ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

КАТАШИНСЬКИЙ ДМИТРО ОЛЕКСАНДРОВИЧ

УДК 004.272+004.032.26

ДИСЕРТАЦІЯ
МЕТОДИ ТА ЗАСОБИ АСОЦІАТИВНОГО ОБРОБЛЕННЯ ДАНИХ ДЛЯ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

123 – Комп'ютерна інженерія

Технічні науки

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Д.О. Каташинський

Науковий керівник:
доктор технічних наук,
професор Мартинюк Тетяна Борисівна

Вінниця – 2025

АНОТАЦІЯ

Каташинський Д.О. Методи та засоби асоціативного оброблення даних для класифікації об’єктів в інтелектуальних системах. – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 «Інформаційні технології» за спеціальністю 123 «Комп’ютерна інженерія». – Вінницький національний технічний університет, МОН України, Вінниця, 2025.

Сучасні комп'ютерні засоби вимагають високопродуктивних методів оброблення значних обсягів даних. Зважаючи на це, особливу увагу варто приділити специфічним аспектам асоціативного оброблення, що передбачає виконання логічних та пошукових операцій, а також альтернативним методам оброблення значних масивів даних.

Це пов’язано, не в останню чергу, з тим, що до складу асоціативних операцій входять вибірка за зовнішнім ключем, пошук даних за аналогією, сортування і ранжування елементів масиву даних. Крім того, в наявності потреба у компактній апаратній реалізації засобів розпізнавання та аналізу об’єктів у системі автономного керування мобільних роботів.

Дисертаційну роботу присвячено розв’язанню науково-прикладної задачі розроблення методів та засобів асоціативного оброблення даних для класифікації та ранжування об’єктів у інтелектуальних системах. Актуальність дослідження визначається необхідністю обробляти в реальному часі значні обсяги інформації для задач штучного інтелекту і відповідно потребою у створенні програмних та апаратно-ефективних засобів для виконання інтелектуальних функцій.

Мета дослідження полягає у розширенні функціональних можливостей нейроподібних класифікаторів за рахунок ранжування результатів класифікації, що забезпечить наочність та зручність користувачам при

прийнятті рішень в інтелектуальних системах різного призначення.

Для досягнення поставленої мети вирішено такі завдання:

- досліджено методи та засоби паралельного оброблення масивів даних на регулярних структурах для асоціативних задач
- визначено функціональні можливості позрізового оброблення одно – та двовимірного масиву даних на базі SM-перетворення.
- досліджено методи реалізації механізму конкуренції типу WTA для нейроподібних класифікаторів об’єктів.
- розроблено апаратні засоби паралельного оброблення масиву даних для асоціативно-логічних операцій.
- розроблено програмні засоби для моделювання процесів класифікації об’єктів з асоціативним обробленням масивів даних.
- виконано імітаційне моделювання процесів класифікації об’єктів для оцінювання функціональних можливостей запропонованих методів.

Об’єктом дослідження є процеси паралельного оброблення масивів даних на регулярних структурах для асоціативних задач з орієнтацією на класифікацію об’єктів.

Предметом дослідження є методи, алгоритми та програмні засоби, що забезпечують паралельне оброблення масивів даних на регулярних структурах для класифікаторів об’єктів.

У Вступі наведено актуальність дослідження, мету і задачі завдання, об’єкт і предмет дослідження, а також методи дослідження. Розглянуто наукову новизну одержаних результатів, а також їх практичну значимість, показано особистий внесок здобувача, апробацію результатів дисертації, список публікацій.

Перший розділ присвячено огляду методів та засобів паралельного оброблення масивів даних на регулярних структурах. Зокрема, розглянуто

асоціативні аспекти оброблення даних в інтелектуальних системах, а саме, інтелектуальна пам’ять, методи асоціативного оброблення даних, типи нейромережної асоціативної пам’яті. Крім того, розглянуто особливості позрізового оброблення одновимірного масиву даних на базі SM-перетворення, а також нейромережний підхід до медичної експрес-діагностики на прикладі особливостей нейромережного класифікатора.

У другому розділі розглянуто паралельне оброблення масиву даних для асоціативно-логічних операцій. Показано особливості сортування як базис асоціативного оброблення одновимірного числового масиву даних, а також особливості обчислювального процесу при класифікації об’єктів.

У третьому розділі розглянуто структурні та функціональні особливості нейроподібних класифікаторів з розширеними функціональними можливостями. Запропоновано структури двох нейроподібних класифікаторів: перший – з позрізовим обробленням нормалізованих дискримінантних функцій на обчислювальній мапі з регулярною структурою, другий – з розширеними функціональними можливостями на базі механізму реалізації конкуренції типу "1 з N" із задіянням операцій інкремента/декремента.

Четвертий розділ присвячено програмним засобам для моделювання процесів класифікації, наведено вихідні дані для імітаційного моделювання процесу класифікації на прикладі медичного діагностування захворювань. Наведено опис комп’ютерної програми та проведено аналіз результатів моделювання.

В ході розв’язання поставлених завдань отримано такі наукові результати:

1. Подальший розвиток отримав метод паралельного оброблення масивів даних для асоціативних задач у процесі сортування та ранжування при класифікації об'єктів через використання ознак обнулення елементів числового

масиву в процесі альтернативного сортування, що розширює функціональні можливості класифікатора за рахунок ранжування результатів.

2. Вперше запропоновано вдосконалення методу позрізового оброблення масиву даних на базі SM-перетворення з нормалізацією елементів дискримінантних функцій, що забезпечує реалізацію процесу класифікації об’єктів у матричному (систолічному) форматі.

3. Вперше запропоновано варіант реалізації механізму конкуренції для нейроподібного класифікатора із застосуванням результатів сортування елементів числового масиву, що забезпечує паралелізм процесу класифікації через задіяння швидкісної операції декремента одночасно до всіх елементів числового масиву.

Практична значимість результатів дослідження полягає в наступному:

1. Розроблено структуру нейроподібного класифікатора об’єктів з позрізовим обробленням нормалізованих елементів дискримінантних функцій на обчислювальній мапі з регулярною (систолічною) структурою з орієнтацією на ПЛІС.

2. Розроблено структуру базових вузлів нейроподібного класифікатора із паралельним застосуванням операцій декремента/інкремента для реалізації сортування/ранжування на реверсивних лічильниках.

3. Промодельовано процес класифікації з ранжуванням результатів на прикладі медичного діагностування захворювань, що підтвердив наочність отриманих результатів при прийнятті рішень в експертних системах.

Практична значимість результатів полягає у можливості використання розроблених методів та апаратних засобів у підсистемах підтримки прийняття рішень в інтелектуальних системах, зокрема, для медичного діагностування, а також у блоці сприйняття та аналізу зображень об’єктів у системах автономного керування мобільних роботів.

Окремі розробки дисертаційної роботи впроваджено на базі ТОВ «Елефантслаб», а також у навчальний процес кафедри обчислювальної техніки ВНТУ у дисципліни «Інтелектуальні системи, технології та нейрокомп’ютери» та «Методи штучного інтелекту та нейромережі».

Основні результати виконаних в дисертаційній роботі досліджень опубліковано в 11 роботах: 6 статтях у фахових виданнях; 5 публікаціях у збірках праць конференцій.

Ключові слова: асоціативне оброблення, паралельне обчислення, нейроподібний класифікатор, сортування, ранжування, SM-перетворення, імітаційне моделювання, медичне діагностування, система автономного керування мобільного робота.

ANNOTATION

Katashynskyi D.O. Methods and Means of Associative Data Processing for Object Classification in Intelligent Systems. – Qualification scholarly work as a manuscript. Dissertation for obtaining the Doctor of Philosophy degree in the field of knowledge 12 “Information Technologies,” specialty 123 “Computer Engineering.” – Vinnytsia National Technical University, Ministry of Education and Science of Ukraine, Vinnytsia, 2025.

Modern computer technologies require high-performance methods for processing large volumes of data. In this context, particular attention should be paid to the specific aspects of associative processing, which involves performing logical and search operations, as well as alternative methods for processing substantial data arrays.

This is due, among other factors, to the fact that associative operations include external-key selection, analogy-based data search, sorting, and ranking of data array elements. In addition, there is a growing need for compact hardware implementations of object recognition and analysis tools within autonomous mobile robot control systems.

The dissertation is devoted to solving the scientific and applied problem of developing methods and means of associative data processing for classification and ranking of objects in intelligent systems. The relevance of the research is determined by the necessity to process large volumes of information in real time for artificial intelligence tasks and, accordingly, by the need to create software and hardware-efficient tools for performing intelligent functions.

The purpose of the research is to expand the functional capabilities of neuro-like classifiers by introducing ranking of classification results, which provides clarity and convenience for users when making decisions in intelligent systems of various applications.

To achieve this purpose, the following tasks were solved:

- methods and tools for parallel processing of data arrays on regular structures for associative tasks were investigated;
- functional capabilities of slice-by-slice processing of one-dimensional and two-dimensional data arrays based on the SM-transform were determined;
- methods for implementing a Winner-Takes-All (WTA) competition mechanism for neuro-like object classifiers were studied;
- hardware tools for parallel processing of data arrays for associative-logical operations were developed;
- software tools for modeling object classification processes with associative data processing were created;
- simulation modeling of classification processes was performed to evaluate the functional capabilities of the proposed methods.

The object of study is the processes of parallel data array processing on regular structures for associative tasks focused on object classification.

The subject of study is the methods, algorithms, and software tools that ensure parallel data array processing on regular structures for object classifiers.

The Introduction presents the relevance of the research, purpose and objectives, object and subject of study, as well as the research methods. The scientific novelty of the obtained results and their practical significance are considered, along with the personal contribution of the author, approbation of the dissertation results, and the list of publications.

The first chapter is devoted to a review of methods and tools for parallel data array processing on regular structures. In particular, associative aspects of data processing in intelligent systems are examined, such as intelligent memory, methods of associative data processing, and types of neural associative memory. Additionally, features of slice-by-slice processing of one-dimensional data arrays based on the SM-

transform are considered, as well as a neural-network approach to rapid medical diagnostics using the example of a neuro-like classifier.

The second chapter discusses parallel data array processing for associative-logical operations. Features of sorting as a basis for associative processing of one-dimensional numerical arrays are presented, as well as features of the computational process during object classification.

The third chapter examines structural and functional features of neuro-like classifiers with extended functional capabilities. Structures of two neuro-like classifiers are proposed: the first one with slice-by-slice processing of normalized discriminant functions on a computational map with a regular structure; the second with extended functional capabilities based on a "1-of-N" competition mechanism using increment/decrement operations.

The fourth chapter is devoted to software tools for modeling classification processes. Input data for simulation of the classification process are presented using the example of medical disease diagnostics. A description of the computer program is provided along with analysis of the simulation results.

The following scientific results were obtained:

1. Further development of the method of parallel data array processing for associative tasks in the process of sorting and ranking during object classification was achieved through the use of element-zeroing features in a numerical array during alternative sorting, which expands the classifier's functional capabilities by introducing ranking of results.

2. For the first time, an improved method of slice-by-slice data array processing based on the SM-transform with normalization of discriminant function elements was proposed, which enables classification processes to be implemented in a matrix (systolic) format.

3. For the first time, an implementation of a competition mechanism for a neuro-like classifier using sorting results of numerical array elements was proposed, which ensures parallelism of the classification process by applying a high-speed decrement operation simultaneously to all elements of the numerical array.

The practical significance of the obtained results is as follows:

1. A structure of a neuro-like object classifier with slice-by-slice processing of normalized discriminant function elements on a computational map with a regular (systolic) structure, oriented toward FPGA implementation, was developed.

2. A structure of basic nodes of a neuro-like classifier with parallel application of decrement/increment operations for implementing sorting/ranking on reversible counters was developed.

3. A classification process with ranking of results was simulated using the example of medical disease diagnostics, confirming the clarity of the results for decision-making in expert systems.

The practical significance also lies in the possibility of using the developed methods and hardware tools in decision-support subsystems of intelligent systems, in particular for medical diagnostics, as well as in perception and image analysis modules in autonomous mobile robot control systems.

Certain developments of the dissertation were implemented at “ElephantsLab” LLC and in the educational process of the Department of Computer Engineering of VNTU in the courses “Intelligent Systems, Technologies and Neurocomputers” and “Artificial Intelligence Methods and Neural Networks”.

The main results of the dissertation research are published in 11 works: 6 articles in professional journals and 5 publications in conference proceedings.

Keywords: associative processing, parallel computing, neuro-like classifier, sorting, ranking, SM-transform, simulation modeling, medical diagnostics, autonomous mobile robot control system.

Список опублікованих праць за темою дисертації

Статті у журналах, що включенні до переліку наукових фахових видань України

1 Т.Б. Мартинюк, Д.О. Каташинський, М.В. Микитюк, та М.О. Зайцев, “Особливості обчислювальних процесів на базі SM-перетворення”, *Оптико-електронні інформаційно-енергетичні технології*, №2(44), с.32-37. 2022. https://doi.org/10.31649/1681-7893-2022-44-2-32-37

2 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, “Структурні особливості нейроподібного класифікатора об’єктів”, *Наукові праці ВНТУ*, №4, с.1-7. 2023. https://doi.org/10.31649/2307-5376-2023-4-1-7

3 Т.Б. Мартинюк, А.В. Кожем’яко, І.В. Булига, та Д.О. Каташинський, “Графічна модель паралельної асоціативної обробки числових даних”, *Наукові праці ВНТУ*, №4, с.58-62. 2024. https://doi.org/10.31649/2307-5376-2024-4-58-62

4 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, “Особливості процесу класифікації у контексті медичного моделювання”, *Наукові праці ВНТУ*, №1, с.80-85. 2025. https://doi.org/10.31649/2307-5376-2025-1-80-85

5 Т.Б. Мартинюк, Д.О. Каташинський, “Особливості асоціативного оброблення даних в інтелектуальних системах”, *Оптико-електронні інформаційно-енергетичні технології*, №1 , с.44-52. 2025. https://doi.org/10.31649/1681-7893-2025-49-1-44-52

6 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, «Особливості обчислювального процесу у матричному класифікаторі об’єктів», *Наукові праці ВНТУ*, №3, 2025. https://praci.vntu.edu.ua/index.php/praci/article/view/850

Публікації в матеріалах конференцій, тезах доповідей

7 Д. О. Каташинський, «Комп’ютерна система для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу попередніх продаж», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2021.
https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13180

8 О. Каташинський, та Т.Б. Мартинюк, “Функціональні можливості оброблення даних на база SM-перетворення”, *Матеріали Всеукраїнської*

науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи», Вінниця: ВНТУ, 2023. https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/17047

9 Д.О. Каташинський, Т.Б. Мартинюк, та І.В. Булига, «Модель двовимірного нейроподібного класифікатора», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2024. https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21168

10 Т.Б. Мартинюк, М. А. Очкуров, Д.О. Каташинський, та І.В. Булига, «Структурні та функціональні особливості класифікаторів об’єктів», на *Міжнародній науково-практичній конференції з оптико-електронних технологій* “ФОТОНІКА − ODS 2025”. Вінниця, 2025. https://conferences.vntu.edu.ua/index.php/ods/ods_2025/paper/view/24668

11 І. В. Булига, Т.Б. Мартинюк, та Д.О. Каташинський, «Особливості асоціативної обробки числових даних за різницевими зрізами», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2025. https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48386/24839.pdf

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ВНТУ – Вінницький національний технічний університет

ДА – дискримінантний аналіз

ЛДФ – лінійна дискримінантна функція

ОТ – обчислювальна техніка

ПЕ – процесорний елемент

ПЛІС – програмована логічна інтегральна схема

РЗ – різницевий зріз

РІА – регулярний ітераційний алгоритм

САПР – система автоматизованого проектування

СТ – лічильник (counter)

ВСТУП

Актуальність роботи. Сучасні комп'ютерні засоби вимагають високопродуктивних методів оброблення значних обсягів даних. Зважаючи на це, особливу увагу варто приділити специфічним аспектам асоціативного оброблення, що передбачає виконання логічних та пошукових операцій, а також альтернативним методам оброблення значних масивів даних.

Перспективність такого підходу пояснюється їх широким застосуванням у таких прикладних областях, як системи управління базами даних (СУБД), пошук і сортування IP-адрес в комп'ютерних мережах, а також ранжування даних, наприклад, у підсистемах прийняття рішень у складі інтелектуальних систем, зокрема, для медичного діагностування. Це пов'язано, не в останню чергу, з тим, що до складу асоціативних операцій входять вибірка за зовнішнім ключем, пошук даних за аналогією, сортування і ранжування елементів масиву даних. Крім того, в наявності потреба у компактній апаратній реалізації засобів розпізнавання та аналізу об'єктів у системі автономного керування мобільних роботів.

Прикладом застосування асоціативного оброблення даних є різновиди нейромереж, які виконують функції авто- та гетероасоціативної пам'яті. Ще одним затребуваним підходом є використання класифікатора з розширеними функціональними можливостями у складі підсистем підтримки прийняття рішень для експертних систем різного призначення. Ці приклади свідчать про конкретний зв'язок методів асоціативного оброблення даних і впровадження нейротехнологій у створення інтелектуальних систем різного призначення.

Значний науковий вклад в розвиток методів та засобів штучного інтелекту і, зокрема, важливого напрямку з асоціативного оброблення даних внесли такі вітчизняні та іноземні вчені, як І.Г. Цмоць, С.Г. Антощук, В.М. Крилов, Г.Е.

Цейтлін, Г.М. Гнатієнко, В.Є. Снитюк, T. Kohonen, D.E. Knuth, C. Foster, K.I. Thurber.

Серед вчених, які сприяли значному розвитку нейромережних технологій у теперішній час, необхідно, в першу чергу, відмітити М.М. Амосова, І.І.Івахненка, О.М. Різника, Е.М. Куссуль, Є.В. Бодянського, О.Г. Руденка, Л.І. Тимченка, S.S. Haykin, S.Osovski, R. Callan та інші.

Отже, практичною задачею є розробка та аналіз методів і засобів паралельного оброблення масивів даних на регулярних структурах з акцентом на асоціативно-логічні операції та позрізове оброблення, а також створення програмних засобів для їх моделювання та імітаційного аналізу.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота здійснювалася здобувачем протягом 2022-2025рр. відповідно до наукового напрямку кафедри обчислювальної техніки Вінницького національного технічного університету, зокрема, у ході виконання науково дослідної роботи «Високопродуктивні багатоканальні аналого-цифрові самокалібровані системи моніторингу й синхронного опрацювання низькочастотних сигналів» (№ держ. реєстрації 0120U002205).

Мета і завдання роботи. Мета дослідження полягає у розширенні функціональних можливостей нейроподібних класифікаторів за рахунок ранжування результатів класифікації, що забезпечить наочність та зручність користувачам при прийнятті рішень в інтелектуальних системах різного призначення.

Завдання дослідження. Для досягнення поставленої мети необхідно вирішити такі завдання:

- Дослідити методи та засоби паралельного оброблення масивів даних на регулярних структурах для асоціативних задач

- Визначити функціональні можливості позрізового оброблення одно – та двовимірного масиву даних на базі SM-перетворення.
- Дослідити методи реалізації механізму конкуренції типу WTA для нейроподібних класифікаторів об’єктів.
- Розробити апаратні засоби паралельного оброблення масиву даних для асоціативно-логічних операцій.
- Розробити програмні засоби для моделювання процесів класифікації об’єктів з асоціативним обробленням масивів даних.
- Виконати імітаційне моделювання процесів класифікації об’єктів для оцінювання функціональних можливостей запропонованих методів.

Об’єктом дослідження є процеси паралельного оброблення масивів даних на регулярних структурах для асоціативних задач з орієнтацією на класифікацію об’єктів.

Предметом дослідження є методи, алгоритми та програмні засоби, що забезпечують паралельне оброблення масивів даних на регулярних структурах для класифікаторів об’єктів.

Методи дослідження:

- методи та алгоритми позрізового оброблення одно – та двовимірних масивів даних на базі SM-перетворення, що забезпечують паралельну організацію обчислень;
- підходи до паралельного виконання асоціативно-логічних операцій, що дозволяють ефективно аналізувати великі масиви даних та виявляти закономірності;
- програмні засоби моделювання, які реалізують запропоновані методи та дозволяють досліджувати їх практичність у різних умовах;

- імітаційне моделювання, що дозволяє оцінити розроблені підходи на тестових наборах даних, визначити їх переваги та обмеження.

Наукова новизна одержаних результатів

В ході розв’язання поставлених завдань отримано наукові результати:

1. Подальший розвиток отримав метод паралельного оброблення масивів даних для асоціативних задач у процесі сортування та ранжування при класифікації об'єктів через використання ознак обнулення елементів числового масиву в процесі альтернативного сортування, що розширює функціональні можливості класифікатора за рахунок ранжування результатів.

2. Вперше запропоновано вдосконалення методу позрізового оброблення масиву даних на базі SM-перетворення з нормалізацією елементів дискримінантних функцій, що забезпечує реалізацію процесу класифікації об’єктів у матричному (систолічному) форматі.

3. Вперше запропоновано варіант реалізації механізму конкуренції для нейроподібного класифікатора із застосуванням результатів сортування елементів числового масиву, що забезпечує паралелізм процесу класифікації через задіяння швидкісної операції декремента одночасно до всіх елементів числового масиву.

Практична значимість результатів дослідження:

1. Розроблено структуру нейроподібного класифікатора об’єктів з позрізовим обробленням нормалізованих елементів дискримінантних функцій на обчислювальній мапі з регулярною (систолічною) структурою з орієнтацією на ПЛІС.

2. Розроблено структуру базових вузлів нейроподібного класифікатора із паралельним застосуванням операцій декремента/інкремента для реалізації сортування/ранжування на реверсивних лічильниках.

3. Промодельовано процес класифікації з ранжуванням результатів на прикладі медичного діагностування захворювань, що підтвердив наочність отриманих результатів при прийнятті рішень в експертних системах.

