_tr_acc -
m_tr_acc)/mdn_tr_acc))
    print("")
    print("Accumulative original testing loss for first {} epochs
{:.3f}".format(index+1, mdn_ts_acc))
```

```
    print("Accumulative modified testing loss for first {} epochs
{:.3f}".format(index+1, m_ts_acc))
    print("Loss modified testing gain (%) {:.3f}".format(100*(mdn_ts_acc -
m_ts_acc)/mdn_ts_acc))
```

Додаток З

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АЛГОРИТМІЧНО-ПРОГРАМНИЙ МЕТОД ДЛЯ ПРОГНОЗУВАННЯ ДАНИХ З МУЛЬТИМОДАЛЬНИМ РОЗПОДІЛОМ НА ОСНОВІ МОДЕЛІ MDN

Доповідач: Федорчук Іван Васильович

Науковий керівник: доцент, к.т.н. Шкурат Оксана Сергіївна

Київ – 2025

Актуальність дослідження

- Мультимодальні розподіли даних є поширеним явищем у багатьох сучасних прикладних задачах – від медичної діагностики до автономного керування.
- Мультимодальна природа деяких даних ускладнює процес їх прогнозування, оскільки традиційні методи часто зводяться до усереднення значень, що не відображає реальної структури.
- Прогнозування даних з мультимодальним розподілом є важливою задачею для багатьох сфер, де рішення приймаються в умовах невизначеності.

Об’єкт, предмет та мета дослідження

Об’єктом дослідження є процес прогнозування даних із мультимодальним розподілом.

Предметом дослідження є методи та програмне забезпечення прогнозування даних з мультимодальним розподілом.

Метою дослідження є підвищення точності прогнозування даних з мультимодальним розподілом та розробка програмного забезпечення на основі моделі машинного навчання архітектури MDN та імовірнісної функції втрат для навчання моделі.

Завдання

1. Провести аналіз існуючих методів та програмних рішень прогнозування даних з мультимодальним розподілом. Виявити їх переваги та недоліки.
2. Розробити алгоритмічно-програмний метод прогнозування даних з мультимодальним розподілом.
3. Провести аналіз результатів роботи методу прогнозування даних з мультимодальним розподілом.
4. Розробити програмне забезпечення, що реалізує запропонований метод прогнозування.
5. Запропонувати шляхи покращення методу.

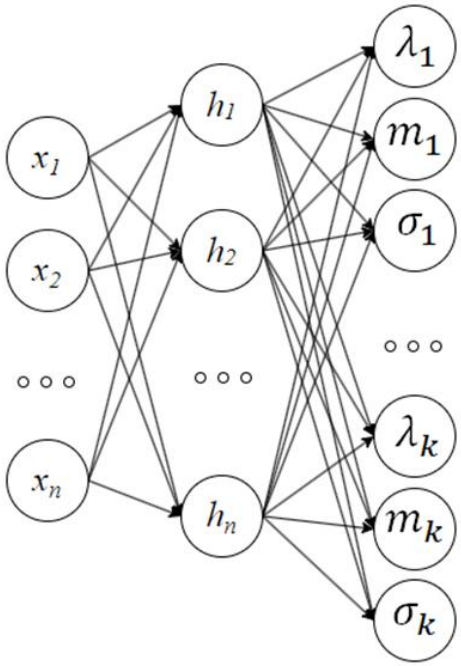
Суміш гаусів

Суміш гаусів (MoG, GMM) – це ймовірнісна модель, яка моделює ймовірнісний розподіл даних як зважену суму кількох нормальних (гаусових) розподілів, кожен з яких описує локальну структуру або підгрупу в даних.

$$p(y) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mu_k, \sigma_k^2) \qquad \sum_{k=1}^{K} \pi_k = 1 \qquad \mathcal{N}\left(y\middle|\mu_k, \sigma_k^2\right) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(y-\mu_k)^2}{2\sigma_k^2}}$$

π_k – ваговий коефіцієнт; $\pi_k \geq 0$.
K – кількість компонентів у суміші.