Окремі розробки дисертаційної роботи впроваджено на базі ТОВ «Елефантслаб», а також у навчальний процес кафедри обчислювальної техніки ВНТУ у дисципліни «Інтелектуальні системи, технології та нейрокомп'ютери» та «Методи штучного інтелекту та нейромережі» для студентів, які навчаються за освітньою програмою «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія».

Особистий внесок здобувача. Всі основні результати дисертаційної роботи отримано автором особисто. В публікаціях, які написані у співавторстві, здобувачеві належить: аналіз векторно-матричних операцій у базисі SM-перетворення [1, 8]; розширений структурний базис моделі нейроподібного класифікатора об'єктів [2, 9]; інформаційний граф процесу сортування чисел за методом різницевих зрізів [3, 11]; розширена структура класичної моделі класифікатора на базі дискримінантного аналізу [4]; аналіз процесу конкуренції типу "1 з N" у базовому блоці класифікатора об'єктів [5, 10]; урахування коефіцієнтів зміщення у лінійних дискримінантних функціях [6]; програма для підтримки інтернет торгівлі з прогнозуванням попиту [7].

Апробація результатів дисертації. Основні наукові і практичні результати роботи доповідались і обговорювались на щорічних науково-технічних конференціях ВНТУ "Молодь в науці: дослідження, проблеми, перспективи" (м. Вінниця, 2021, 2023-2025рр.) та X міжнародній науково-практичній конференції з оптико-електронних інформаційних технологій "Фотоніка - ODS 2025" (м. Вінниця, 2025р.).

Публікації. Основні результати виконаних в дисертаційній роботі досліджень опубліковано в 11 роботах: 6 статтях у фахових виданнях; 5 публікаціях у збірках праць конференцій.

Структура та обсяг роботи. Дисертаційна робота складається складається зі вступу, чотирьох розділів, висновків до розділів, загальних висновків, використаних джерел з 78 найменувань, 4 додатків на 26 сторінках. Загальний обсяг дисертації становить 169 сторінок. Основний зміст викладено на 128 сторінках друкованого тексту, містить 1 таблицю та 1 рисунок на 2 сторінках.

РОЗДІЛ 1
МЕТОДИ ТА ЗАСОБИ ПАРАЛЕЛЬНОГО ОБРОБЛЕННЯ МАСИВІВ ДАНИХ НА РЕГУЛЯРНИХ СТРУКТУРАХ

1.1 Асоціативні аспекти оброблення даних в інтелектуальних системах

Асоціативне оброблення даних, яке містить такі необчислювальні операції, як вибірка за ключем, пошук за аналогією, сортування, а також ранжування елементів масиву даних [34,35] є однією з важливих процедур штучного інтелекту [3-6]. Отже, необхідність подальшого розвитку асоціативних принципів оброблення даних пов'язано, не в останню чергу, з використанням автоматизованих систем збереження та паралельного оброблення значних масивів даних за їх символьними іменами або за їх змістом, тобто за так званими «асоціаціями» [7,8]. Тому областями широкого застосування асоціативного оброблення даних є, зокрема, системи управління базами даних (СУБД), пошук і сортування IP-адреси у комп'ютерних мережах і підсистеми підтримки прийняття рішень в інтелектуальних системах [9-11].

Для обчислювальної техніки характерним є асоціативне оброблення у більшості випадків над елементами числового масиву [2]. Разом з тим, такі асоціативно-логічні операції як сортування і ранжування [1,2,12] активно використовуються в нейротехнологіях, зокрема, при побудові окремих типів нейромереж [4,5,13]. Наприклад, у нейромережних класифікаторах відому процедуру "1 з N" для реалізації конкуренції у шарах нейромереж можна виконати не тільки із застосуванням структур нейромереж типу MAXNET [6], але й в процесі сортування векторного (лінійного) масиву чисел з визначенням максимального елемента [9,14].

Крім того, разом з широко використовуваним програмним забезпеченням знаходять застосування апаратні методи реалізації асоціативного оброблення даних [15], оскільки асоціативні операції задіяно в інтелектуальних системах різного призначення [16,17].

1.1.1 Асоціативність та асоціативне оброблення даних

Важливою особливістю пам’яті людини є її асоціативний характер, що містить такі властивості [7]:

- пошук інформації базується на мірі схожості із ключовим образом;
- образи зберігаються як структуровані послідовності;
- вибірка інформації представляє собою динамічний процес.

Разом з тим, одне із узагальнених понять “асоціації” містить таке пояснення: асоціація - це явище, коли одне представлення викликає інше за схожістю, суміжністю або протилежністю [7]. В обчислювальній техніці, до речі, вважається що поняття “асоціативний” відображає лише факт наявності взаємозв’язків між даними і не має відношення до механізму збереження інформації у пам’яті [7]. Таким чином позначається розмежування понять "асоціативне оброблення" і "асоціативна пам'ять".

Останнє поняття в обчислювальній техніці відноситься до так званої "пам'яті з адресацією за змістом". Отже, асоціативне оброблення числової інформації, в першу чергу, передбачає пошук за асоціацією, тобто за зовнішнім ключем [1,7]. Але, разом з тим, може виконуватись оброблення даних без зовнішньої ключової інформації, зокрема, при пошуку максимального або мінімального числа, тобто як результат сортування даних. Крім того, існує такий варіант пошуку, коли серед деякої сукупності даних необхідно знайти

такі, що найкраще відповідають ключовому зразку. У цьому випадку асоціативна вибірка відповідає певним чином процесу розпізнавання образів [3-6].

В результаті, введення у перелік виконуваних асоціативно-логічних операцій інших, які не пов'язані з вибіркою за змістом, розширює функціональні можливості асоціативної пам’яті, що робить доречним позначення її як інтелектуальної пам’яті. До додаткових операцій такої пам'яті можна віднести такі розповсюджені масово-паралельні операції над елементами векторів, як визначення мінімального або максимального елемента та середнього k чисел [18].

1.1.2 Нейромережна асоціативна пам’ять

Як приклад відомих класичних структур нейромереж, які виконують функції асоціативної пам'яті, необхідно відмітити, в першу чергу, мережу Хопфілда, мережу Хеммінга і мережу ВАМ – двоспрямовану асоціативну пам'ять [4-6]. Базовими функціями такої пам'яті є запам'ятовування, відтворення та забування образів, на яких вона навчалась [19]. При цьому треба розрізняти автоасоціативну пам'ять у вигляді мережі Хопфілда і мережі ВАМ, які можуть за спотвореним (зашумленим) образом відновити найближчий до нього еталонний образ [5,6]. На відміну від них прикладом гетероасоціативної пам'яті є мережа Хеммінга з можливістю вибрати еталонний образ за його міткою [4,5], тобто мережа з функцією класифікації образів.(рис 1.1).

Для наочності випадку доцільно показати взаємозв'язок між асоціативною вибіркою у розподіленій асоціативній пам'яті і процесом розпізнавання образів у нейромережах. Об'єднуючим їх фактором є здатність як асоціативної пам'яті, так і нейромереж до селективної видачі інформації у відповідь на вхідні образи, які в даному випадку розглядаються як аргументи пошуку [7]. Цей

взаємозв'язок підтверджують базові математичні залежності процесів, що відбуваються у двох наведених способах асоціативного оброблення даних.

Рисунок 1.1 – Приклади асоціативності у нейромережах

В асоціативній пам’яті характерним є векторно-матричне множення у вигляді [7]:

$$Y = \boldsymbol{M} \cdot X = \sum \gamma_k \cdot Y_k , \tag{1.1}$$

де X - вхідний образ (вектор) як функція пошукового аргументу; $\boldsymbol{M}$ - матриця, що визначає структуру асоціативної пам’яті; Y - вихідний образ (вектор) як

деякий “спогад” у пам’яті $\boldsymbol{M}$; γ_k - коефіцієнт лінійної регресії X на множині $\{ X_k \}$.

Для нейромереж як моделей асоціативної пам’яті також можна використовувати векторно-матричне множення, щоб показати асоціацію між ключовими векторами X_k і запам’ятовуваним вектором Y_k y вигляді [6]:

$$Y_k = \boldsymbol{W}_k \cdot X_k, \quad k = 1, \ldots, K, \tag{1.2}$$

де $\boldsymbol{W}_k$ - вагова матриця, яка визначається навчальними парами (X_k, Y_k).

В цьому випадку суму матриць вагових коефіцієнтів всього набору асоціацій можна розглядати як матрицю “пам’яті”, а саме, як “досвід”, отриманий при навчанні нейромереж [6], тобто встановити таке співвідношення:

$$\boldsymbol{M} = \sum_{k=1}^{K} \boldsymbol{W}_k . \tag{1.3}$$

Разом з тим, особливий інтерес для нейромереж представляє використання такої асоціативно-логічної операції, як сортування елементів векторного лінійного масиву чисел при реалізації процедури “1 з N” [20]. У більшості випадків вона реалізується у нейромережних класифікаторах як механізм конкуренції нейронів через їх від’ємні зв’язки у конкурентному шарі нейромереж [4-6].

1.1.3 Асоціативні рівні оброблення даних в інтелектуальних системах

Одним з наочних прикладів ефективного застосування асоціативного оброблення даних є інтелектуальні системи керування у робототехніці [21, 22]. Для таких систем керування доцільним є забезпечення таких базових умов [23, 24]:

- апаратна підтримка процедур керування;

- відстеження та класифікація (розпізнавання) поточного стану у реальному часі;
- прийняття рішення у реальному часі.

За таких умов найбільш прийнятним є використання нейромережних методів та засобів [21, 23]. Саме для оброблення сенсорної інформації використовуються такі процедури, як пошук асоціацій між образами, ситуаціями та їх позначеннями, що у певній мірі відповідає інтелектуалізації керування мобільними роботами [21,23,24]. При цьому необхідно відмітити, що для створення інтелектуальних систем керування роботами активно впроваджуються нейронні елементи та їх базові складові на новітніх технологіях [25-27].

Ще одним наочним прикладом використання як нейротехнологій, так і асоціативних операцій можна розглядати принципи функціонування класифікаторів об'єктів [13, 17], які, як правило, задіяні у підсистемах підтримки прийняття рішень [10-12, 28]. Особливо це актуально для процесу медичного діагностування [29,30].

На рисунку 1.2 показано приклад застосування методів та засобів асоціативного оброблення данних з орієнтацією на медичне діагностування.

Для цього випадку як приклад можна навести класичну модель класифікації за лінійними дискримінантними функціями (ЛДФ) у вигляді таких базових співвідношень [29,31,32]:

$$g_i(X) = \sum_{j=1}^{n} w_{ij} \cdot x_j - w_{io} \cdot x_o, i = 1, \dots, m \qquad (1.4)$$

$$X \in C_k \Leftrightarrow k = argmax\{g_i(X)\}, k = 1, \dots, m \qquad (1.5)$$

де x_j - j-й елемент n-вимірного вхідного вектора ознак X; w_{ij} - ваговий коефіцієнт j-го входу i-ої ЛДФ $g_i(X)$; $w_{io} \cdot x_o$ - вільний елемент i-ої ЛДФ $g_i(X)$; C={ C_1 ,..., C_m } - множина класів; m- кількість класів.

В результаті базова структура класифікатора об'єктів має вигляд як на рис.1.3

Формувач ЛДФ у структурі класифікатора реалізує функцію (1.4) в процесі векторно-матричного множення, а максимізатор типу "1 з N" - функцію (1.5), тобто визначає k-й клас з позначенням виходу y_k за максимальною величиною k-ої ЛДФ $g_k(X)$ серед сформованих m ЛДФ.

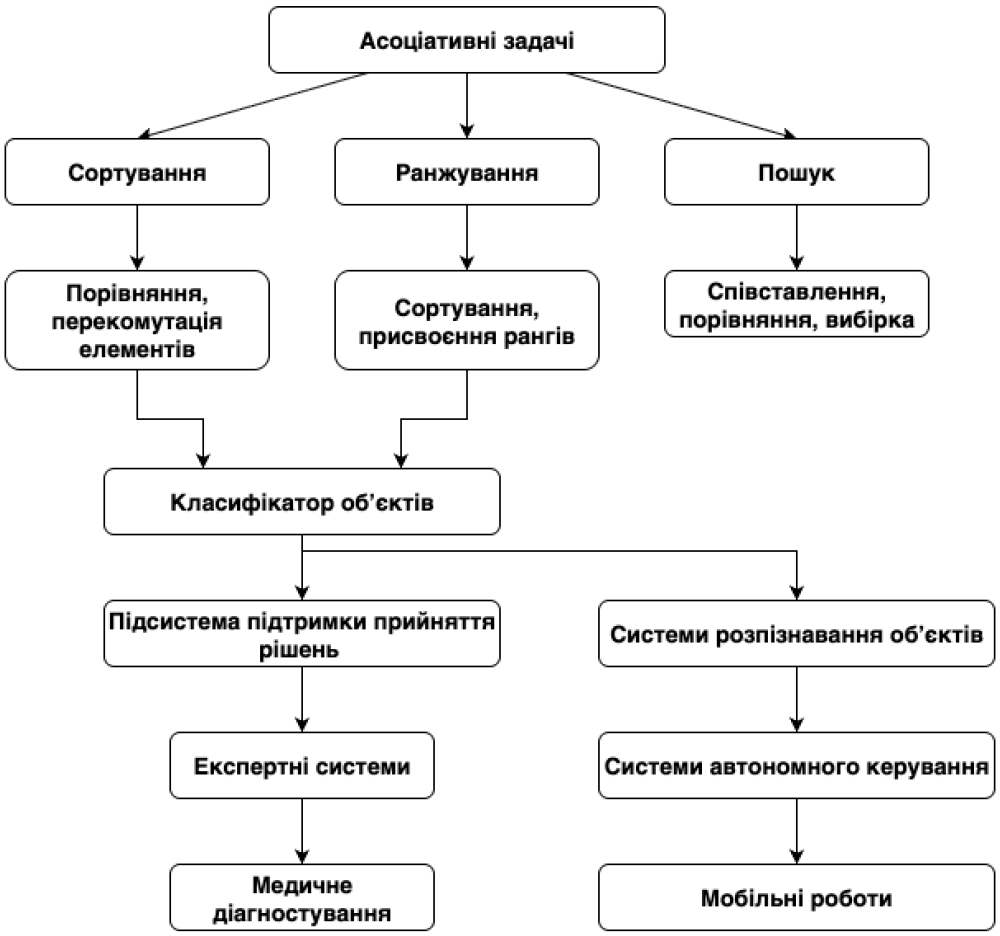


Рисунок 1.2 – Асоціативний аспект оброблення даних

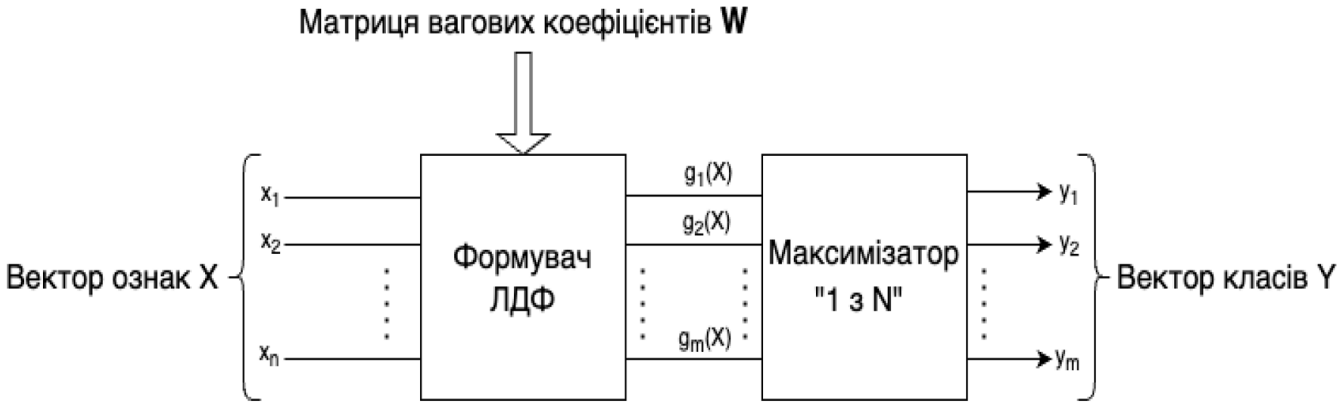


Рисунок 1.3 - Базова структура класифікатора об’єктів

Аналіз структури класифікатора об’єктів (рис. 1.3) показав, що найбільш перспективним з точки зору вдосконалення структурного та функціонального базису класифікатора є максимізатор. Якщо розглядати нейромережний варіант такої структури класифікатора, то функцію формувача ЛДФ може виконувати одношаровий персептрон, а функцію максимізатора - деякі мережі типу MAXNET [4,6], серед яких або нейромережа з від’ємними латеральними зв’язками у нейронів конкурентного шару, або нейромережа зі структурою Feed-Forward MAXNET.

Разом з тим, як для нейромережного, так і для класичного варіантів структури класифікатора одним із способів реалізації функції максимізатора можна задіяти асоціативну операцію сортування одновимірного (векторного) масиву чисел, оскільки в результаті можна визначити максимальний за значенням елемент числового масиву [20, 33].

1.2 Особливості позрізового оброблення одновимірного масиву даних на базі SM-перетворення

Відомий метод оброблення за різницевими зрізами в базисі SM-перетворення векторних (одновимірних) масивів чисел досить докладно розглянуто у працях [34,35], де доведено ефективність такого методу оброблення на прикладі паралельного багатоопérандного підсумовування векторного масиву чисел. Разом з тим, у працях [34,35] математично доведено багатофункціональність методу різницево-зрізового оброблення, а саме, можливість формування певних матриць бінарних масок, що дозволяє не тільки відновити за необхідності початковий масив чисел, що був обнуленний в процесі оброблення, але й відсортувати його елементи за зростанням їх значень [36].

Отже, процес сортування числових даних залишається досить затребуваною процедурою [1,37,38] у прикладних задачах. Зокрема, це стосується медіанної фільтрації в процесі передоброблення зображень та сигналів [9, 39]. При цьому значна увага приділяється новим підходам до апаратної реалізації процесу сортування значних масивів даних, що дозволяє збільшити його швидкодію [2,7]. Це досягається або за рахунок задіяння значної кількості апаратних засобів, що спрацьовують паралельно [37,38], або за рахунок використання швидкісних операції інкремента/декремента [20] замість вузлів попарного порівняння та перекомутації елементів числового масиву.

При SM-перетворенні векторного масиву чисел базовою операцією є послідовне формування різницевих зрізів (РЗ) у векторному вигляді, при цьому розглядаючи початковий числовий масив як нульовий n-вимірний РЗ $\boldsymbol{a}_0$, де n – кількість елементів масиву. Для формування кожного поточного РЗ $\boldsymbol{a}_j$, j =

1,…,N, де N - кількість циклів оброблення, задіяно ненульовий внутрішній поріг q_j оброблення, який визначається таким чином [34,35]:

$$q_j = min\boldsymbol{a}_{j-1} = min\{a_{i,j-1}\}_1^n, \tag{1.6}$$

де $a_{i,j-1}$ – i-й елемент попереднього (j-1)-го РЗ $\boldsymbol{a}_{j-1}$. В результаті поточний РЗ $\boldsymbol{a}_j$ формується таким чином:

$$\boldsymbol{a}_j = \{a_{i,j}\}_1^n = \{a_{i,j-1} - q_j\}_1^n, \tag{1.7}$$

тобто у векторному вигляді:

$$\boldsymbol{a}_j = \boldsymbol{a}_{j-1} - q_j. \tag{1.8}$$

Крім того, одночасно можна сформувати зрізи $\boldsymbol{g}_j$ і $\boldsymbol{f}_j$ як вектор-стовпці відповідно двох матриць бінарних масок **G** і **F** [34,35], кожний елемент яких визначається таким чином:

$$g_{i,j} = \begin{cases} 1, \text{якщо } a_{i,j} = 0, \\ 0, \text{якщо } a_{i,j} \neq 0, \end{cases} \tag{1.9}$$

$$f_{i,j} = \begin{cases} 1, \text{якщо } a_{i,j} \geq 0, \\ 0, \text{якщо } a_{i,j} < 0, \end{cases} \tag{1.10}$$

Обидві сформовані матриці бінарних масок **G** і **F** фактично є матрицями відповідно нульових та додатних ознак згідно із виразами (1.9) і (1.10). Ці ознаки містять інформацію про внутрішнє топологічне розташування елементів початкового масиву $\boldsymbol{a}_0$, в залежності від їх числових значень. Все це дозволяє реалізувати сортування і ранжування елементів масиву $\boldsymbol{a}_0$, остільки топологічні ознаки відіграють головну роль в цих асоціативних операціях [36,40].

1.2.1 Векторно-матричні операції в базисі SM-перетворення

Попередні дослідження [34,35] показали, що відсортований $\boldsymbol{a}_0^S$ початковий масив $\boldsymbol{a}_0$ можна отримати як результат операції векторно-матричного множення вигляду:

$$\boldsymbol{a}_0^S = \boldsymbol{G}^T \cdot \boldsymbol{a}_0, \tag{1.11}$$

де $\boldsymbol{a}_0^S$ - векторний масив елементів початкового РЗ $\boldsymbol{a}_0$, що відсортовані за зростанням їх числових значень.

Одночасно, в процесі різницево-зрізового оброблення у кожному циклі можна поступово сформувати послідовність елементів відсортованого масиву $\boldsymbol{a}_0^S$, використовуючи залежність (1.6) таким чином [35]:

$$a_i^S = \sum_{j=1}^{i} q_j \quad i = 1, \dots, n. \tag{1.12}$$

Крім того використовуючи матрицю бінарних масок **F** та вектор внутрішніх порогів вигляду:

$$\boldsymbol{q} = \{q_1, \dots q_n\}, \tag{1.13}$$

можна відновити початковий масив $\boldsymbol{a}_0$ таким чином [34,35]

$$\boldsymbol{a}_0 = \boldsymbol{F} \cdot \boldsymbol{q}. \tag{1.14}$$

При цьому необхідно зауважити, що процес різницево-зрізового оброблення завершується за наявності нульового значення q_j [34,35].

1.3 Нейромережний підхід до медичної експрес-діагностики

Медичні експертні системи широко використовуються для діагностування, моніторингу, прогнозування, підтримки прийняття рішень [10,41]. Сучасні системи медичної діагностики можуть виконувати складні і важливі завдання медичної експрес-діагностики, а саме [10,41]: класифікувати вид захворювання (диференціальна діагностика); визначати найдоцільніший спосіб лікування; прогнозувати тривалість і результат захворювання; оцінювати ризик можливих ускладнень; виявляти основні синдроми; знаходити найхарактерніші для певного захворювання сукупності симптомів.

З точки зору цінності можна зауважити, що медичні експертні системи здатні миттєво аналізувати і узагальнювати безліч чинників і прецедентів, така можливість недоступна людині, навіть якщо вона є фахівцем [10,41]. На сьогоднішній день в медичній практиці все частіше застосовують комп'ютерні технології [10]. Лікарі вдаються до послуг роботів під час операцій, а комп'ютерна діагностика взагалі вважається нормою.

1.3.1 Підсистема формування висновків та рекомендацій

Базова структура експертної системи (ЕС) показано на рис. 1.4. Вона реалізує дві основні функці в комп'ютерній програмі інтерфейсу, а саме навчання в експерта і ведення діалогу з користувачем, тобто здатна давати поради, пояснення, ставити запитання користувачеві [10,12,28,41].

В роботі [43] наведено автоматизований медичний комплекс (АМК) для моніторингу та діагностики фізичного здоров'я, що підвищує ефективність інформаційних процесів взаємодії пацієнтів з комплексом спеціалізованого контролю за динамікою їх стану. В АМК інформація про всі дослідження

заноситься до блока отримання оцінок діагностичних ознак, а результати обстеження зберігаються в базі даних.

Підсистема підтримки прийняття рішень збирає та аналізує інформацію, а підсистема порівняння результатів діагностики надає інформацію про всі результати діагностування пацієнта, дані в яку надсилаються з підсистеми формування висновків і рекомендацій.

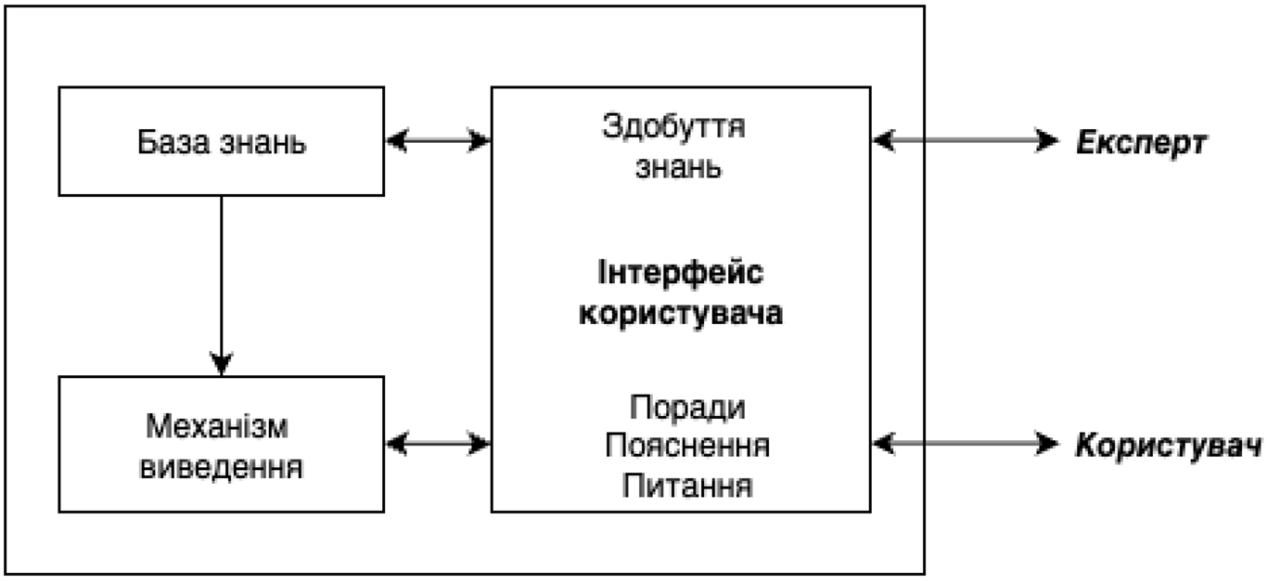


Рисунок 1.4 - Базова структура експертної системи

Отже, автоматизоване робоче місце лікаря виконує функцію підсумовування проведення огляду і виведення на екран результатів у зрозумілій для користувача формі [43]. Для вдосконалення роботи АМК за рахунок застосування нейромережевих технологій особливий інтерес становить підсистема формування висновків та рекомендацій.

Комплексний метод діагностування в АМК можна представити алгоритмічно у вигляді сімохетапів [43], починаючи з аналітичного етапу і завершуючи етапом телемедичної підтримки. На рис. 1.5 показано блок-схему

алгоритму роботи АМК у режимі поглибленого обстеження пацієнтів [43]. Для такого підходу важливим є блок 8, де визначається ступінь відхилення отриманихрезультатів від діапазону значень, за якого приймається одне з трьох рішень щодо лікування.

Але разом з тим, у випадку значних відхилень (блок 10) виконується перевірка повноти отриманої інформації, тобто саме після цього ставиться остаточний діагноз (блоки 14 і 15). Біомедичні експертні системи [29] розробляються для вирішення різних проблем, але основні типи їх діяльності можна згрупувати в категорії, наведені в таблиці 1.1 [42].

Таблиця 1.1

Категорії біомедичних експертних систем

Категорія	Проблема що вирішується
Інтерпретація	Опис ситуації за інформацією від сенсора
Діагностика	Виявлення причин захворювання
Прогноз	Визначення ймовірних наслідків певних ситуацій
Лікування, реабілітація	Виконання послідовності запропонованих дій, спрямованих на приведення до норми
Навчання	Діагностика і корекція поведінки
Планування	Визначення послідовності дій
Управління	Управління станом об’єкта

Застосування нейромережевого підходу до медичного діагностування, а саме на базі дискримінантного аналізу [30-32] з поєднанням нейромережі у складі підсистеми формування висновків і рекомендацій дозволяє зробити процес підтримки прийняття рішень гнучкішим через здатність такої

підсистеми навчатись [4,5]. При цьому зміни стосуються блоків 8-11, 15 у блок-схемі алгоритму (див. рис. 1.5).

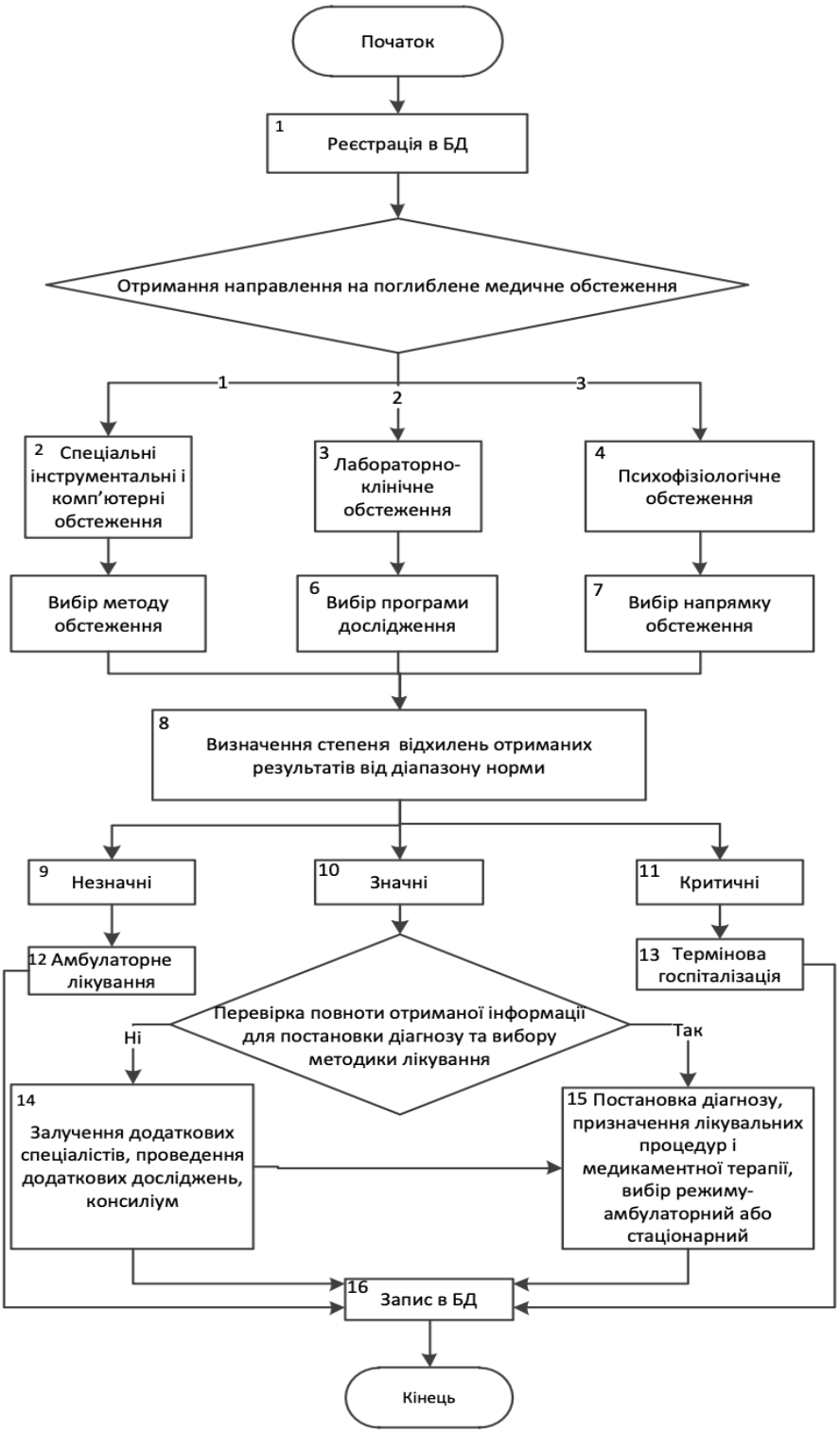


Рисунок 1.5 - Блок-схема алгоритму роботи АМК [44]

1.3.2 Приклад побудови системи медичної експрес-діагностики

Пропонується варіант підсистеми формування висновків та рекомендацій (рис. 1.6) у складі АМК[44].

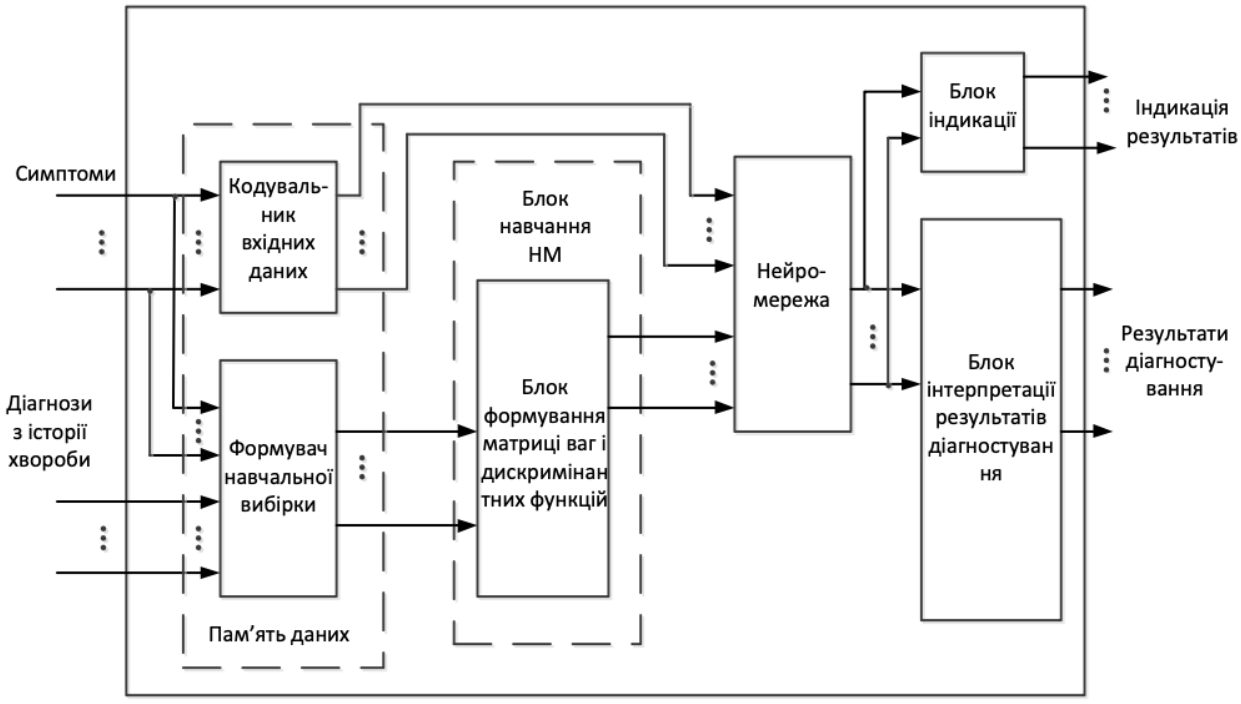


Рисунок 1.6 - Підсистема формування висновків та рекомендацій

Наведена на рис.1.6 підсистема містить кодувальник вхідних даних, формувач навчальної вибірки, блок формування матриці ваг і дискримінантних функцій, нейромережу, блок індикації та блок інтерпретації результатів діагностування. На вхід підсистеми подаються діагнози з історії хвороби, які надходять до формувача навчальної вибірки, та симптоми, які надходять як до кодувальника вхідних даних, так і до формувача навчальної вибірки. А на виході підсистеми формується результат діагностування з його індикацією.

Базовим блоком підсистеми є нейромережа, яка виконує функції класифікатора, що дозволяє діагностувати певні захворювання за їх симптомами. У підсистемі враховано таку важливу властивість нейронних мереж, як здатність навчатися. В процесі навчання нейронна мережа знаходить приховані нелінійні залежності між вхідними параметрами і кінцевим рішенням, а також оптимальну комбінацію вагових коефіцієнтів нейронів, що з'єднують сусідні шари, за якої похибка визначення класу образу (діагнозу) прагне до мінімуму [4,5].

Основні переваги нейромережевих експертних систем перед звичайними такі [4,6,30,32]:

1) нейромережі приймають рішення на основі досвіду, придбаного ними самостійно;
2) рішення, прийняте нейромережею, не є категоричним: мережа видає рішення разом зі ступенем впевненості в ньому, що залишає користувачеві можливість критично оцінювати її відповідь;
3) нейромережа дозволяє моделювати ситуацію прийняття рішення;
4) нейромережі дають відповідь в секундному діапазоні, що дозволяє їх в різних динамічних системах, які вимагають негайного прийняття рішення.

Все це доводить необхідність, актуальність та затребуваність використання штучних нейронних мереж для розв'язання медичних задач [30,46,47].

1.3.3 Особливості нейромережного класифікатора

Використання нейромережних технологій є пріоритетним напрямком в таких прикладних галузях, як медична діагностика, економіка, бізнес, зв'язок (телекомунікації), інтернет, безпека тощо [30, 50-51]. Для медичного діагностування разом з кластеризацією доволі часто використовується дискримінантний аналіз, а саме, класифікація об'єктів за дискримінантними

функціями [30-32]. Цей підхід зумовлює використання, наприклад, класифікатора на базі нейромережі Хеммінга [4,5] з подальшим вдосконаленням його структури [13]. Отже, використання дискримінантного аналізу в АМК замість визначення ступеня відхилення результатів від діапазону норми (блоки 8-11,15 на рис. 1.5) дозволяє використовувати нейротехнологію.

Структуру одного з варіантів нейромережевого класифікатора на базі вдосконаленої нейромережі Хеммінга показано на рис. 1.7 [13]. Такий класифікатор містить три шари взаємозв'язаних нейроелементів. Виходи бінарних нейроелементів вихідного шару є виходами ознаки належності вхідних сигналів відповідному класу, причому прихований шар складається з *m* лінійних нейроелементів, де *m* - кількість класів, а кожний з n входів класифікатора з'єднаний з входом відповідного сенсорного нейроелемента вхідного шару.

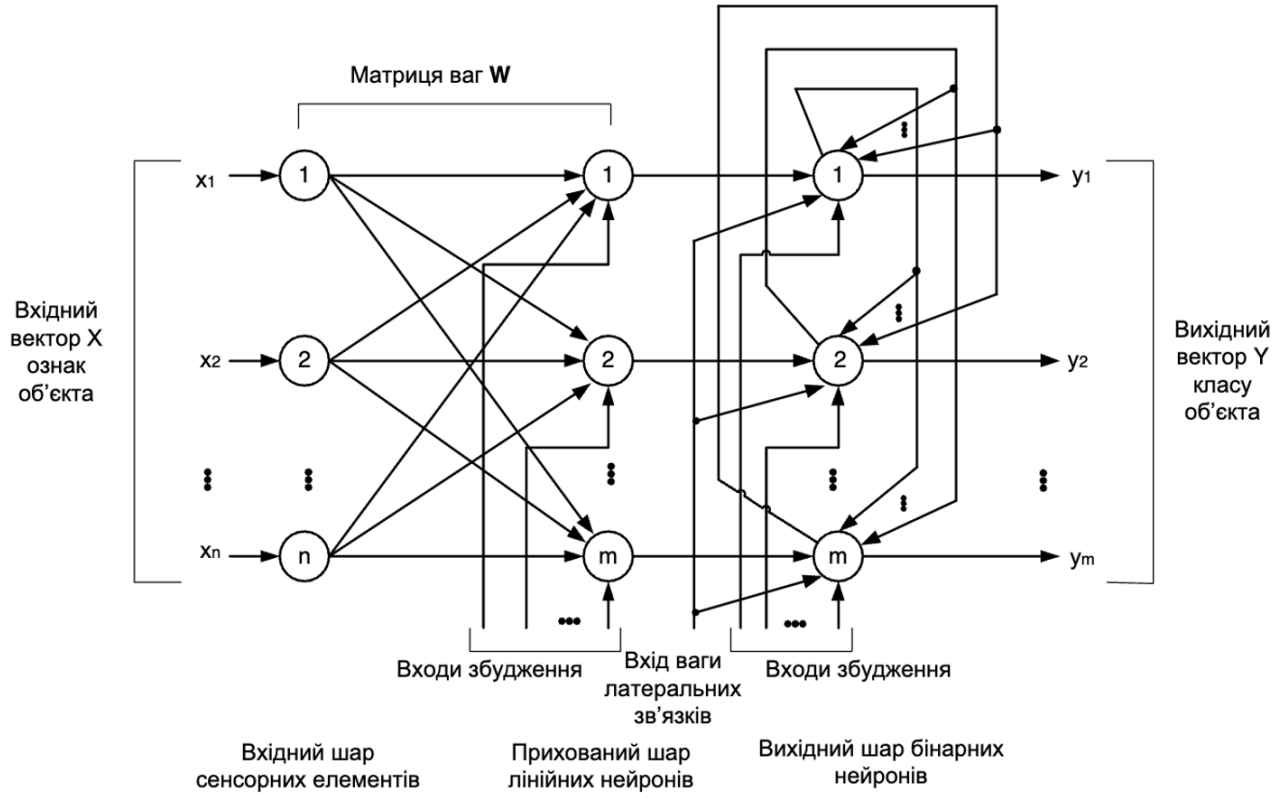


Рисунок 1.7 - Структурна схема класифікатора

Перший етап роботи класифікатора - це етап налаштування, на якому встановлюються значення ваг $w_{ij}{}^{(2)}$ зв'язків входів нейронів прихованого шару з виходами сенсорних елементів вхідного шару, тобто відбувається навчання класифікатора без вчителя [4,5].

Другий етап - робочий, на якому відбувається основне функціонування класифікатора, при цьому на входи класифікатора подається n-елементний вхідний вектор **X** ознак з елементами x_j, де $j = 1, \dots, n$. На виході i-го нейрона прихованого шару формується сума S_i зважених вхідних сигналів $x_1, \dots, x_n$, тобто відповідна дискримінантна функція вигляду

$$S_i = \sum_{\boldsymbol{j=0}}^{\boldsymbol{n}} w_{ij}{}^{(2)} x_j + b_i\,, i = 1, \dots, m. \tag{1.15}$$

де b_i - зовнішній сигнал збудження класифікатора. Отримані значення дискримінантних функцій $S_1, \dots, S_m(1)$ задають початкові стани бінарних нейронів вихідного шару. Після цього надходження сигналів S_i припиняється, а зі сформованого цими сигналами початкового стану запускається ітераційний процес всередині вихідного шару. Нейрони вихідного шару зв'язані між собою латеральними зв'язками, які мають вагу $w_{il}{}^{(3)}$ вигляду

$$w_{il}{}^{(3)} = \begin{cases} 0, \text{якщо i} = l\,, \\ -\varepsilon \le \dfrac{1}{m}, \text{якщо i} \ne l, \end{cases} \tag{1.16}$$

де ε - значення ваги латеральних зв'язків. Отже, кожний i-й бінарний нейрон вихідного шару з'єднаний від'ємним (гальмівним) латеральним зв'язком з додатковими виходами нейронів цього шару, крім себе самого. Функція

активації $f^1(S_i)$ на додаткових виходах відповідних бінарних нейронів вихідного шару має вигляд

$$f^1(S_i) = \begin{cases} S_i, \text{якщо } S_i > 0, \\ 0, \text{якщо } S_i \leq 0, \end{cases} \quad (1.17)$$

На початку процесу функціонування нейронів вихідного шару на їх виходах встановлюються одиничні значення за зовнішніми сигналами збудження.