Модель MDN



$$L[\phi] = \sum_{i=1}^{I} -\log\left(\sum_{k=1}^{K} \lambda_k(x_i,\phi)\mathcal{N}\left(y_i \middle| m_k(x_i,\phi), \sigma_k(x_i,\phi)\right)\right)$$

Запропонований метод

Обрахунок ймовірності потрапляння спостереження у вузький інтервал $[y_1 = y_i - \varepsilon, y_2 = y_i + \varepsilon]$ замість обрахунку точкового значення щільності ймовірності:

$$p(y_1, y_2) = \int_{y_1}^{y_2} \sum_{k=1}^{K} \pi_k(x_i, \phi) \mathcal{N}\left(y_i \middle| \mu_k(x_i, \phi), \sigma_k^2(x_i, \phi)\right) dy$$

Відповідно, маємо модифіковану формулу для функції втрат:

$$L[\phi] = \sum_{i=1}^{I} -\log\left(\int_{y_i-\varepsilon}^{y_i+\varepsilon} \sum_{k=1}^{K} \pi_k(x_i, \phi) \mathcal{N}\left(y_i \middle| m_k(x_i, \phi), \sigma_k^2(x_i, \phi)\right) dy\right)$$

Аналіз методів наближеного обчислення визначеного інтегралу суміші гаусів

Метод	Точність	Швидкодія	Складність реалізації	Придатність для вузьких інтервалів	Обчислювальні витрати
Метод прямокутників	Низька	Висока	Низька	Низька	Низькі
Метод трапецій	Середня	Середня	Низька	Середня	Низькі
Метод Сімпсона	Висока	Середня	Середня	Середня	Середні
Монте-Карло (класичний)	Низька	Низька	Низька	Низька	Високі
Монте-Карло (геометричний)	Низька	Низька	Низька	Низька	Високі
Аналітичний метод	Дуже висока	Дуже висока	Низька (за наявності бібліотек)	Висока	Низькі

Аналіз методів інтепретації результуючого ймовірнісного розподілу

Метод	Швидкодія	Обчисл. ресурси	Точність	Здатність знаходити усі моди	Ефективність у високій розмірності	Робота з багатьма компонентами	Стабільність
Семплювання з кластеризацією	Низька	Висока	Висока	Висока	Низька	Залежить від реалізації	Висока
Градієнтний підйом	Висока	Низька	Обмежена	Обмежена	Низька	Низька	Залежить
Mean-Shift	Середня	Середня	Висока	Висока	Низька	Середня	Залежить
EM-подібний пошук	Висока	Низька	Середня	Обмежена	Середня	Низька	Залежить
Адаптивна дискретизація	Середня	Висока	Висока	Висока	Низька	Висока	Висока
Інтервальний аналіз	Середня	Середня	Висока	Висока	Низька	Висока	Висока

Задача прогнозування для синтетичних даних

Формування датасету:

- Згенеровано 1000 чисел x_i, рівномірно розподілених на інтервалі $[-15, 15]$.