Отже, бінарні нейрони вихідного шару функціонують в режимі WTA (Winner Takes All), в якому у фіксованій (кінцевій) ситуації активізується тільки один бінарний нейрон, а всі інші перебувають у стані спокою. Функція активації $f^2(S_i)$ на виходах відповідних бінарних нейронів вихідного шару задається виразом

$$f^2(S_i) = \begin{cases} 1, \text{якщо } S_i > 0, \\ 0, \text{якщо } S_i \leq 0, \end{cases} \quad (1.18)$$

Ітераційний процес завершується у момент, коли всі нейрони вихідного шару, крім одного бінарного нейрона (переможця з вихідним сигналом, не рівним нулю), перейдуть в нульовий стан. Нейрон-переможець з ненульовим вихідним сигналом y_k є представником k-го класу, до якого належить вхідний вектор **X**.

Таким чином, на виході ненульового нейрона вихідного шару формується в класифікаторі відгук у вигляді одиничного сигналу y_k відповідно до виразу (4), що дає можливість візуалізувати результати діагностування за допомогою лінійки світлодіодів [48]. Свічення певного світлодіода дозволитьшвидко ідентифікувати отриманий результат на виході нейромережевого класифікатора [49].

У запропонованому нейромережевому класифікаторі [44] використано критерій максимуму лінійних дискримінантних функцій [32]. В класифікації об’єктів (симптомів) такий підхід широко використовується у медичному діагностуванні [30-32], що забезпечує за певних умов мінімум критерію середньої імовірності помилкової класифікації. На рис 1.7 показано структуру нейромережевого класифікатора [44], в табл. 1.2 наведено дані про його функціональні особливості, а у табл. 1.3 наведено його структурні особливості[52].

Таблиця 1.2

Функціональні та навчальні особливості нейромережного класифікатора

Назва мережі	Властивість мережі	Навчання	Переваги	Недоліки
Нейромережевий Класифікатор Neural Network Classifier	Класифікація об’єктів за максимумом серед дискримінантних функцій	В процесі налаштування мережі ваги прихованого шару і зміщення: за коефіцієнтами дискримінантних функцій. Ваги зворотних зв’язків вихідного шару: $0 < -\varepsilon < 1/m$ (без навчальних ітерацій)	Можливість класифікації у медичному діагностуванні і інших патологій за умови зміни матриці ваг та перекодування вхідних векторів (симптомів)	Орієнтація на конкретну задачу класифікації. В протилежному випадку необхідна зміна у структурі мережі: кількості входів і виходів і, відповідно, кількості нейронів в обох шарах

Таким чином, розглянутий нейромережевий класифікатор за структурною організацією є подальшим удосконаленням мережі Хеммінга, отже, є рекурентною мережею [4-6] і його можна віднести до різновиду гетероасоціативної пам’яті.

Але у функціональному плані нейромережевий класифікатор має значно розширенні властивості, оскільки може обробляти не тільки бінарні вхідні сигнали, але й розпізнавати їх за складнішим критерієм класифікації, а саме, за максимумом серед дискримінантних функцій, які формуються прихованим шаром лінійних нейронів (рис.1.7).

Таблиця 1.3

Структурні особливості нейромережевого класифікатора

Назва мережі	Структура мережі			
	Кількість шарів	Кількість нейронів у шарах	Зв’язки між шарами	Функції активації
Нейромережевий класифікатор Neural Network Classifier	Три шари: вхідний, прихований, вихідний.	n - кількість входів; m - кількість виходів мережі; m - кількість нейронів у прихованому та вихідному шарах	Зворотний зв'язок у вихідному шарі: від’ємні зворотні (інгібіторні) зв’язки усіма, крім себе самого ($w_{ii} = 0$)	Лінійна симетрична у прихованому шарі. Порогова несиметрична у вихідному шарі: $f(S_i) = \begin{cases} 1, \text{якщо } S_i > 0, \\ 0, \text{якщо } S_i \leq 0, \end{cases}$

1.3.4 Застосування дискримінантного аналізу для класифікації об’єктів

Дискримінантний аналіз є одним із ключових методів математичної статистики та машинного навчання, що використовується для класифікації об’єктів на основі їхніх ознак. Основна ідея методу полягає у побудові дискримінантних функцій, які дозволяють віднести досліджуваний об’єкт до певного класу, мінімізуючи ймовірність помилки. Цей підхід має важливе значення при обробленні багатовимірних даних, особливо у випадках, коли спостерігаються кореляції між змінними та накладання класів.

Методи дискримінантного аналізу ґрунтуються на пошуку гіперплощини, що найкраще розділяє простір ознак між класами. Для цього визначається така лінійна комбінація вхідних змінних, яка забезпечує максимальне співвідношення між дисперсією міжкласових відмінностей та внутрішньокласових коливань. Отримана функція називається лінійною дискримінантною функцією (ЛДФ). У більш складних випадках, коли класи не можуть бути лінійно розділені, застосовують квадратичний дискримінантний аналіз або нелінійні узагальнення.

Теоретичні основи дискримінантного аналізу Нехай існує множина об’єктів, кожен з яких описується вектором ознак $X = [x_1, x_2, ..., x_n]$. Об’єкти належать до одного з K класів $C_1, C_2, ..., C_k$. Завдання полягає у побудові дискримінантних функцій $f_1(X), f_2(X), ..., f_k(X)$, які дозволяють визначити, до якого класу належить невідомий об’єкт X_0. Найчастіше приймається рішення за правилом максимуму

У практичних застосуваннях дискримінантний аналіз використовується в різних сферах: біометрії, медичній діагностиці, технічній експертизі, системах розпізнавання образів, фінансовій аналітиці тощо. Наприклад, у медичних

задачах ЛДФ дозволяє визначати наявність патологій на основі сукупності фізіологічних показників пацієнта. У промислових системах контроль якості може бути реалізований шляхом аналізу параметрів сигналів сенсорів з подальшою класифікацією продуктів як «якісних» або «дефектних».

Особливої уваги заслуговує застосування дискримінантного аналізу у високопродуктивних інтелектуальних системах, де класифікація об’єктів відбувається у режимі реального часу. Для цього необхідно адаптувати обчислювальні структури, зокрема нейроподібні класифікатори, які реалізують дискримінантні функції на апаратному рівні з використанням паралельної обробки даних. Такі системи забезпечують швидке прийняття рішень і можуть бути використані в системах автономного керування мобільних роботів.

Класичний алгоритм дискримінантного аналізу передбачає обчислення середніх векторів для кожного класу, матриці коваріацій та оберненої матриці дисперсій. Проте при великій кількості ознак і невеликих вибірках виникають проблеми надлишкової кореляції, що може призвести до нестійкості моделі. Для подолання цих недоліків застосовують регуляризацію або метод головних компонент для зменшення розмірності простору ознак перед побудовою ЛДФ.

Крім того, сучасні підходи інтегрують дискримінантний аналіз із нейронними мережами, використовуючи гібридні архітектури. Такі моделі поєднують здатність нейронних мереж до нелінійного відображення даних із стабільністю статистичних методів дискримінантного аналізу. Розробка таких систем відкриває перспективи для створення адаптивних класифікаторів, здатних працювати з неповними або зашумленими даними.

Порівняно з методами на основі дерев рішень, байєсівських мереж чи SVM, дискримінантний аналіз має переваги у швидкості обчислення та зрозумілості результатів. Його математична структура дозволяє інтерпретувати вплив кожної ознаки на результат класифікації, що важливо у прикладних

задачах, де необхідно пояснювати прийняті рішення. Однак метод має і обмеження: він чутливий до нормальності розподілу та рівності дисперсій у класах, що не завжди виконується у реальних даних.

Розглянемо приклад застосування дискримінантного аналізу для класифікації медичних зображень. Нехай кожен об’єкт описується набором параметрів: контраст, яскравість, текстурна щільність, симетрія та інші статистичні ознаки. На основі навчальної вибірки формується матриця ознак і обчислюються дискримінантні коефіцієнти. Після цього для кожного нового зображення обчислюються значення ЛДФ, а класифікація відбувається за правилом найбільшого значення функції. Такий підхід дозволяє автоматизувати процес діагностики та підвищити точність розпізнавання патологій.

Дискримінантний аналіз є універсальним інструментом для вирішення задач класифікації в інтелектуальних системах. Його поєднання з методами паралельної обробки даних, нейроподібними структурами та сучасними обчислювальними архітектурами дозволяє підвищити швидкодію та точність прийняття рішень. Перспективним напрямом подальших досліджень є реалізація дискримінантних алгоритмів у вигляді спеціалізованих апаратних модулів, оптимізованих для роботи в системах реального часу, зокрема для автономних роботизованих платформ.

1.4 Порівняльний аналіз дискримінантного аналізу з іншими методами класифікації

1.4.1 Порівняння дискримінантного аналізу та методів на основі дерев рішень

Методи класифікації на основі дерев рішень та дискримінантного аналізу належать до фундаментальних інструментів інтелектуальної обробки даних,

однак вони ґрунтуються на принципово різних концепціях і мають різні можливості з точки зору інтерпретованості, обчислювальної складності та придатності до апаратної реалізації. Порівняння цих методів є важливим для побудови високопродуктивних класифікаційних систем, особливо в тих випадках, коли висуваються жорсткі вимоги до швидкодії та ресурсообмеженості, наприклад у системах автономного керування мобільних роботів або системах технічного зору.

Дерева рішень реалізують класифікацію шляхом послідовного поділу простору ознак на підмножини за допомогою логічних правил виду *«якщо-то»* (рис.1.8). На кожному вузлі дерева здійснюється вибір найбільш інформативної ознаки відповідно до міри інформаційного приросту, ентропії або індексу Джині. Це надає методу високої інтерпретованості, оскільки модель можна представити у вигляді набору зрозумілих правил. Водночас підхід на основі дерев рішень є нерегулярним: структура дерева залежить від статистичних властивостей вибірки та може суттєво відрізнятися навіть за незначних змін даних. Така нестабільність структури призводить до того, що глибина дерева, кількість гілок і логічних операцій на шляху класифікації не є фіксованими.

У свою чергу, дискримінантний аналіз ґрунтується на побудові лінійних або квадратичних дискримінантних функцій, які відображають об’єкти багатовимірного простору у скалярні значення, що дозволяє здійснити розділення класів за принципом максимального розділення міжкласових та мінімізації внутрішньокласових варіацій. Основна перевага методу полягає у строгій математичній формалізації — вагові коефіцієнти ЛДФ визначаються аналітично на основі статистичних характеристик вибірки. Обчислювальні процедури при цьому є лінійними й детермінованими, що забезпечує прогнозований час виконання та стабільність класифікаційного рішення.

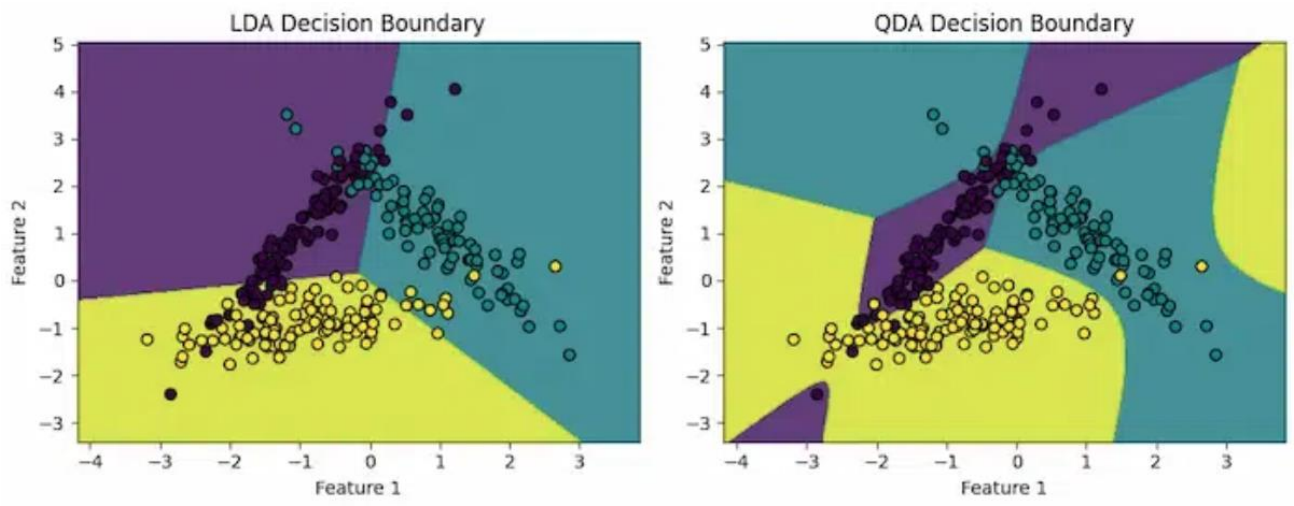


Рисунок 1.8 - Лінійна межа ЛДФ в порівнянні з деревом рішень

У порівнянні з деревами рішень, дискримінантний аналіз демонструє вищу стійкість до варіацій у навчальних даних. Оскільки модель представлена системою вагових коефіцієнтів, її структура не змінюється при розширенні або частковому оновленні вибірки. Це контрастує з деревами, де мінімальні зміни даних можуть призвести до повної реконфігурації дерева, зменшення інтерпретованості або погіршення узагальнювальної здатності. Додатково, моделі на основі дерев часто страждають від перенавчання та потребують оптимізації параметрів глибини, критерію зупинки та методів обрізки дерева.

Дерева рішень (CART, C4.5) виконують класифікацію через ієрархічне розгалуження простору ознак їхні переваги:

- інтерпретованість;
- робота з категоріальними ознаками;
- стійкість до шуму.

Однак недоліки критичні для систем реального часу:

- нерівномірна структура дерева → нерівномірні часові витрати;
- складність глибинної апаратної реалізації через *послідовну* природу прийняття рішень;

- низька стабільність при змінах вибірки (висока варіативність структури дерева).

Дискримінантний аналіз виграє тим, що:

- обчислюється через *фіксовану кількість операцій*;
- всі операції — *алгебраїчні* (множення, додавання), які легко реалізуються апаратно;
- класифікаційне рішення приймається за правилом max, що природно лягає на паралельні нейроподібні структури.

З позиції апаратної реалізації дискримінантний аналіз має суттєві переваги. Дерева рішень вимагають реалізації багатьох умовних переходів, що утворює нерівномірний обчислювальний граф. У ПЛІС або ASIC реалізація таких умовних структур потребує великої кількості комутуючих елементів, що знижує швидкодію та збільшує складність маршрутизації сигналів. Дискримінантний аналіз, навпаки, складається з регулярних операцій множення та додавання, які легко реалізуються у вигляді систолічних масивів із повторюваними осередками. Така регулярність обчислень дозволяє реалізувати паралельне обчислення множинних ЛДФ з наступною апаратною реалізацією процедури WTA-конкуренції.

Отже, дискримінантний аналіз має значні переваги у задачах апаратної класифікації завдяки регулярності обчислень, детермінованості, стійкості моделі та мінімальним вимогам до ресурсів, тоді як дерева рішень більше підходять для програмних систем, орієнтованих на інтерпретованість, але не на апаратну ефективність.

1.4.2 Порівняння дискримінантного аналізу з методом k-ближчих сусідів

Метод k-ближчих сусідів (k-Nearest Neighbors, k-NN) та дискримінантний аналіз (ДА) належать до широко застосовуваних алгоритмів класифікації, однак вони реалізують принципово різні підходи до побудови рішень та характеризуються різними вимогами до обчислювальних ресурсів. Порівняння цих методів є особливо важливим у контексті систем, де пред'являються високі вимоги до швидкодії, енергоефективності та можливості апаратної реалізації класифікатора.

Метод k-NN є прикладом інстанс-орієнтованої класифікації (рис. 1.9), в якій модель безпосередньо зберігає всі навчальні дані, а класифікаційне рішення приймається шляхом обчислення відстаней між новим об'єктом та всіма зразками навчальної вибірки. Такий підхід забезпечує високу гнучкість і часто хорошу якість класифікації, особливо в задачах з нелінійними границями між класами. Проте ці переваги супроводжуються суттєвими обмеженнями: складність класифікації становить O(N × d), де N — кількість еталонних зразків, а d — кількість ознак. Це означає, що час обчислення прямо пропорційний обсягу навчальної вибірки. Тому в системах реального часу k-NN швидко втрачає ефективність.

Дискримінантний аналіз, навпаки, належить до параметричних методів. На етапі навчання він оцінює набір вагових коефіцієнтів дискримінантних функцій, які потім використовуються для класифікації нового об'єкта. Таким чином, модель відображається через мінімальну кількість параметрів — лише вагові коефіцієнти та константи для кожного класу. Класифікаційне рішення формується шляхом обчислення лінійної комбінації ознак, що забезпечує час виконання O(d), тобто незалежний від обсягу навчальних даних.

Порівнюючи k-NN та ДА з точки зору стійкості до шуму, k-NN показує хороші результати лише за умови правильно підібраної метрики відстані та значення k. Однак у випадках, коли дані містять багато нерелевантних ознак або значний шум, метод k-NN може давати нестабільні результати через рівнозначну вагу всіх зразків при обчисленні відстаней.

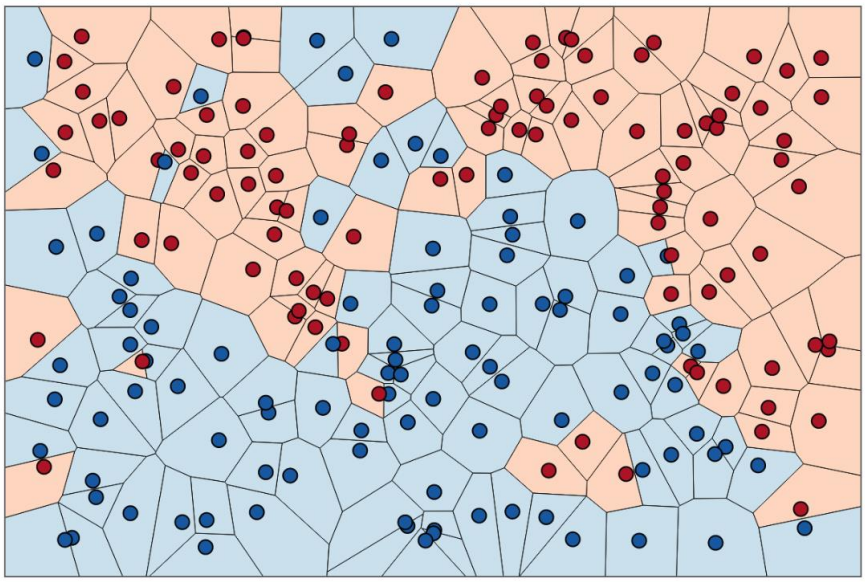


Рисунок 1.9 - Візуалізація роботи k-NN у просторі ознак

Дискримінантний аналіз у таких умовах також може втрачати точність через порушення припущень щодо нормальності або рівності коваріацій матриць, проте його модель залишається стабільною та легко коригується за допомогою регуляризації або зменшення розмірності.

Найбільш суттєві відмінності між методами проявляються у вимогах до пам’яті та можливості апаратної реалізації. Метод k-NN вимагає зберігання повного набору навчальних даних, що є критичним недоліком у вбудованих або апаратно обмежених системах. Апаратна реалізація k-NN потребує або великої кількості енергонезалежної пам’яті, або складної структури паралельних модулів для одночасного обчислення відстаней до великої кількості зразків. Це призводить до високої апаратної складності та затримок у прийнятті рішення.

Дискримінантний аналіз, своєю чергою, оперує невеликим набором коефіцієнтів, що дозволяє реалізувати класифікатор у вигляді компактного систолічного масиву або набору суматорів і множників. Висока регулярність структури обчислень і можливість паралельного виконання операцій для кількох класів робить ДА надзвичайно ефективним саме для апаратної реалізації. У системах, де критично важливо забезпечити стабільний, детермінований час класифікації, дискримінантний аналіз стає значно привабливішим порівняно з k-NN.

Метод k-NN простий і дає хорошу точність, але має великі недоліки:

- потрібен доступ до всієї навчальної вибірки під час класифікації;
- обчислювальна складність O(N × d) для кожного нового об’єкта;
- слабка придатність до апаратної реалізації через необхідність зберігати великі масиви даних.

У системах реального часу, наприклад в автономному мобільному роботі, k-NN є неприйнятним, бо:

- потребує багато пам’яті,
- погано масштабується,
- вимагає багаторазового паралельного обчислення відстаней.

Переваги ЛДФ:

- модель компактно описується векторами коефіцієнтів;
- рішення приймається за мілісекунди;
- обсяг пам’яті фіксований і мінімальний.

Таким чином, хоча k-NN має певні переваги в умовах складних нелінійних розділень класів та великої кількості навчальних даних, дискримінантний аналіз значно перевершує його у задачах реального часу та в

середовищах із обмеженими обчислювальними ресурсами. Мінімальні вимоги до пам’яті, лінійність, регулярність обчислень та придатність до паралельної апаратної реалізації роблять дискримінантний аналіз одним з найбільш ефективних методів для реалізації високошвидкісних класифікаторів.

1.4.3 Порівняння дискримінантного аналізу з нейронними мережами

Нейронні мережі є одним із найпотужніших сучасних інструментів для класифікації, розпізнавання образів та прогнозування. Вони здатні апроксимувати нелінійні залежності довільної складності, що робить їх придатними для широкого спектра задач — від обробки зображень до аналізу сигналів та елементів поведінки автономних систем. Однак у порівнянні з дискримінантним аналізом (ДА) нейронні мережі мають суттєво інші властивості з точки зору структури обчислень, вимог до ресурсів та можливостей апаратного прискорення.

Дискримінантний аналіз є параметричним статистичним методом, який формує рішення на основі лінійної або квадратичної комбінації ознак. Кожна дискримінантна функція являє собою скалярний добуток вектора вхідних даних і вектора коефіцієнтів, що доповнюється пороговим значенням. Модель ДА має фіксовану, регулярну структуру, де кількість параметрів дорівнює кількості ознак, а обчислювальні операції обмежуються множенням та додаванням. Це забезпечує високу швидкодію, детермінованість виконання та передбачувану складність алгоритму.

Нейронні мережі, натомість, складаються з одного або кількох прихованих шарів, кожен із яких виконує лінійне перетворення даних із подальшою нелінійною активацією. Для досягнення високої точності класифікації в задачах середньої та високої складності нейронні мережі потребують сотень або тисяч параметрів, а в глибоких архітектурах їхня

кількість може сягати мільйонів. Це робить модель значно більш гнучкою, але одночасно й значно вимогливішою до апаратних та обчислювальних ресурсів.

Однією з ключових відмінностей є особливості навчання моделей. У дискримінантному аналізі параметри визначаються аналітично на основі статистичних характеристик — середніх значень та матриць коваріації. Навчання є одностадійним і не потребує ітеративної оптимізації. Нейронні мережі, навпаки, використовують метод зворотного поширення похибки та чисельні оптимізатори (SGD, Adam), що вимагає багаторазового проходження даних та тривалої обчислювальної процедури. Це ускладнює вбудовування нейронної мережі в системи, де навчання модельних параметрів має бути здійснене апаратно або в реальному часі.

З погляду стійкості до шуму, нейронні мережі демонструють високу здатність до узагальнення за умови достатнього обсягу навчальних даних, проте вони є вразливими до надмірного навчання, особливо при роботі з малими вибірками. Дискримінантний аналіз у таких випадках показує стабільнішу поведінку, оскільки базується на статистичному описі класів, а не на оптимізації великої кількості параметрів.

Нейронні мережі (MLP, CNN) забезпечують високу точність, але:

- вимагають складних нелінійних обчислень;
- потребують великих ресурсів пам’яті;
- мають багатошарову структуру, що суттєво ускладнює апаратну реалізацію;
- потребують FPGA/ASIC великого масштабу і енергоспоживання.

У задачах швидкого розпізнавання параметричних даних (наприклад, класифікація за 8–12 ознаками) використання нейронних мереж виправдане лише частково.

Дискримінантний аналізі переважає тим, що:

- має однорівневу структуру обчислення;
- не потребує багатошарової нелінійності;
- не залежить від великого набору ваг;
- може бути реалізований у вигляді лінійної або систолічної обчислювальної структури з регулярними зв’язками.

Найбільш суттєві відмінності виявляються під час розгляду можливості апаратної реалізації. Нейронні мережі вимагають наявності блоків множення, додавання, пам’яті для ваг, модулів активації, а також механізмів паралельного обчислення всередині шару. Реалізація навіть невеликої нейронної мережі на ПЛІС або ASIC потребує значної кількості логічних елементів та вбудованих апаратних множників. Крім того, різні шари мають різну розмірність, що утворює нерегулярну структуру, складну для маршрутизації.

Дискримінантний аналіз має суттєві переваги. Він не використовує нелінійні функції, не потребує багатошарової структури, має мінімальну кількість параметрів і може бути реалізований як регулярний систолічний масив. Кожна дискримінантна функція обчислюється незалежно, а фінальне рішення формується за допомогою апаратної WTA-конкуренції. Це робить ДА надзвичайно ефективним для високошвидкісних, енергоощадних та компактних апаратних класифікаторів.

1.4.4 Переваги дискримінантного аналізу

Лінійність обчислень ЛДФ базується на лінійних операціях, що легко реалізуються:

- у ПЛІС,
- у систолічних масивах,

- у нейроподібних структурах (WTA, суматори, інкремент/декремент),
- у асоціативних процесорах.

Це дозволяє створити апаратний класифікатор з мінімальними витратами логіки.

Регулярність структури обчислень ЛДФ має повторюваний шаблон операцій:

для кожного класу така регулярність:

- легко масштабується,
- дозволяє створити обчислювальний масив з однаковими комірками,
- мінімізує маршрутизацію сигналів у ПЛІС,
- дає можливість повністю паралелізувати обчислення.

Мінімальна кількість параметрів, на відміну від нейромереж:

- немає тисячів ваг,
- лише n+1 коефіцієнт для кожної ЛДФ.

У системах реального часу це критично:

- менше пам’яті, вищий коефіцієнт надійності;
- менше комутацій, менше енергії;
- простіша верифікація апаратного модуля.

Висока швидкість та детермінованість, обчислення ЛДФ має постійну часову складність O(1):

- незалежно від розміру навчальної вибірки;
- незалежно від складності структури класів.

Природне поєднання з WTA-конкуренцією, після обчислення ЛДФ здійснюється конкуренція типу “1 з N”:

- вибір максимуму,

- ранжування результатів,
- швидке прийняття рішення.

WTA (Winner-Takes-All) легко реалізується за допомогою:

- інкрементних/декрементних схем,
- паралельних мінімум/максимум-компараторів,
- асоціативних осередків.

Це ідеально інтегрується в твій дисертаційний підхід із нейроподібними класифікаторами.

1.5 Висновки до першого розділу

1. Сформовані в процесі різницево-зрізового оброблення n-елементного початкового масиву чисел $\boldsymbol{a}_0$ дві матриці бінарних масок G і F дозволяють реалізувати сортування і ранжування елементів масиву за зростанням їх числових значень, що свідчить про багатофункціональність такого методу оброблення.
2. Матриця бінарних масок G дає можливість сформувати відсортований масив $\boldsymbol{a}_0^s$, а матриця бінарних масок F дає можливість сформувати вектор рангів $\boldsymbol{r}$ елементів масиву $\boldsymbol{a}_0$, а також відновити початковий масив $\boldsymbol{a}_0$, враховуючи сформований вектор внутрішніх порогів оброблення q в процесі відповідного векторно-матричного множення, що свідчить про двовимірність процесу різницево-зрізового оброблення.
3. Порівняння математичного опису як процесу асоціативної вибірки даних у відповідній розподіленій пам’яті за змістом, так і процесу розпізнавання у нейромережах класичного типу свідчить про їх аналогічність, що підтверджує наявність властивості асоціативності у нейромережах.

4. Наявність асоціативних рівнів оброблення даних у засобах інтелектуальних систем підтверджує ефективність задіяння нейромережних технологій на таких прикладах, як керування динамічними системами у складних умовах, а також розпізнавання об’єктів з ранжуванням результатів класифікації.

РОЗДІЛ 2
ПАРАЛЕЛЬНЕ ОБРОБЛЕННЯ МАСИВУ ДАНИХ ДЛЯ АСОЦІАТИВНО-ЛОГІЧНИХ ОПЕРАЦІЙ

2.1 Сортування як базис асоціативного оброблення одновимірного масиву даних

Для наочноссті наведених математичних співвідношень (1.9), (1.10), (1.11) та (1.14) у табл.2.1 послідовно по циклах наведено приклад різницево-зрізового оброблення елементів масиву $\boldsymbol{a}_0 = \{11,3,5,8,15\}$ [53]. Показано визначення внутрішніх порогів q_j, відсортованих елементів a_i^s масиву $\boldsymbol{a}_0^s$, а також елементів $g_{i,j}$ (1.9) та $f_{i,j}$ (1.10) відповідно матриць бінарних масок **G** і **F**. Знаком «-» у табл.2.1 позначено від'ємні елементи $a_{i,j}$.

Для наведених в табл.2.1 даних масиву $\boldsymbol{a}_0$ і матриці бінарних масок **G** можна перевірити слушність формули (1.11) векторно-матричного множення, а саме:

$$\boldsymbol{a}_0^s = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 11 \\ 3 \\ 5 \\ 8 \\ 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 8 \\ 11 \\ 15 \end{bmatrix}. \qquad (2.1)$$

Разом з тим, як приклад нижче наводиться результат відновлення за формулою (1.14) елементів масиву $\boldsymbol{a}_0$ з використанням сформованої матриці бінарних масок **F** і вектора внутрішніх порогів **q** (табл.2.1) у вигляді векторно-матричне множення:

$$\boldsymbol{a}_0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \\ 5 \\ 8 \\ 15 \end{bmatrix}. \quad (2.2)$$

Крім того, для матриці бінарних масок **F** у табл.2.1 введено додаткові рядки, де поступово підраховується кількіть одиничних елементів $f_{i,j}$ у кожному рядку цієї матриці. Отримані значення відповідають рангу r_i кожного з елементів $a_{i,0}$ масиву $\boldsymbol{a}_0$ і представляють зріз (вектор) рангів $\boldsymbol{r}$ [53] з елементами r_i вигляду:

$$r_i = \sum_{j=1}^{n} f_{i,j}. \quad (2.3)$$

Цей факт можна пояснити тим, що кожний одиничний елемент $f_{i,j}$ у рядку сформованої матриці бінарних масок **F** означає входження відповідних мінімальних складових – внутрішніх порогів оброблення q_j у початкове числове значення відповідного елемента $a_{i,j}$. В результаті кількість одиничних елементів $f_{i,j}$ фіксує співвідношення між елементами $a_{i,0}$ масиву $\boldsymbol{a}_0$ з урахуванням розкладання їх числових значень на мінімальні складові q_j (1.6).

Отже, одиничні елементи $g_{i,j}$ матриці бінарних масок **G** свідчать про топологічне розташування елементів $a_{i,0}$ масиву $\boldsymbol{a}_0$ за зростанням їх числових значень, а сума одиничних елементів $f_{i,j}$ у і-му рядку матриці бінарних масок **F** (1.10) формує ранг r_i відповідного елемента $a_{i,0}$ у початковому масиві $\boldsymbol{a_0}$. Крім того, матриця бінарних масок **F** та вектор внутрішніх порогів **q** (1.13) забезпечують відновлення елементів масиву $\boldsymbol{a}_0$ згідно з виразом (1.14).

Таблиця 2.1

Приклад процесу різницево – зрізового оброблення числового масиву

Елементи $a_{i,j}$ різницевих зрізів $\boldsymbol{a}_j$	Різницеві зрізи $\boldsymbol{a}_j$					
	$\boldsymbol{a}_0$	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	$\boldsymbol{a}_3$	$\boldsymbol{a}_4$	$\boldsymbol{a}_5$
$a_{1,j}$	11	8	6	3	0	-
$a_{2,j}$	3	0	-	-	-	-
$a_{3,j}$	5	2	0	-	-	-
$a_{4,j}$	8	5	3	0	-	-
$a_{5,j}$	15	12	10	7	4	0
Цикли оброблення		1	2	3	4	5
Внутрішні пороги q_j		3	2	3	3	4
Відсортовані елементи a_i^s		3	3 + 2 = 5	5 + 3 = 8	8 + 3 = 11	11 + 4 = 15
Елементи $g_{i,j}$ зрізів $\boldsymbol{g}_j$		Зрізи $\boldsymbol{g}_j$ матриці **G**				
		g_1	g_2	g_3	g_4	g_5
$g_{1,j}$		0	0	0	1	0
$g_{2,j}$		1	0	0	0	0
$g_{3,j}$		0	1	0	0	0
$g_{4,j}$		0	0	1	0	0
$g_{5,j}$		0	0	0	0	1
Елементи $f_{i,j}$, зрізів $\boldsymbol{f}_j$ і ранги r_i		Зрізи $\boldsymbol{f}_j$ матриці **F**				
		$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$	$\boldsymbol{f}_4$	$\boldsymbol{f}_5$
$f_{1,j}$		1	1	1	1	0
r_1		1	1 + 1 = 2	2 + 1 = 3	3 + 1 = 4	**4**
$f_{2,j}$		1	0	0	0	0
r_2		1	1	1	1	**1**
$f_{3,j}$		1	1	0	0	0
r_3		1	1 + 1 = 2	2	2	**2**
$f_{4,j}$		1	1	1	0	0
r_4		1	1 + 1 = 2	2 + 1 = 3	3	**3**
$f_{5,j}$		1	1	1	1	1
r_5		1	1 + 1 = 2	2 + 1 = 3	3 + 1 = 4	4 + 1 = **5**

В процесі різницево-зрізового оброблення векторного масиву чисел використовується не тільки паралельне багатооперандного підсумування його елементів, але й існує можливість їх відновлення після повного обнуління масиву, а також виконання таких асоціативних операцій, як сортування і ранжування елементів масиву[36].

2.1.1 Графічна модель позрізового оброблення для асоціативно-логічних операцій

Для наочності на рис. 2.1 показано сформовані поточні групи чисел $\boldsymbol{a}_j$, поточний мінімальний елемент q_j у кожному j-му циклі, а також відсортовану послідовність елементів $a_1^s, \dots, a_5^s$. Групи чисел на рис. 2.1 показано у вигляді вектор-стовпців через збільшення ілюстративності процесу сортування по циклах. Кількість робочих циклів складає 5, тобто $N{=}n$, оскільки початкова група чисел не містить жоден елемент, що дорівнює нулю. Шостий цикл не є робочим, він використаний для визначення умови завершення процесу сортування (q_6 = 0).

Отже, базові операції (1.6) і (1.7) дозволяють визначити РЗ як масив чисел вигляду (1.7), кожний елемент якого формується як різниця однойменного елемента поточного масиву і конкретної величини – порогу обробки q_j, який є величиною постійною у межах поточного циклу обробки і дорівнює найменшому ненульовому елементу попереднього масиву чисел (1.6). Причому елементи q_j приймають участь не тільки у формуванні поточних РЗ, але й у визначенні поступово всіх елементів відсортованого за їх зростанням масиву a_i^s (1.12).

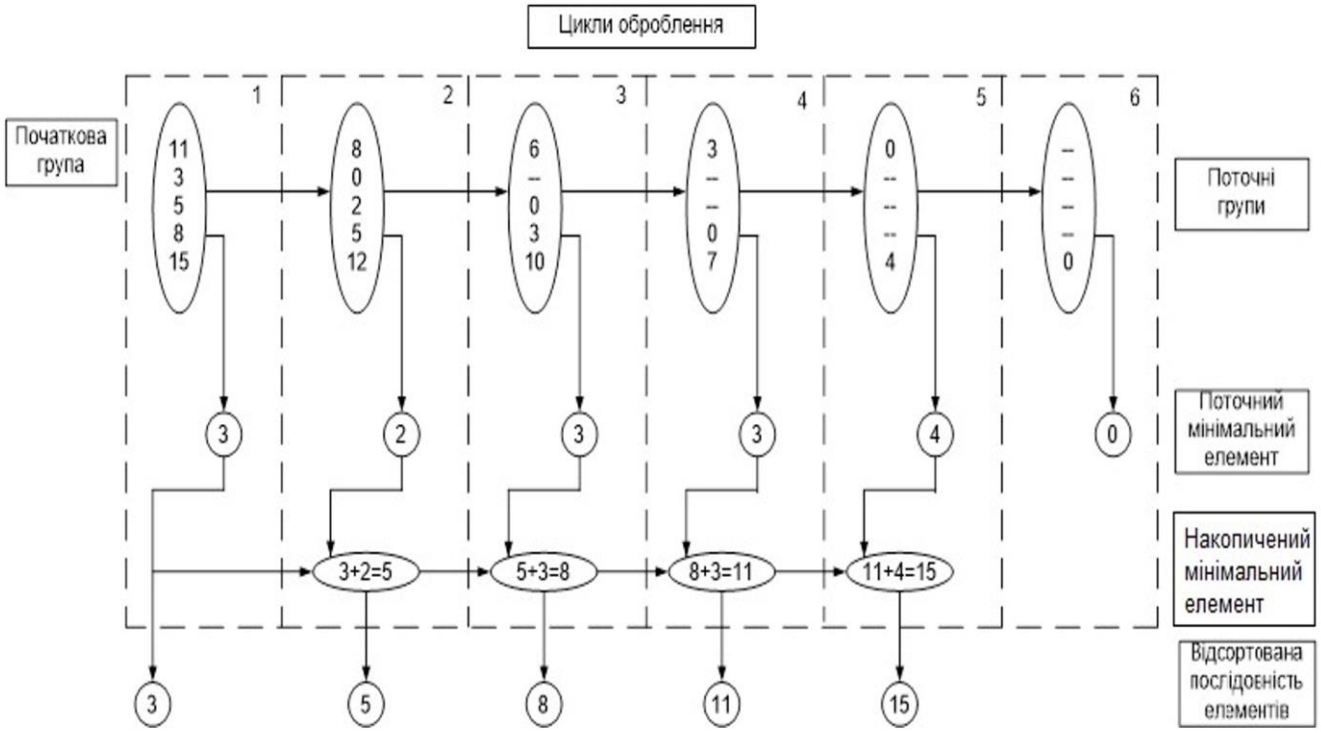


Рисунок 2.1 - Приклад сортування елементів векторного масиву чисел

Відомо, що будь-яка зв’язна структура даних може бути інтерпретована у вигляді різновида абстракції – графа [54,55]. Це стосується, в першу чергу, пошукових систем під час виявлення місцерозташування інформації, наприклад, у Web [54].

Разом з тим, навіть найпростіші алгоритми на графах мають достатню наочність особливо з метою їх подальшого комп’ютерного моделювання. Так, відомо базові властивості графа для алгоритму вирішення задачі топологічного сортування (topologicalsorting) [54]. Як приклад можна також навести графо-структурні моделі, використані у монографії [56] для ілюстрації та дослідження процесів цифрової обробки сигналів.

Отже, для сортування числового масиву за методом РЗ інформаційний граф (ІГ) цього процесу представлено на рис. 2.2 [57,58].

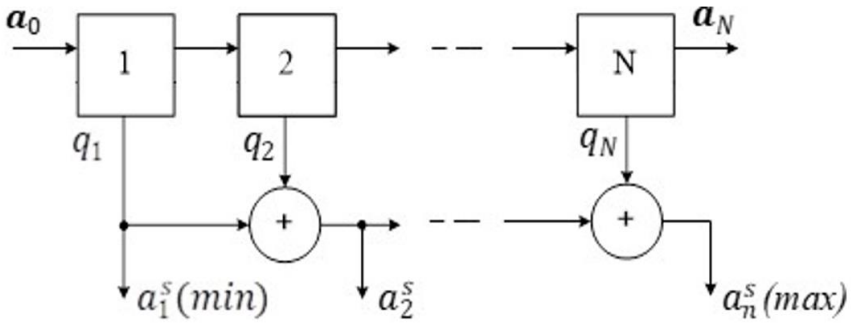


Рисунок 2.2 - Інформаційний граф процесу сортування чисел за методом РЗ

На рис. 2.2 базовий цикл обробки за РЗ, що містить операції (1.6) і (1.7), показано у вигляді квадратів з фіксацією номера циклу. А формування елементів a_i^s відсортованого масиву a^s показано функціонально як результат підсумовування відповідних елементів q_j.

Оскільки у загальному випадку попередньо обумовлюється, що всі елементи a_{i0} початкового масиву $\boldsymbol{a_0}$ не є нульовим, то максимальна кількість робочих циклів N дорівнює розмірності n початкового масиву [34]. Тому на рис. 2.2 показано, що у N-му циклі формується останній найбільший елемент a_i^s відсортованого масиву a_i^s. Крім того, побічним результатом процесу сортування є можливість визначення екстремальних (максимального і мінімального) елементів початкового масиву $\boldsymbol{a_0}$.

У цьому випадку з рис. 2.2 видно, що відповідний ІГ сортування свідчить про простоту і поступовість процесу сортування, тобто без зворотних зв'язків, а також передбачає спрощену структуру обчислювальних засобів для його реалізації.

Подальший аналіз вузлів ІГ процесу сортування (рис. 2.2) дає можливість розглядати варіанти їх апаратної побудови. Так, одним із перспективних

варіантів реалізації базових операцій (1.6) і (1.7) є їх суміщення, тобто одночасне формування як поточного РЗ $\boldsymbol{a}_j$, так і поточного порогу обробки q_j.

Ще одна графічна модель, що представлена на рис 2.3 схематично показує процес різницевих зрізів $\boldsymbol{a}_0, \boldsymbol{a}_1, \dots, \boldsymbol{a}_n$, матриці бінарних масок **F,** вектора внутрішніх порогів $\boldsymbol{q} = \{ q_1, \dots, q_n, \}$ та вектора рангів $\boldsymbol{r} = \{ r_1, \dots, r_n, \}$[53]. Для прикладу на рис 2.3 за замовчуванням прийнято, що всі елементи a_{i0} початкового вектора $\boldsymbol{a}_0$ є різними за числовим значенням і не є нульовими. В цьому випадку кількість циклів *N* залежить від *O(n)* [34].

Отже саме використання сформованих в процесі оброблення за різницевими зрізами елементів q_j вектора внутрішніх порогів $\boldsymbol{q}$ та числових даних рядків матриці бінарних масок **F** дозволяє реалізувати такі асоціативні операції, як сортування елементів початкового вектора $\boldsymbol{a}_0$ та визначення їх рангів за умови зростання числових значень.