- Для значення x обчислюються два значення y_1 і y_2 згідно формул:

$$y_1 = \sin(\phi_0 + \phi_1 \cdot x) \cdot e^{-\frac{(\phi_0 + \phi_1 \cdot x)^2}{32}} + 0.01 \cdot \epsilon_1,$$

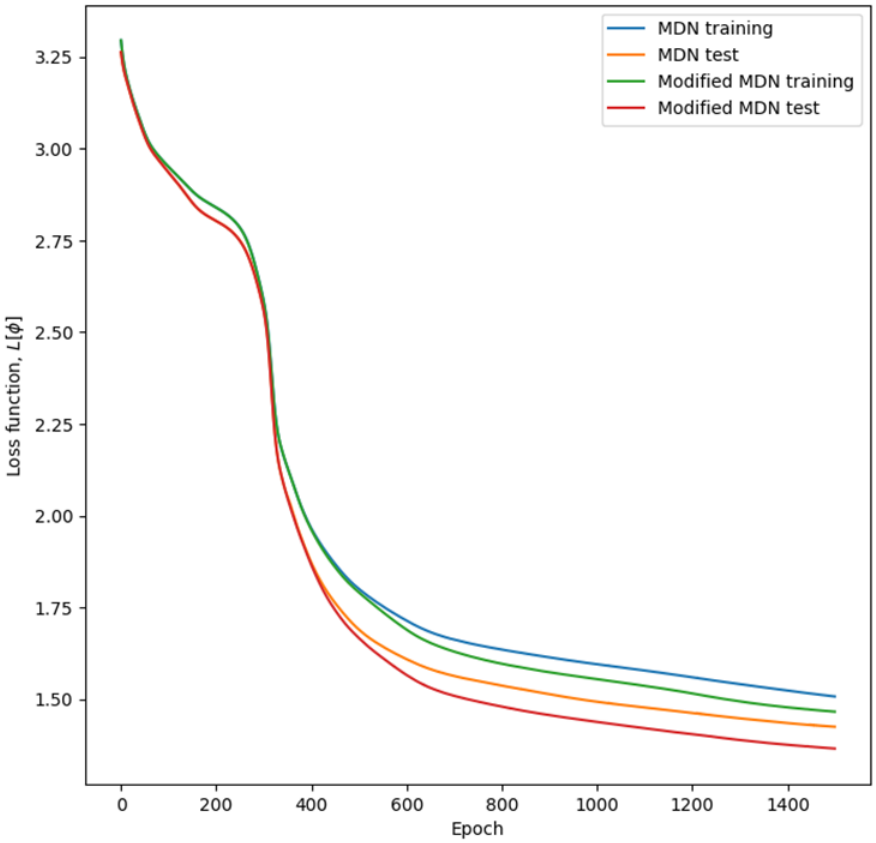
$$y_2 = \log(x + 15) + 0.01 \cdot \epsilon_2,$$

де параметри ϕ_0 та ϕ_1 були встановлені на 3 та 0,9 відповідно. ϵ_1 та ϵ_2 – шумові складові, вибрані з нормального розподілу.

Архітектура моделей:

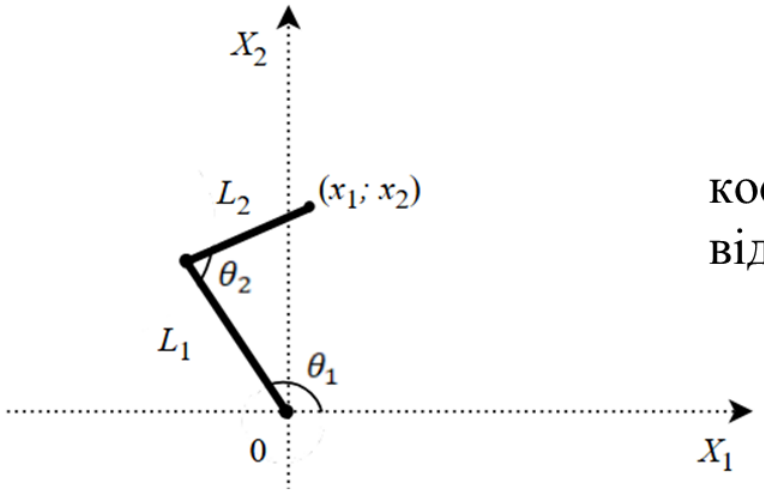
- Кількість компонентів у Gaussian Mixture: 2.
- Архітектура нейронної мережі: один прихований шар з 20 нейронами, активаційна функція – tanh.
- Оптимізатор: Adam (full-batch).
- Кількість епох навчання: 1500.