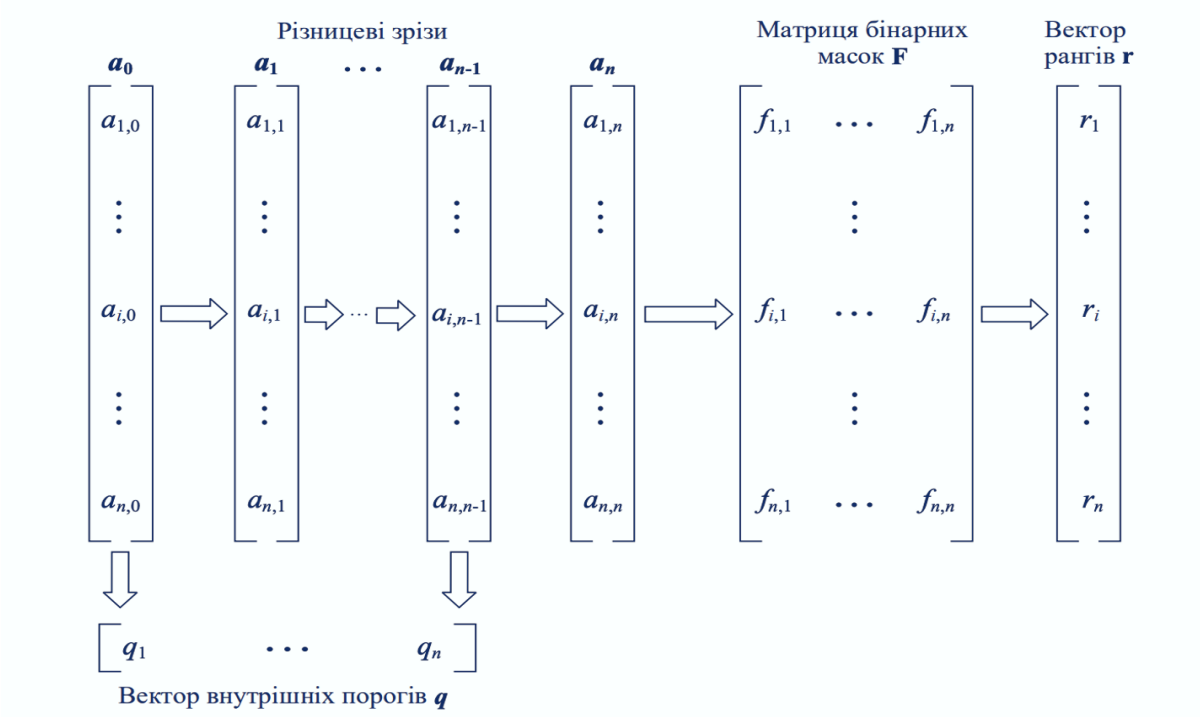


Рисунок 2.3 – Схематичне відображення процесу обробки за різницевими зрізами

2.2 Особливості обчислювального процесу при класифікації об’єктів

Після аналізу базових співвідношень (1.4) і (1.5) процесу класифікації об’єктів за лінійними дискримінантними функціями (ЛДФ) в даній роботі пропонується два альтернативних варіанти їх реалізації (рис 2.4)

Отже, перший спосіб визначення максимального значення ЛДФ вигляду (1.4) формуються тільки зважені вихідні сигнали:

$$a_{ij} = w_{ij} * x_j, \qquad j = 1, \dots, n, i = 1, \dots, m \tag{2.4}$$

що представляють собою елементи матриці **A** вигляду:

$$A = \begin{bmatrix} a_{11} \dots a_{1n} \\ a_{m1} \dots a_{mn} \end{bmatrix}. \tag{2.5}$$

Рисунок 2.4 – Реалізаційні моделі процесу класифікації об’єктів

Далі застосовується відомий метод оброблення елементів a_{ij} матриці $\boldsymbol{A}$ за методом різницевих зрізів (РЗ) [34,60]. Цей спосіб оброблення матричних даних має ознаки систолічного оброблення [60], що передбачає його паралельну реалізаію на систолічному обчислювачі [61,62].

Другий спосіб передбачає обчислення значень всіх ЛДФ за формулою (1.4). Але пропонується виконання співвідношення (1.5) як сортування з використанням операції декремента [20,63]. В результаті такий сортувальник апаратно можна реалізувати на реверсивних лічильниках [20].

2.2.1 Класифікація об'єктів з нормалізацією елементів ЛДФ

Для вдосконаленння за 1-м способом процесу класифікації об'єктів необхідно врахувати, по-перше, наявність відємного елемента $w_{io}x_o$ у кожній ЛДФ за формулою (1.4); а по-друге, залишити незмінним процес систолічного оброблення елементів ЛДФ у матричному вигляді [61,62]. Такий підхід зорієнтовоно на апаратну реалізацію систолічної матриці [61] на мікросхемі ПЛІС, закрема, на мікросхемі FLEX10K фірми ALTERA [64].

Для врахування від'ємного елемента $w_{io}x_o$ у формулі (1.4) необхідно виконати нормування відповідних доданків $w_{ij}x_j$. Це необхідно зробити до процесу матричного оброблення всіх елементів (2.4), які, у свою чергу, представлено як елементи відповідної матриці $\boldsymbol{A}$ розмірністю mxn вигляду (2.5).

Якщо представити вагові коефіцієнти w_{oi} для всіх m ЛДФ $g_i(X)$ (1.4) у вигляді вектора зміщень B, враховуючи, що x_o=1, то можна задіяти такий варіант їх подання для врахування у матриці $\boldsymbol{A}$ (2.5). Пропонується сформувати матрицю зміщень $\boldsymbol{B}$ розмірністю mxn з елементами вигляду:

$$b_{ij} = \frac{w_{io}}{m}, \tag{2.6}$$

де m- кількість класів.

В результаті матриця зміщень $\boldsymbol{B}$ матиме вигляд:

$$b_{ij} = \frac{w_{io}}{m}, \tag{2.7}$$

Отже, якщо елементи кожного рядка матриці $\boldsymbol{A}$ представляють собою доданки i-ої ЛДФ $g_i(X)$, то зменшення їх на однакову для кожної ЛДФ величину w_{io} дозволить врахувати від'ємний елемент у формулі (1.4). Такий підхід з використання співвідношення (2.6) можливий через те, що за замовчуванням приймається $x_0 = 1$.

Таким чином, остаточно нормалізована матриця елементів ЛДФ формується так [65]:

$$\boldsymbol{A^o} = \boldsymbol{A} - \boldsymbol{B}, \tag{2.8}$$

де $\boldsymbol{A^o}$ – матриця з нормованими елементами a_{ij}^o початкової матриці $\boldsymbol{A}$ (2.5).

В подальшому елементи матриці $\boldsymbol{A}$ (2.8) обробляються за методом РЗ, а саме, спочатку паралельно по стовпцях, а потом паралельно по рядках в кожному циклі оброблення [61].

У цьому випадку основними операціями, які здійснюються над елементами a_{ij}^0 вхідної матриці $\boldsymbol{A^o}$ вигляду (2.8) є такі три операції [61]:

- визначається метрика як міра подібності елементів кожного j-го стовпця (t–1)-ої поточної матриці $\boldsymbol{A^{t-1}}$, де $t = \overline{1,N}$:

$$q_j^t = min\, A_j^{t-1} = \; min\, a_{ij}^{t-1}, i = \overline{1,m} \tag{2.9}$$

тут a_{ij}^{t-1} –- елемент матриці $\boldsymbol{A^{t-1}}$; N - кількість ітерацій;

- виконується корекція елементів стовпців матриці $\boldsymbol{A^{t-1}}$ **:**

$$\overline{a_{ij}^t} = a_{ij}^{t-1} - q_j^t, \tag{2.10}$$

тобто формується матричний (двовимірний) різницевий зріз:

$$\overline{\boldsymbol{A^t}} = \{\overline{a_{ij}^t}\}, \tag{2.11}$$

- виконується впорядкування матриці $\overline{\boldsymbol{A^t}}$ (2.11) в процесі транспозиції (просування) праворуч до краю нульових елементів $\overline{a_i^t}$ у всіх рядках $\overline{A_i^t}$ формування впорядкованого двовимірного різницевого зрізу $\boldsymbol{A^t}$:

$$A_i^t = Tr\left(\overline{A_i^t}\right). \tag{2.12}$$

Перед кожною транспозицією (2.12) перевіряється умова наявності принаймні одного нульового рядка $\overline{A_i^t}$, тобто

$$\exists\, \overline{A_i^t} = 0, t = \overline{1, N} \tag{2.13}$$

а також умова обнуління всіх $\overline{A_i^t}$ матриці $\overline{\boldsymbol{A^t}}$:

$$\forall\, \overline{A_i^t} = 0. \tag{2.14}$$

Виконання умови (2.13) є ознакою оптимальної суми елементів *i*-го рядка A_i^0 вхідної матриці $\boldsymbol{A^0}$, а виконання умови (2.14) свідчить про завершення процесу оброблення. При цьому останній обнулений *l*-й рядок A_l^N відповідає максимальній сумі елементів *l*-го рядка A_l^0 вхідної матриці $\boldsymbol{A^0}$.

Отже, весь процес оброблення елементів вхідної матриці $\boldsymbol{A^0}$ виконується до повного обнуління її рядків A_i^0, а сигнали ознаки нуля елементів цих рядків в

подальшому використовуються для формування вихідних сигналів класифікатора об'єктів за вирішальним правилом (1.5)

Таким чином, у формулах (2.10) і (2.12) до всіх елементів a_{ij}^{t} рядків матриці $\boldsymbol{A^t}$ правомірно застосовано такі властивості суми доданків, як комутативність та асоціативність [60]. Отже, всі елементи рядків матриці $\boldsymbol{A^t}$ можна реверсивно зсувати в межах кожного рядка, реалізуючи властивість комутативності суми доданків.

Крім того, враховуючи визначення максимальної суми елементів кожного рядка матриці $\boldsymbol{A^t}$ не є важливою величина кожної ЛДФ, а тільки співвідношення типу “>” між ними. В результаті всі ЛДФ правомірно одночасно зменшувати на однакову величину, а отже, зменшувати всі одноіменні елементи кожного стовпця матриці $\boldsymbol{A^t}$ на однакову величину що дорівнює значенню найменшого з них. Таким чином реалізується власивість асоціативності суми доданків, що непротирічять такому співвідношенню двох величин:

$$A > B = A - C > B - C. \tag{2.15}$$

Отже, видалення відповідного мінімального ненульового елемента паралельно у всіх стовпцях матриці $\boldsymbol{A}$ дозволяє виконати порівняння всіх m ЛДФ $g_{\text{i}}(\text{X})$, оскільки порівнюються їх одноіменні елементи a_{ij}, що в результаті забезпечує виділення серед всіх ЛДФ максимальної. А операція транспозиції (просування) праворуч або ліворуч сформованих нульових елементів a_{ij} у i-му циклі у всіх рядках матриці $\boldsymbol{A^t}$, $t = 0, .., N$ паралельно виконується з урахуванням властивості комутативності для сум елементів. Такі дії дозволяють уникнути в подальшому впливу сформованих нульових елементів у кожному стовпці матриці $\boldsymbol{A^t}$.

Отже, в процесі виконання класичного методу класифікації на базі ЛДФ необхідно спочатку розрахувати сумарне значення кожної з $ЛДФ_1, \dots, ЛДФ_m$ за виразом (1.4), а потім визначити серед них максимальну за величиною в процесі попарного порівняння і сформувати ознаку цього результату, а саме одиничне значення відповідного елемента вихідного вектора Y [6, 19].

У запропонованій моделі класифікації застосовується одночасне оброблення за Р3 у всіх стовпцях матриць $\boldsymbol{A^t}$, починаючи з матриці $\boldsymbol{A^0}$ (2.8) [5]. В результаті такий підхід дозволяє не “вирощувати” m сум вигляду $ЛДФ_1, \dots, ЛДФ_m$, а розпочати їх порівняння в процесі саме оброблення відповідних елементів по стовпцях [20].

Крім того, в процесі оброблення елементів матриць $\boldsymbol{A^0}$ і $\boldsymbol{A^t}$, де $t = \overline{1, N}$, можна сформувати не тільки вихідний вектор Y за вирішальним правилом (1.5), але й сформувати вектор рангів R [5].

На рис. 2.5 наведено схематичну діаграму процесу класифікації з нормуванням елементів початкових ЛДФ. На діаграмі показано не тільки вигляд і розмірність задіяних масивів данних, але й виконуванні операції, а саме, множення вектора на матрицю, нормалізацію, віднімання матриць, оброблення елементів матриці $\boldsymbol{A^0}$, задіяння вектора ознак для формування як вектора класів Y так і вектора рангів R [60, 61,66].

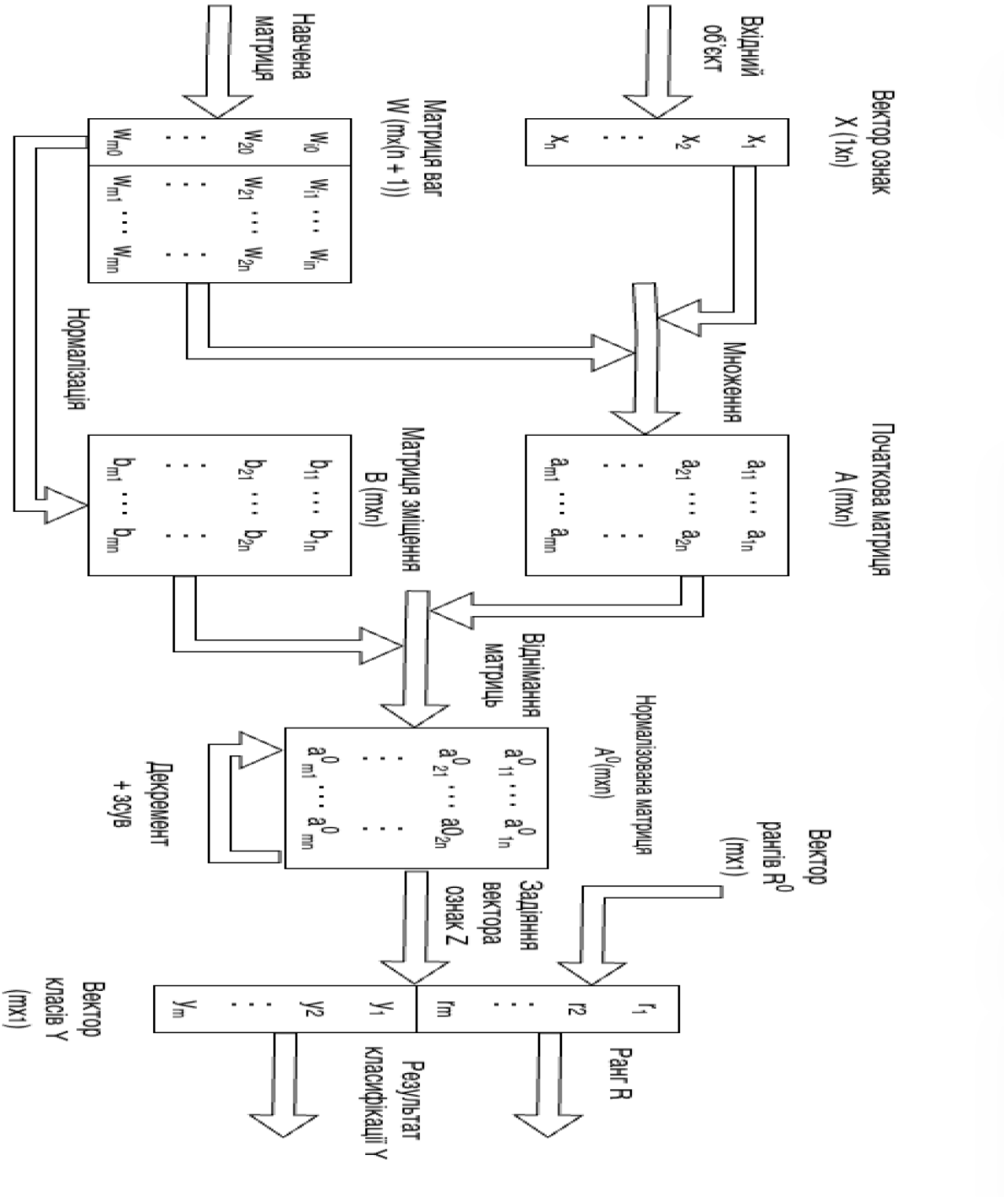


Рисунок 2.5 – Схематична діаграма процесу класифікації з нормуванням

2.2.2 Нейромережний підхід до класифікації об’єктів

З точки зору використання вирішального правила процесу класифікації об’єктів за ЛДФ вигляду (1.5) можливий альтернативний підхід стосовно реалізації операції “1 з N”, що фактично представляє собою механізм класифікації WTA (Winner Takes All) широко відомий з теорії нейромереж [4-6].

На рис. 2.6 показано два способи реалізації механізму конкуренції типу “1 з N”, акцент зроблено саме на мережні засоби пошуку числового максимуму.

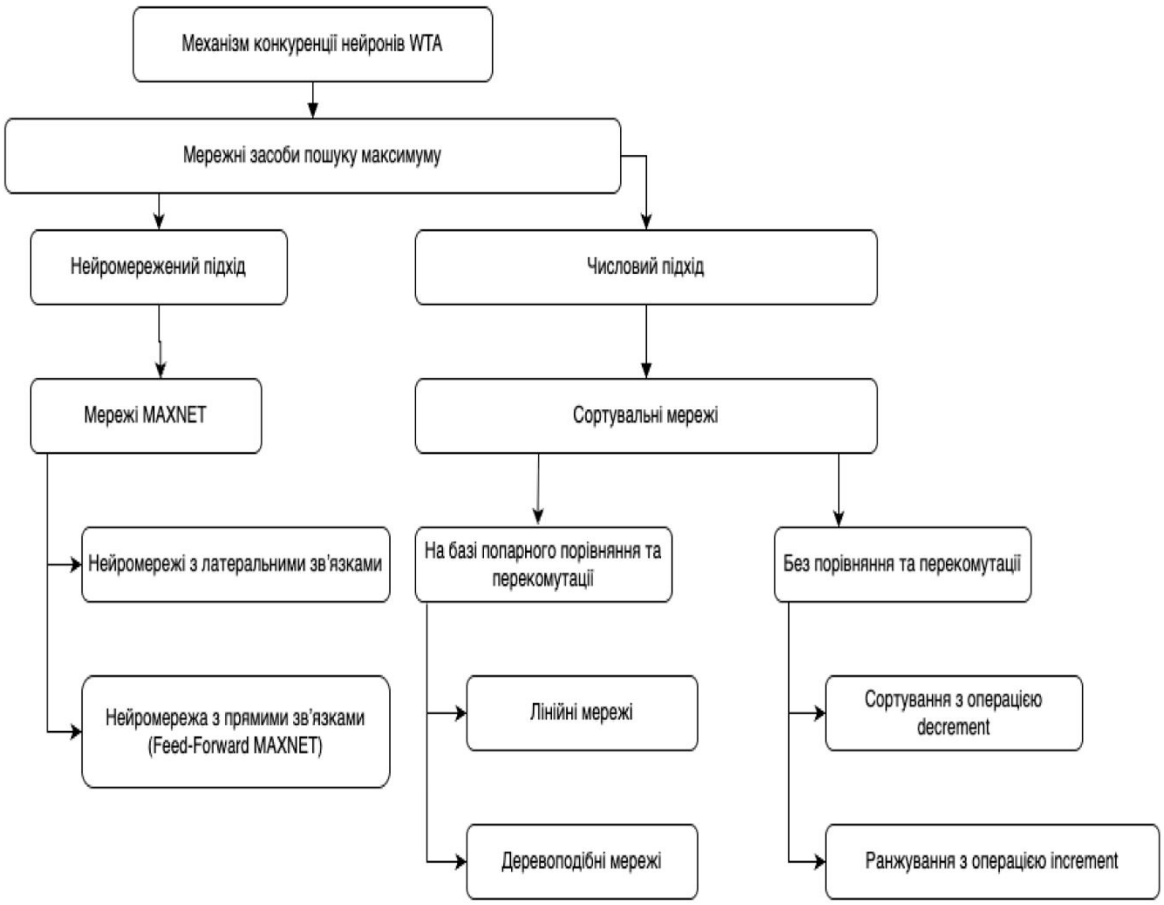


Рисунок 2.6 - Два способи реалізації механізму конкуренції типу “1 з N”

Окремо виділено два підходи: нейромережний та числовий. У нейромережному виконанні задіяно варіанти нейромереж типу MAXNET [4-6],

а числовий підхід в цьому випадку передбачає використання сортувальних мереж [1,2].

За умови наявності чи відсутності таких операцій, як порівняння чи перекомутація елементів числового масиву, інтерес з точки зору мінімізації апаратних та числових витрат передбачають останні [63]. Саме такі швидкісні операції, як декремент та інкремент дозволяють ефективно виконати відповідно сортування та ранжування елементів числового масиву [67].

У табл 2.2 наведено приклад процесу сортування з ранжуванням одновимірного числового масиву {1,4,3,5}. Оброблення масиву чисел виконується за 5 циклів, в результаті сформовано ранги всіх елементів за зростанням їх числових значень. Серед особливостей цього процесу необхідно виділити таке:

- у нульовому циклі всім елементам масиву присвоюється однаковий найменший ранг “1”;
- обнуління елементів масиву поступове, починаючи з найменшого числа за операцією декремента;
- момент обнулення конкретного елемента масиву визначає його відповідний ранг, ранг всіх інших ненульових елементів збільшується на одиницю за рахунок операції інкремента.

Аналіз процесу сортування з ранжуванням у табл. 2.2 показав, що сортування у числовому масиві виконується за за вдосконалиним методом оброблення за РЗ [67]. Особливості вдосконалення полягає в об’єднанні в єдиному циклі двох базових операцій: визначення ненульового внутрішнього порогу q_j^t (2.9) та корекції елементів масиву $\overline{a_{iJ}^t}$ (2.10), що відповідає формуванню поточного РЗ.

Такий підхід обумовлений тим, що для сортування важливим є процес упорядкуваня елементів масиву, починаючи з найменшого за значенням, що

можна визначити за умови поступового обнулення відповідних елементів числового масиву з врахуванням внутрішнього порогу оброблення. Момент обнулення також дозволяє сформувати відповідні ранги елементів.