Результати прогнозування



	Метод MDN	Запропо-нований метод	Покра-щення (%)
Training loss	1.208	1.152	4.559
Test loss	1.173	1.120	4.561
Acc. Training loss (500+)	1337.074	1304.965	2.401
Acc. Test loss (500+)	1318.767	1291.521	2.066

Задача зворотної кінематики для двохсегментного маніпулятора

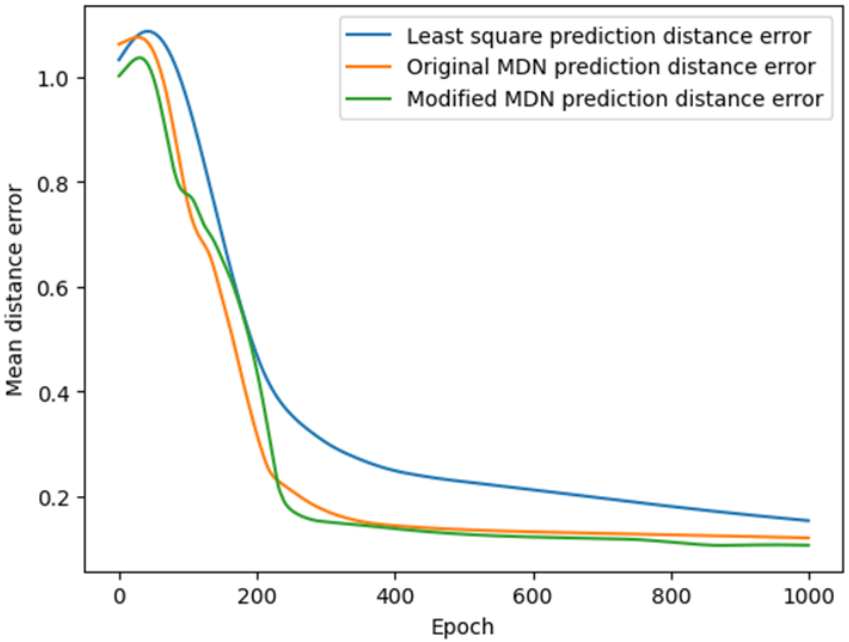


Математична модель:

$$x_1 = L_1 \cos(\theta_1) - L_2 \cos(\theta_1 + \theta_2),$$
$$x_2 = L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2)$$

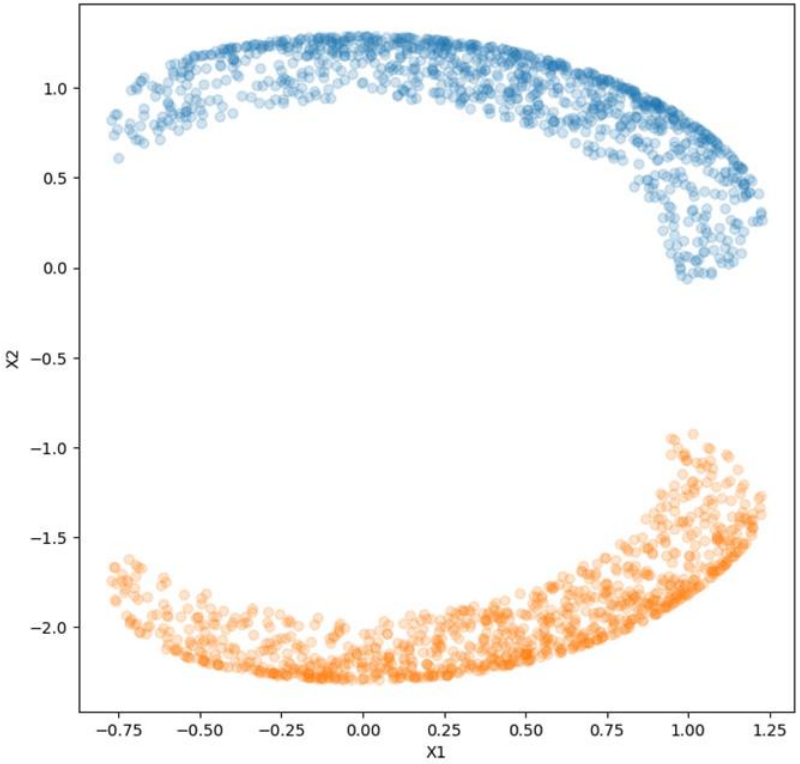
Задача: для заданої кінцевої координати $(x_1; x_2)$ спрогнозувати відповідні кути повороту.

Результати прогнозування



	Метод найменших квадратів	Метод MDN	Запропонований метод	Покращення (%)
Mean Distance Error (MDE)	0.154	0.121	0.107	30. 584(LS) 11.573 (MDN)
Accumulative MDE (epoch 200+)	182.645	115.610	107.909	40.919 (LS) 6.661 (MDN)

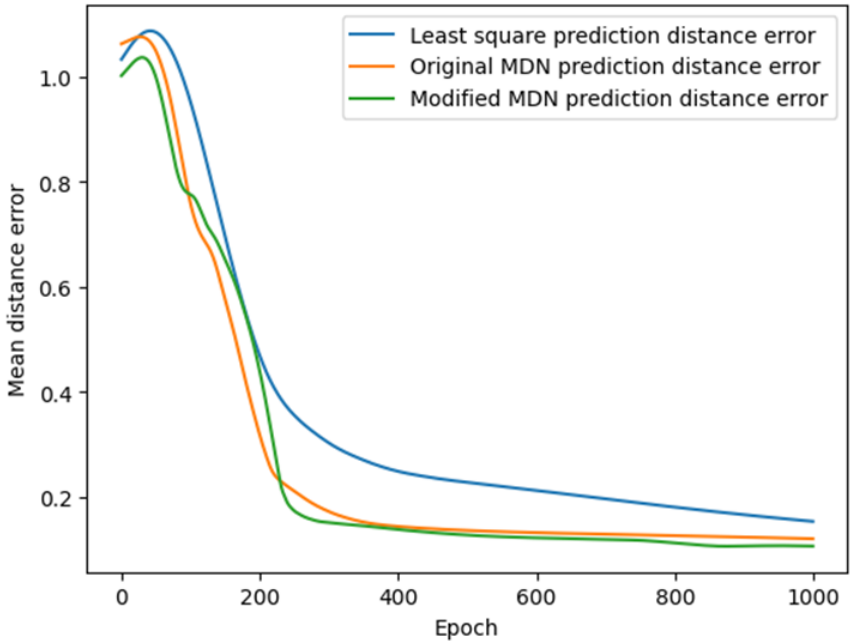
Задача з двома маніпуляторами



Модифікація попередньої задачі:

- Перша рука починається в точці (0,0).
- Друга має зсув вздовж осі X_2.
- Кути другої руки мають протилежний напрям обертання.

Результати прогнозування



	Метод найменших квадратів	Метод MDN	Запропонований метод	Покращення (%)
Mean Distance Error (MDE)	1.113	1.046	1.020	8.392 (LS) 2.467 (MDN)
Accumulative MDE (epoch 200+)	944.796	847.868	835.355	11.584 (LS) 1.476 (MDN)

Результати оцінювання запропонованого методу

Результати для задачі прогнозування синтетичних даних

Метрика	Запропонований метод	Покращення (%)
Остаточна втрата (loss) на тренувальному наборі	1.152	+4.559
Остаточна втрата на тестовому наборі	1.120	+4.561
Кумулятивна втрата (епохи 500–1500) – тренувальний набір	1304.965	+2.401
Кумулятивна втрата – тестовий набір	1291.521	+2.066

Результати для задачі зворотної кінематики двосегментного маніпулятора

Метрика	Запропонований метод	Покращення (%)
Середня помилка відстані (на тестових даних)	0.107	+30.584 (проти LS), +11.573 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	107.909	+40.919 (LS), +6.661 (MDN)

Результати для задачі з двома роботизованими маніпуляторами

Метрика	Запропонований метод	Покращення (%)
Середня помилка відстані (на тесті)	1.020	+8.392 (проти LS), +2.467 (проти MDN)
Кумулятивна помилка (епохи 200–1000)	835.355	+11.584 (LS), +1.476 (MDN)

Технології розроблення програмного забезпечення

PyTorch

python™

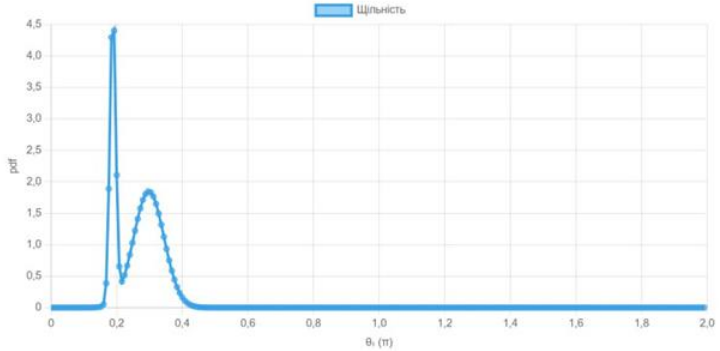
kaggle

node

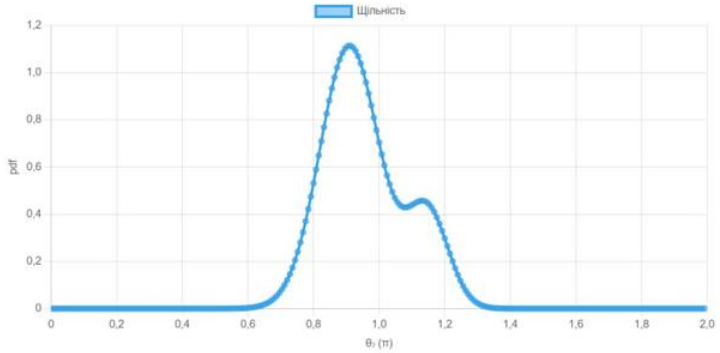
ONNX

Приклад роботи програмного забезпечення

Розподіл ймовірності для кута θ1



Розподіл ймовірності для кута θ2



Оптимальні кути та результуючі координати

Оптимальний θ_1: 0.5911 рад

Оптимальний θ_2: 3.6011 рад

x_1: 0.6887

x_2: 0.7159

Похибка: 0.0195

Візуалізація положення руки

Наукова новизна