Таблиця 2.2

Приклад процесу сортування з ранжуванням числового масиву

0-й цикл		1-й цикл		2-й цикл	
Числа	Ранги	Числа	Ранги	Числа	Ранги
1	1	**[0]**	1	0	1
4	1	3	2	2	2
3	1	2	2	1	2
5	1	4	2	3	2
3-й цикл		4-й цикл		5-й цикл	
Числа	Ранги	Числа	Ранги	Числа	Ранги
0	1	0	1	0	1
1	3	**[0]**	3	0	3
[0]	2	0	2	0	2
2	3	1	4	**[0]**	4

Переваги використання альтернативного методу сортування як варіант реалізації механізму конкуренції типу WTA (Winner Takes All):

- прискорення операції «1 з N» за рахунок використання швидкісної операції декремента паралельно до всіх елементів масиву;
- розширення функціональних можливостей процесу класифікації об’єктів за рахунок використання швидкісної операції інкремента для ранжування результатів класифікації;

- одночасне задіяння швидкісних операцій декремента/інкремента для сортування/ранжування в процесі застосування механізму конкуренції;
- забезпечення регулярності структурної організації класифікатора об’єктів при його апаратній реалізації за рахунок застосування реверсивних лічильників для операцій інкремента/декремента.

Отже, реалізація механізму конкуренції типу WTA запропонованим способом сортування з ранжуванням дозволяє визнати такий перехід до процесу класифікації об’єктів як нейроподібний із задіянням процесів конкуренції (рис. 2.4).

2.3 Висновки до другого розділу

1. Обом методам класифікаці об’єктів притаманна багатофункціональність оброблення масиву числових значень ЛДФ за рахунок формування їх рангів, що дозволяє визначити не тільки екстремальні (максимальний/мінімальний) елементи числового масиву, а, наприклад, найближчі до максимального.
2. Базовою операцією оброблення одно - та двовимірних масивів чисел за РЗ є альтернативний метод сортування із задіянням швидкісної операції декремента одночасно до всіх елементів векторного масиву чисел, що забезпечує максимальний паралелізм оброблення $O(n)$.
3. Розглянутий алгоритм реалізує просторово-розподілену обробку двовимірного масиву елементів дискримінантних функцій на основі принципу різницевих зрізів. Такий підхід до обробки елементів матричного масиву чисел дозволяє використати швидкісну операцію декремента паралельно для всіх елементів у кожному стовпці матриці.

РОЗДІЛ 3
НЕЙРОПОДІБНІ КЛАСИФІКАТОРИ З РОЗШИРЕНИМИ ФУНКЦІОНАЛЬНИМИ МОЖЛИВОСТЯМИ

Аналізуючи з точки зору апаратної реалізації базову структуру класифікатора об’єктів (рис. 1.3) [68] можна провести аналогію використання його базових блоків у нейромережному контексті (рис. 3.1)

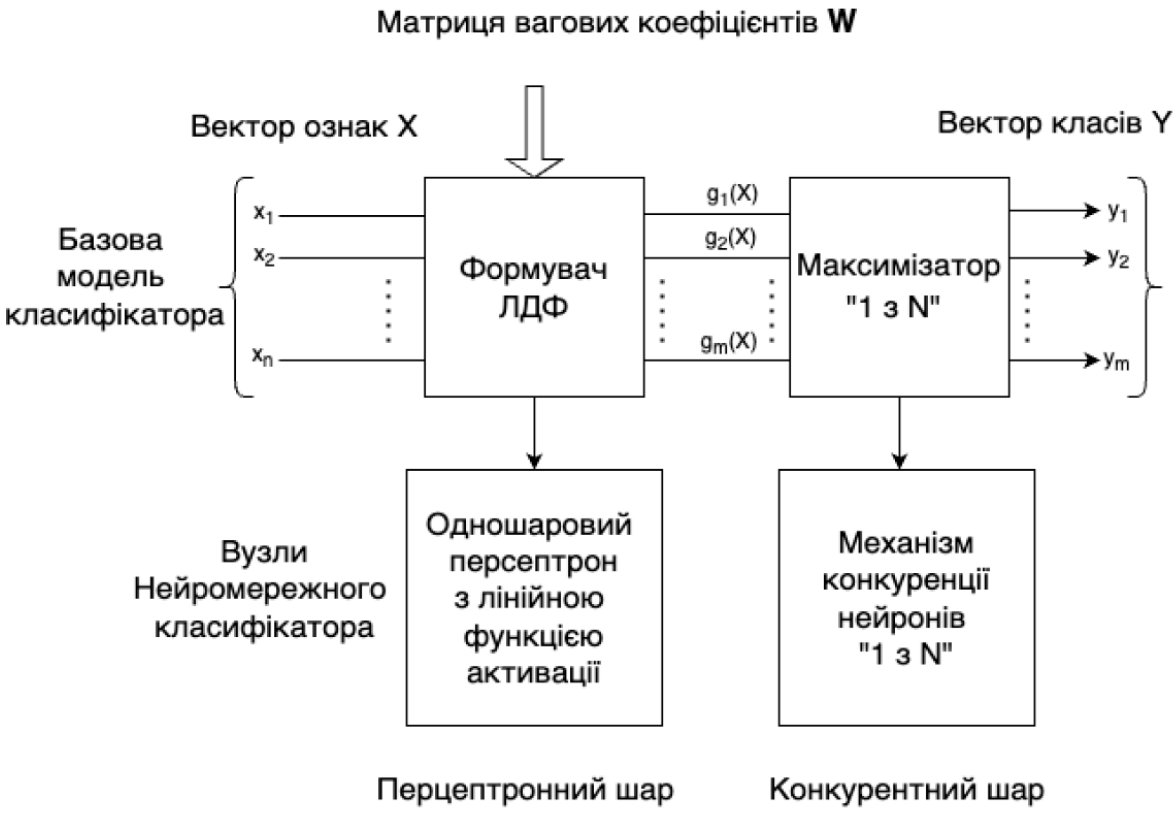


Рисунок 3.1 - Базова структура класифікатора об'єктів за дискримінантним аналізом

Так, функції формувача ЛДФ може виконати одношаровий персептрон з лінійною функцією активації, а функції максимізатора “1 з *N*” реалізуються механізмом конкуренції нейронів WTA (рис. 2.6) [4-6].

Такий структурний вигляд має відомий нейромережний класифікатор об’єктів (рис. 1.7) [13, 44], в якості прототипу якого було використано класичну нейромережу Хеммінга [4-6]. Конкретну структуру наведеного нейромережного класифікатора заявлено у патенті України [69].

У базовій структурі класифікатора об’єктів (рис.1.3) функціонально формувач у вигляді векторно-матричного помножувача обчислює *m* ЛДФ за формулою (1.4), для чого задіяно *mxn* помножувачів та *m* багатовхідних суматорів. Але саме максимізатор “1 з *N*” представляє інтерес для апаратної реалізації, оскільки тут можливі варіанти, два з яких розглядатимуться в даній роботі (рис.2.4).

Кожний варіант процесу класифікаціх об’єктів пов’язаний з тим, що, враховуючи необхідність обчислення вагових коефіцієнтів для ЛДФ за формулою (1.4), у базову структуру класифікатора об’єктів (рис. 1.3) обов’язково треба ввести начальний блок (рис. 3.2) [65].

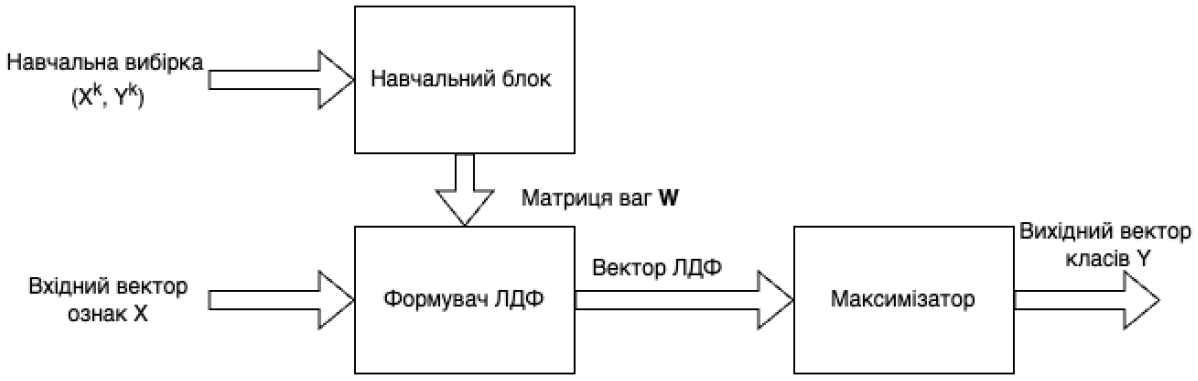


Рисунок 3.2 – Структура класифікатора об’єктів

Функціонально цей блок, використовуючи навчальні вибірки , де *K* – кількість виборок, формує матрицю вагових коефіцієнтів ***W,*** що дозволяє

формувачу ЛДФ обчислити набір відповідних ЛДФ. Для апаратної реалізації в подальшому інтерес представляють в залежності від обраного способу побудови класифікатора об’єктів (рис. 3.2) обидва його базових блока: формувач ЛДФ та максимізатор “1 з N”.

3.1 Нейроподібний класифікатор з позрізовим обробленням дискримінантних функцій

Для 1-го способу реалізації класифікаційного процесу з паралельним обробленням елементів ЛДФ за методом РЗ (рис. 2.4) на рис. 3.3 показано етапи синтезу обчислювальної структури класифікатора об’єктів. В результаті відображення регулярних ітераційних алгоритмів (РІА) за методом РЗ на систолічну структуру [62] побудовано багатофункціональний матричний обчислювач, який виконує функції як максимізатора, так і ранжувальника результатів класифікації об’єктів [61,64,67].

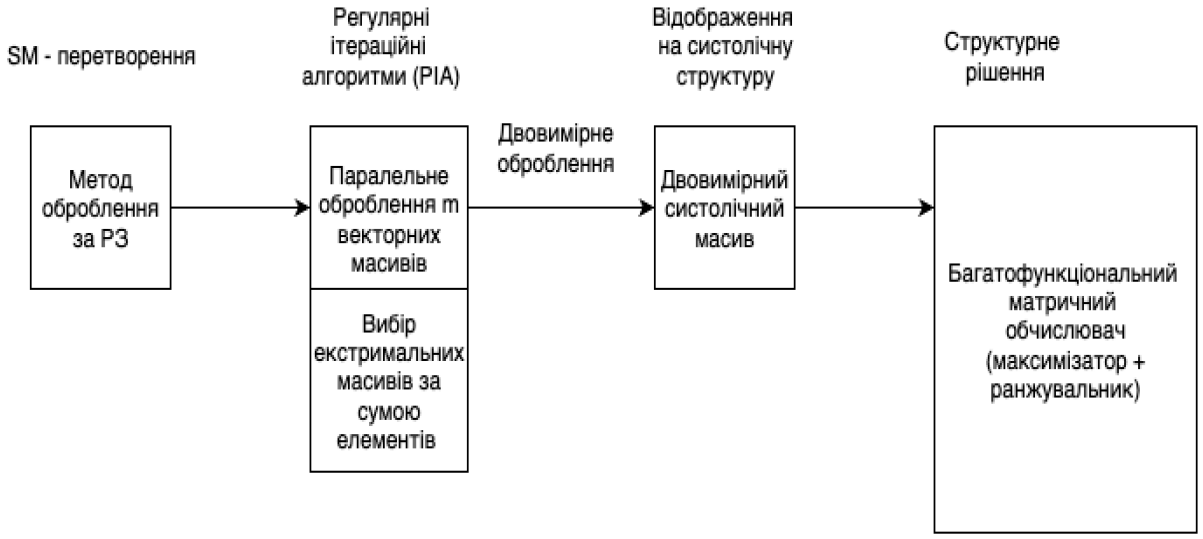


Рисунок 3.3 – Етапи синтезу обчислювальних структур для класифікації об’єктів

Разом з тим, враховуючи необхідність задіяння вільних елементів у співвідношенні (1.4) способом нормалізації елементів матриці ***A*** (2.5) на рис 3.4 показано удосконалену структуру класифікатора об'єктів [65].

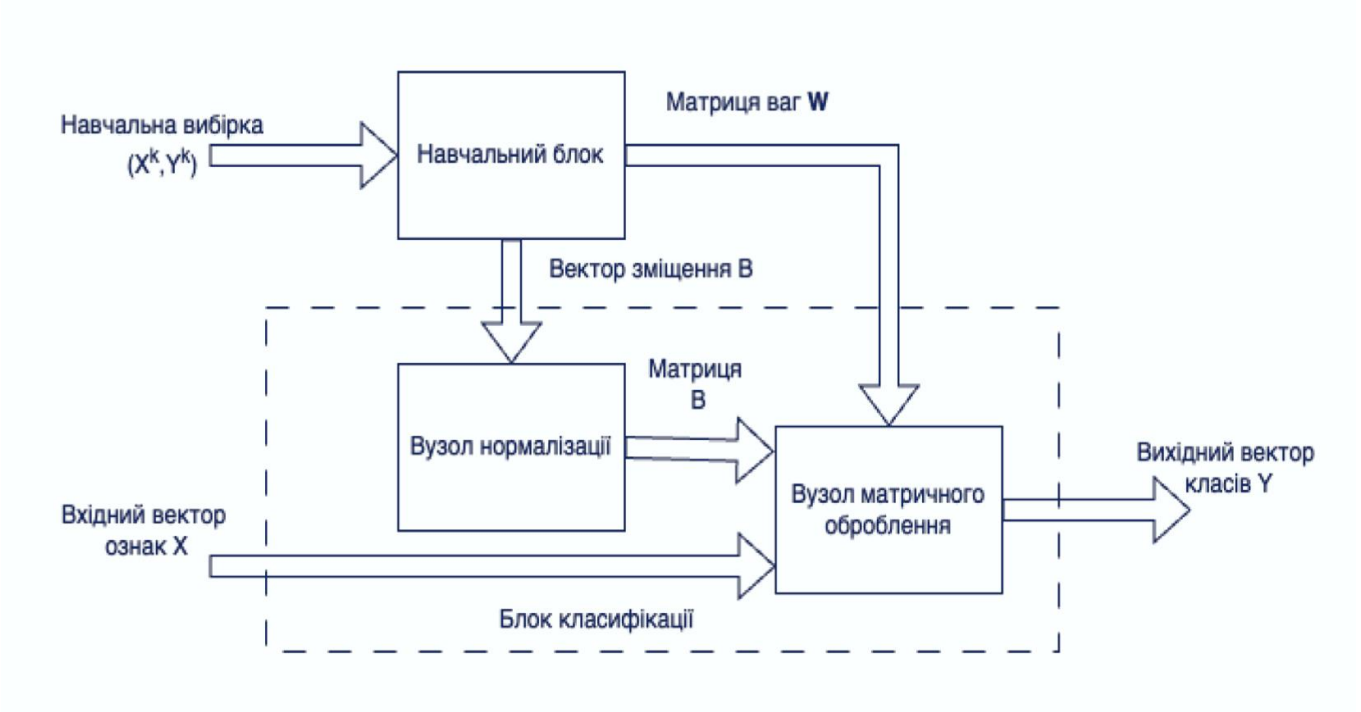


Рисунок 3.4 - Удосконалена структура класифікатора об'єктів з нормалізацією ЛДФ

Отже, у блок класифікації об'єднано вузол нормалізації, який перетворює вектор зміщення *B* з елементами (2.6) у матрицю зміщень ***B*** (2.7), та вузол матричного оброблення елементів матриці (2.8).

На рис 3.4 також показано, що на виході блока класифікації формується вихідний вектор класів *Y*. Разом з тим, функціонально, не змінюючи структури матричного обчислювача, у вузлі матричного оброблення можна забезпечити формування також вектора рангів *R* [66].

Отже, за базовими операціями (2.9), (2.10) та (2.12) процес систолічного оброблення елементів ЛДФ у вигляді матриці (2.8) для

конкретного прикладу має такий вигляд (рис. 3.5) [70]. При цьому нормалізована матриця має вигляд:

$$\boldsymbol{A}^0 = \begin{bmatrix} 25 & 18 & 12 & 8 \\ 20 & 9 & 6 & 20 \\ 10 & 21 & 30 & 4 \\ 15 & 6 & 424 & 28 \end{bmatrix} \tag{3.1}$$

На рис. 3.5 показано 10 циклів оброблення, де наведено: початкову матрицю (перший цикл) і впорядковану матрицю (починаючи з другого циклу).

Цю матрицю отримано в результаті видалення мінімального елемента (внутрішнього порогу) елементів кожного стовпця (2.9), результт транспозиції елементів у кожному рядку з просуванням всіх нульових елементів праворуч (2.12) та накопичені поточні суми.

В результаті сформовано вихідний вектор класів *Y*, в якому останній одиничний елемент вказує на відповідний рядок матриці з максимальною сумою його елементів, а також отримано суми кожного рядка матриці, що підтверджує слушність отриманого результату класифікації у вигляді вектора класів *Y*. Отже, вхідний об’єкт за його ознаками належить до 4-го класу із задіянням методу РЗ.

При цьому початкова матриця в процесі оброблення за РЗ обнуляється (крім останього рядка), а отже, поява ознаки нуля у кодному рядку матриці використовується для формування вектора *Y*. Необхідно відзначити, що операції найбільш трнивалі (2.9) та (2.10) прискорюються за рахунок застосування швидкісної операції декремента в кожному стовпці поточної матриці.

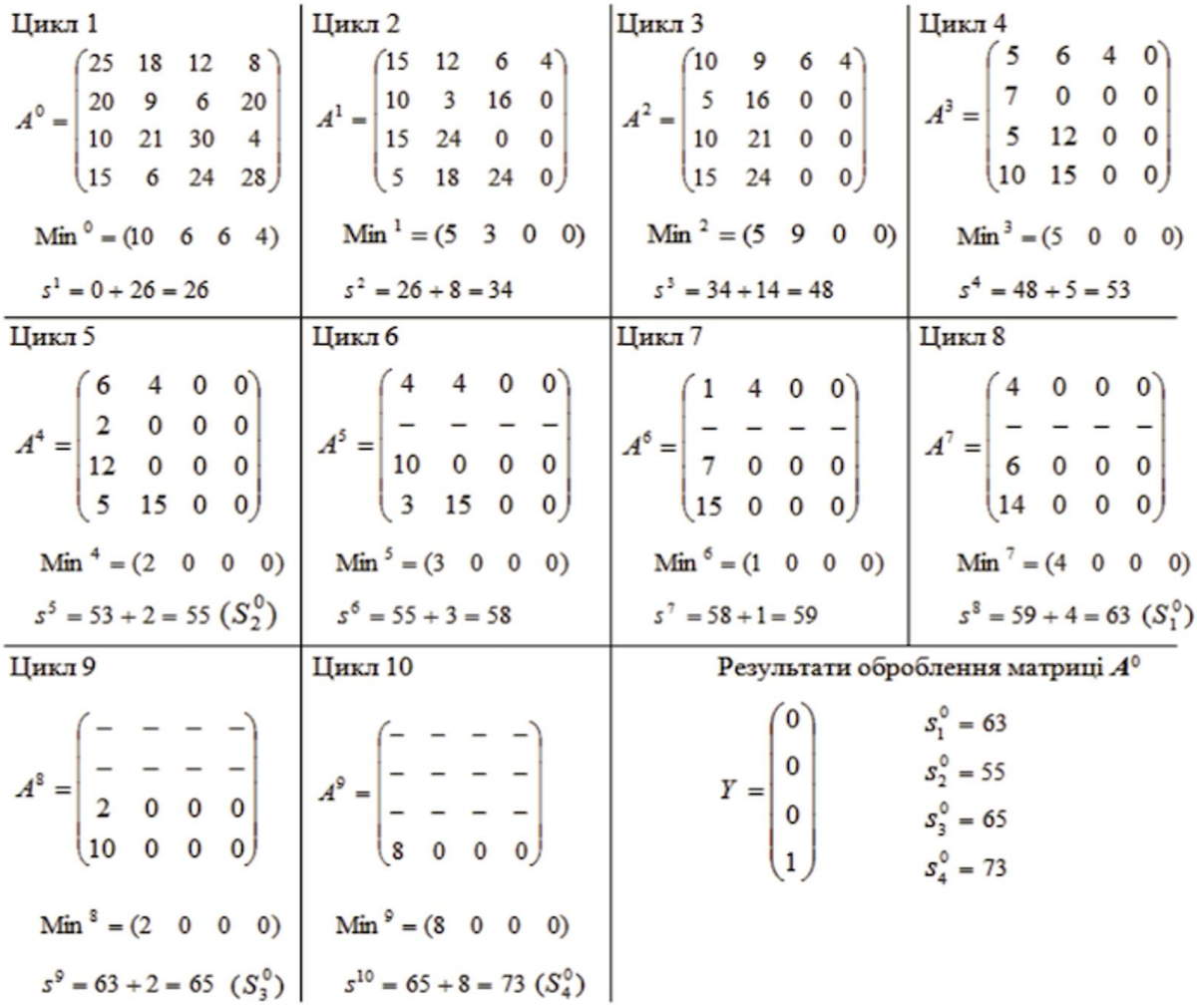


Рисунок 3.5 - Приклад оброблення двовимірної матриці $\boldsymbol{A}^0$ за РЗ по циклах

На рис. 3.6 [61] показано схематичне рішення частини вузла матричного оброблення блока класифікації (рис. 3.4), що фактично виконує функцію максимізатора (рис. 3.2). Тут показано сам матричний обчислювач, формувач вихідних сигналів та вузли синхронізації та аналізу реакцій, що формують відповідні сигнали обнуління (рис. 3.6).

На рис. 3.7 наведено функціональну схему *ij*-го ПЕ матричного обчислювача [61]. Основну роль в ПЕ відіграє лічильник СТ, який призначений для зберігання даних та виконання операції декремента над його вмістом. У лічильника СТ інформаційний вихід задіяно для передачі даних, а вихід сигналу

нуля - для керування процесом оброблення (рис. 3.6), а також як результуючий сигнал. На інформаційний вхід лічильника СТ надходять дані, на вхід скидання – сигнал Reset, на вхід лічби на спадання - сигнал Decrement .