Запропоновано метод прогнозування даних з мультимодальним розподілом на основі моделі машинного навчання архітектури MDN та імовірнісної функції втрат для навчання моделі, що дозволяє збільшити точність прогнозування на 4,56% для синтетичних наборів даних з чітко визначеними мультимодальними характеристиками, а також на 11,57% та 2,47% для двох наборів даних, що відображають прикладні задачі.

Практична значимість

Розроблений алгоритмічно-програмний метод та програмне забезпечення може використовуватись користувачами як програмний застосунок з графічним інтерфейсом для прогнозування даних з мультимодальним розподілом в умовах невизначеності.

Шляхи покращення запропонованого методу

1. Удосконалення запропонованого методу для прогнозування даних більш високої розмірності.
2. Розроблення динамічного підходу для підбору гіперпараметра.
3. Розширення типів задач, зокрема розгляд послідовностей для прогнозування динаміки систем із неоднозначними станами.
4. Інтеграція з іншими типами нейронних архітектур, такими як рекурентні чи трансформери, для моделювання мультимодальних залежностей у контексті часових або просторових даних.

Висновки

1. Проведено аналіз існуючих методів та програмних рішень прогнозування даних з мультимодальним розподілом. Виявлені їх переваги та недоліки.
2. Розроблено алгоритмічно-програмний метод для прогнозування даних з мультимодальним розподілом на основі моделі MDN, що дозволив збільшити точність прогнозування на 4,56% для задач із синтетичним набором та на 11,57% та 2,47% для прикладних задач.
3. Проведено аналіз отриманих результатів прогнозування запропонованим методом.
4. Розроблено та проведено тестування програмного забезпечення, що реалізує запропонований метод.
5. Запропоновано шляхи вдосконалення розробленого методу.

Апробація роботи

Основні положення та результати виконаної роботи були:

1. Викладені в статті «Algorithmic and Software Method for Predicting Data with Multimodal Distribution based on the MDN Model», що подана до журналу категорії «Б» «Herald of Khmelnytskyi National University. Technical sciences».
2. Викладені та обговорені на XVII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2024 та опубліковані у збірнику тез доповідей.

Дякую за увагу!