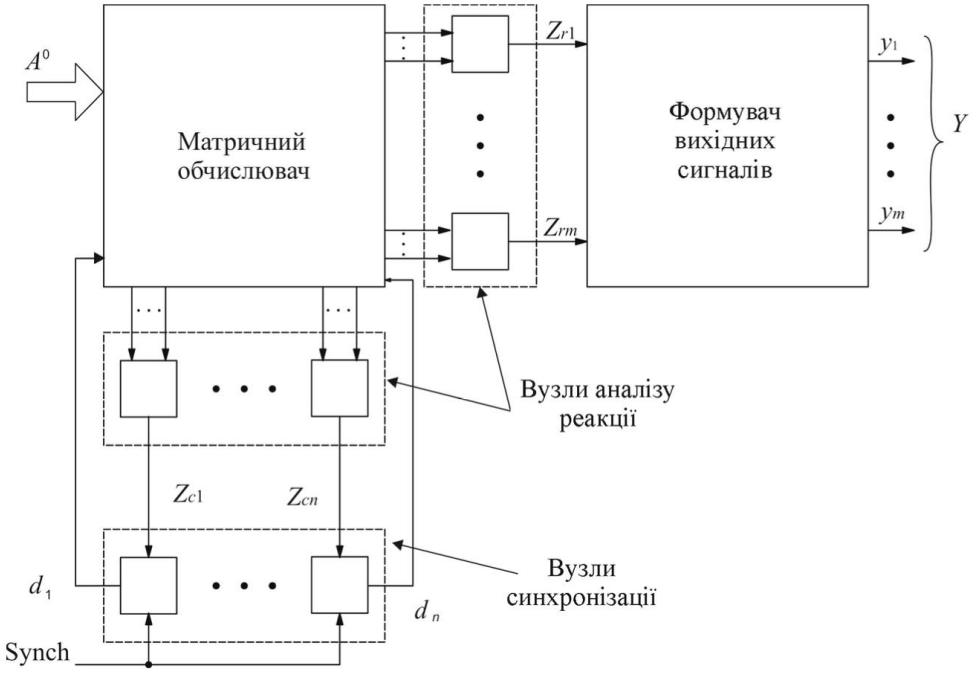


Рисунок 3.6 – Структурна схема вузлів матричного оброблення

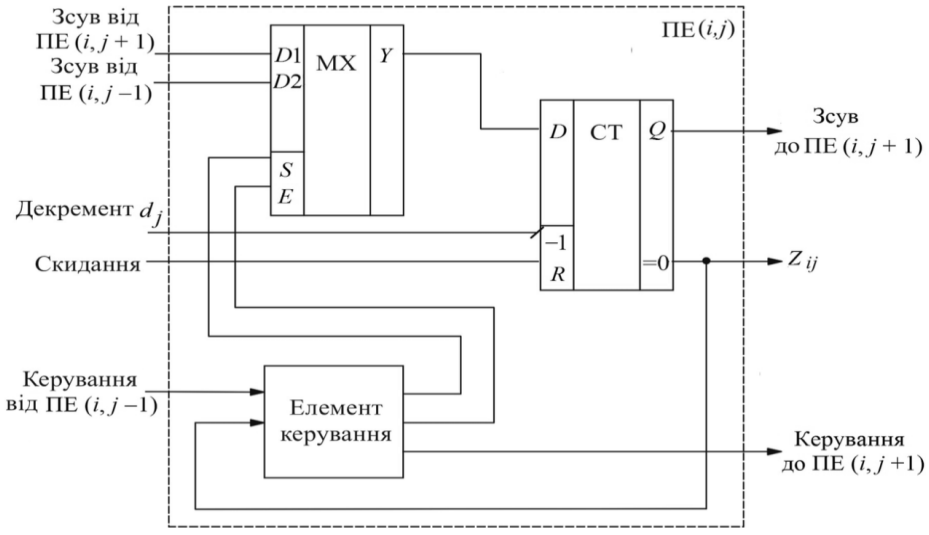


Рисунок 3.7 – Функціональна схума ПЕ матричного обчислювача

Мультиплексор МХ використовується для двостороннього інформаційного зв’язку з сусідніми ПЕ разом із елементом керування Control element. У мультиплексорі МХ також передбаченно інформаційні зв’язки (Shift) від сусідніх зліва і справа ПЕ.

На рис.3.8 показано особливості функціональної схеми матричного обчислювача та формувача вихідних сигналів

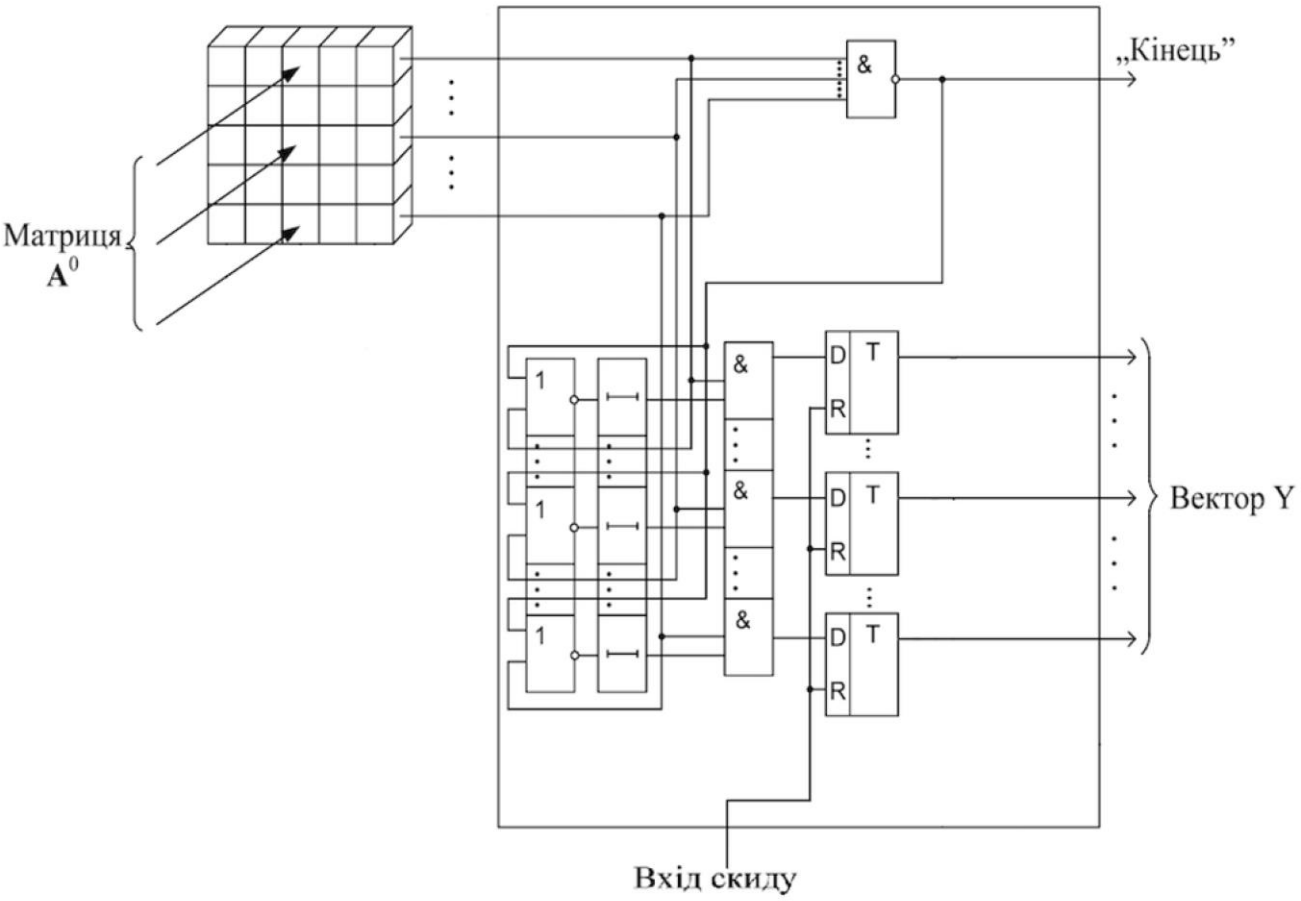


Рисунок 3.8 – Функціональна схема матричного обчислювача та формувача вихідних сигналів

Необхідно відмітити, що розглянуту структуру матричного обчислювача (рис. 3.6) за допомогою САПР Xilinx ISE Design Suite 13.2

було спроектовано і розміщено на ПЛІС XC6SIX45T сімейства Spartan-6. В результаті сегмент систолічного матричного обчислювача розміром 4x8 ПЕ і розрядністю 8 біт має такі показники: час спрацювання - 10.5 мс, частота - 100 Мгц, завантаження ресурсів введення/виведення – 88%, завантаження внутрішніх ресурсів – 10% [61].

3.1.1 Структурні та функціональні особливості нейроподібного класифікатора об’єктів

Аналіз структурних особливостей запропонованого матричного обчислювача разом з формувачем вихідних сигналів (рис. 3.6) можна розглянути як обчислювальну мапу, яку доповнено мапою ознак класифікації (рис. 3.9) [71, 72]. Отже, в цьому сенсі обчислювальна мапа має вигляд двовимірної гратки ПЕ (рис 3.7) (матричний блок), а мапа ознак – двовимірної гратки ПЕ (вузол аналізу).

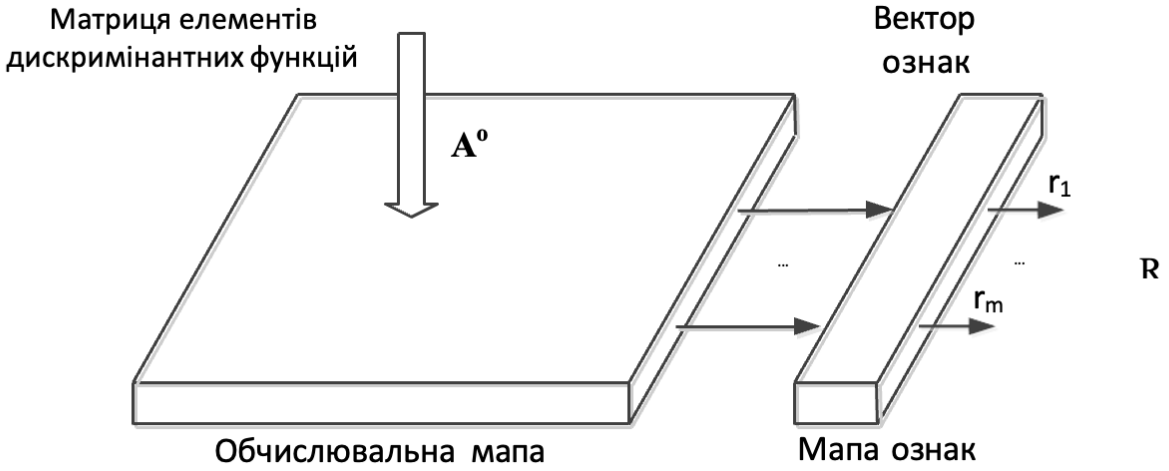


Рисунок 3.9 – Структура матричного обчислювача

В результаті такого аналізу обчислювальну мапу разом з мапою ознак (рис. 3.8) можна класифікувати як топологічну мапу.

Це пояснюється тим що вона є моделлю відображення ознак класифікації і має такі властивості[71]:

- просторове положення виходів відповідає конкретній області ознак даних, що виділені у вхідному просторі;
- топологія відображення містить перетворення двовимірного вхідного простору в одновимірний простір ознак;
- перетворення має адаптивний характер;
- в якості критерію відповідності використовується критерій максимуму дискримінантних функцій;
- в якості міри схожості використовується вектор мінелементів (внутрішніх порогів) як складових елементів векторних масивів;
- мапа ознак має топологію одновимірної гратки, яку задано вихідним простором;
- процес оброблення має релаксаційний характер.

Порівнюючи за своїми характеристиками цю мапу з мапою Кохонена [4-6], її можна визнати як мапу із самоорганізацією, а мапу ознак – як шар нейроподібних елементів, виходами якого можуть бути як вектор класів *Y*, так і вектор рангів *R*, оскільки вони обидва формуються за результатами вихідних сигналів обнуління *Z* (рис. 3.6)

У свою чергу, саме вектор рангів *R* представляє собою вектор вагів відповідних ЛДФ, які входять до матриці $\boldsymbol{A}^0$ у вигляді векторних масивів елементів . Причому, початкове одиничне значення елементів вектора рангів *R* можна розглядати як ініціалізацію синаптичних ваг ЛДФ у нейроподібного класифікатора. Крім того, нейроподібність матричної структури (рис. 3.8)

можна підтвердити тим, що тут застосовано три основні процеса саморганізації [4-6]:

- конкуренція, яка відбувається при визначенні максимальної ЛДФ;
- кооперація, оскільки в процесі сортування отриманих рангів можна визначити не тільки відповідний клас для вхідного масиву сигналів за максимальним рангом, але й найближчий до нього клас, як можливий варіант при кластеризації;
- синаптична адаптація, оскільки збільшуються ранги відповідних ЛДФ з кожним вилученням найменшої з них за сумою її елементів до моменту, коли залишається як мінімум одна ЛДФ у вигляді ненульового масиву.

У табл. 3.1 наведено порівняльну характеристику мапи Кохонена та запропонованох структури матричного обчислювача (рис. 3.8).

Отже, можлива організація топографічної мапи, яка відрізняється від відомої самоорганізаційної мапи ознак Кохонена за критерієм відповідності і метрики як кількісної міри схожості, а також за принципом функціонування. Але при цьому досягається однакова мета, а саме, реалізація процедури класифікації об’єктів та отримання результатів структурного (топологічного) представлення вхідних даних у вигляді векторів ознак.

3.2 Нейроподібний класифікатор з ранжуванням результатів

Розширення функціональних можливостей класифікатора об’єктів (рис. 3.10)

За рахунок ранжування результатів класифікації потребує включення ранжувальника у структуру максимізатора (рис.3.10).

Таблиця 3.1

Порівняльна характеристика мапи Кохонена (SOFM) і мапи вузла матричного оброблення

Характеристики мапи	Мапа Кохонена	Матричний обчислювач
Топологія мапи	Гратка	Гратка
Розмірність	Двовимірна	Двовимірна
Кількість шарів мапи	Один	Два
Топологія зв'язків елементів	Зв'язок із сусідніми нейронами	Зв'язок по стовпцях і по рядках в обчислювальній мапі
Співвідношення між розмірністю мапи і результатом оброблення	Кількість нейронів у шарі визначає кількість кластерів, що розпізнається	Кількість рядків в обчислювальній мапі визначає кількість класів, кількість стовпців – розмірність вхідного вектора
Вид вхідного простору	Неперервне	Дискретне
Вид вихідного простору	Дискретне	Дискретне
Вид мапи ознак	Двовимірна гратка	Одновимірна гратка
Налаштування ваг	Конкурентне навчання (без вчителя)	Обчислення рангів як ваг відповідних рядків матриці зважених вхідних сигналів
Метрика	Евклідова відстань	Вектор мінелементів
Максимальна правдоподібність (критерій відповідності)	Критерій мінімуму евклідової відстані	Критерій максимуму дискримінантних функцій
Функціональні можливості	Розділення векторів вхідних сигналів на групи (кластери). Синаптичні ваги мережі визначають кластери після навчання	Визначення масиву з максимальною сумою його елементів. Визначення рангів (ранжування масивів за сумою їх елементів). Сортування за рангами масивів.
Сфери застосування	Розпізнавання образів, стиснення зображень.	Класифікація, стиснення даних.

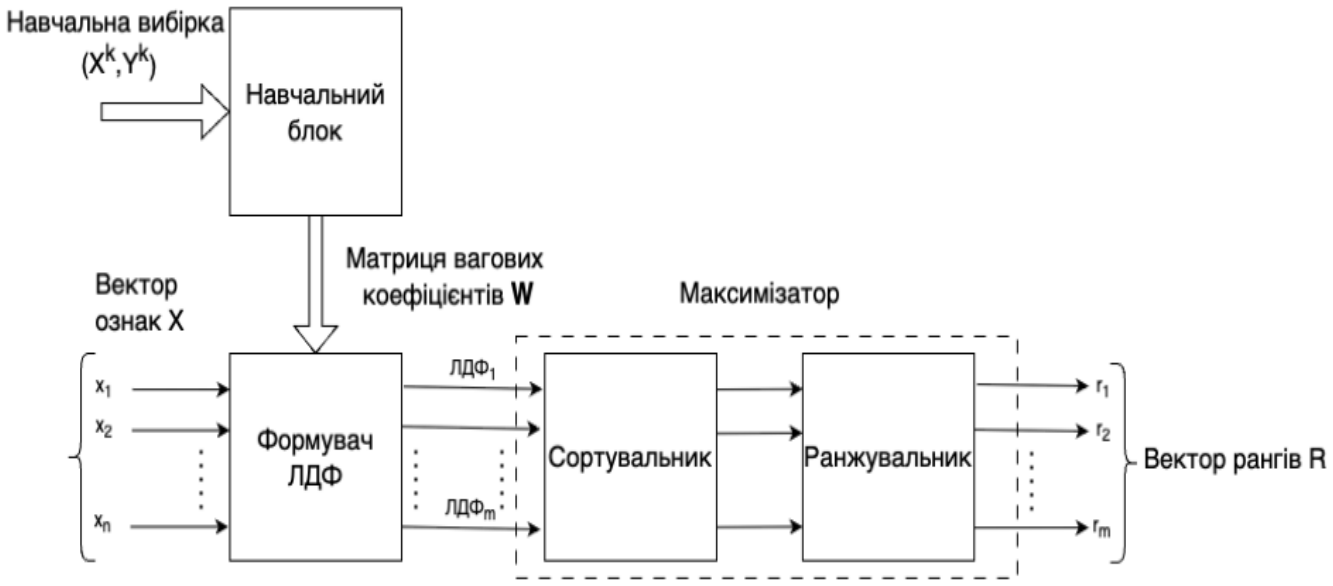


Рисунок 3.10 - Удосконалена структура класифікатора об’єктів з ранжуванням результатів

У свою чергу, застосування для виконання опереації “1 з *N*” мережного способу пошуку максимуму (рис. 2.6), а саме сортувальної операції, потребує реалізації максимізатора у вигляді сортувальника (рис. 3.10).

В даній роботі пропонується альтернативний метод сортування без порівняння та перекомутації в процесі сортування елементів числового масиву (рис.2.6), що фактично представляє 2-й спосіб процесу класифікації об’єктів (рис.2.4). Запропонований спосіб сортування базується на використанні швидкісної операції декремента одночасно до всіх елементів числового масиву, причому апаратно цей спосіб реалізується класифікатором із задіянням процесу конкуренції WTA (рис.2.4).

Приклад, наведений у табл. 2.2, показав можливість одночасно в єдиному процесі оброблення використати сортування із задіянням швидкісної операції декремента та ранжування із задіянням відповідно швидкісної операції інкремента.

Таким чином, можна відмітити застосування пари операцій декремент/інкремент. Такий підхід дає можливість сформувати на виході класифікатора вектор рангів *R*, але втрачається можливість сформувати вектор класів *Y*. З іншого боку, саме вектор рангів *R* вказує на найбільш вірогідний елемент вектора класів *Y*, тобто визначає результат класифікації об’єкта за його ознаками.

Отже, ранжування результатів класифікації дозволяє визначити не тільки найбільш вірогідний клас, до якого належить вхідний об’єкт, але найближчі до максимального ранги дозволяють розширити аналіз результатів класифікації або через повторний процес класифікації, або через розширення розмірності вхідного вектора ознак *X*.

Такий підхід знайде своє застосування у підсистемах підтримки прийняття рішень у складі експертних систем [10-12], зокрема при медичному діагностуванні [43,44,68].

3.2.1 Базові вузли нейроподібного класифікатора

Враховуючи альтернативний підхід до структури максимізатора (рис. 3.10), інтерес представляють два його базові вузли: сортувальник та ранжувальник. Якщо для реалізації сортувальння та ранжування задіяти швидкісні операції декремента та інкремента відповідно (табл. 2.2), то найбільш прийнятним рішенням є використання риверсивних лічильників [73].

На рис. 3.11 наведено структурну схему двох вузлів максимізатора. У свою чергу, сортувальник представляє собою обчислювач на лічильниках із задіянням операції декремента (зворотньої лічби), а ранжувальник містить аналізаттор реакцій із задіянням ознак нуля від обчислювача та пам'ять рангів на лічильниках із задіянням операціх інкремента (прямої лічби) [20]. На вхід

сортувальника подається *m* ЛДФ у вигляді числового масиву, а на виході ранжувальника формується результат у вигляді відповідних рангів *m* ЛДФ.

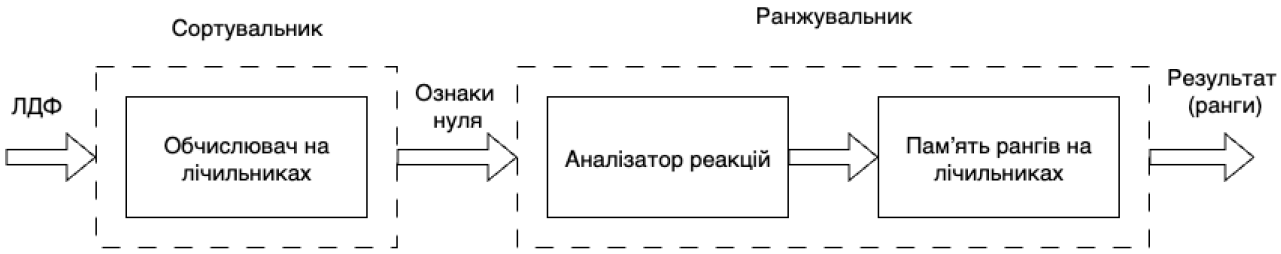


Рисунок 3.11 – Структурна схема сортувальника і ранжувальника

На рис. 3.12 представлено функціональну схему обох вузлів сортувальника та ранжувальника, зокрема показано *m* елементів маски ME та логічний елемент АБО, що входять до складу аналізатора реакцій у ранжувальнику (рис.3.10).

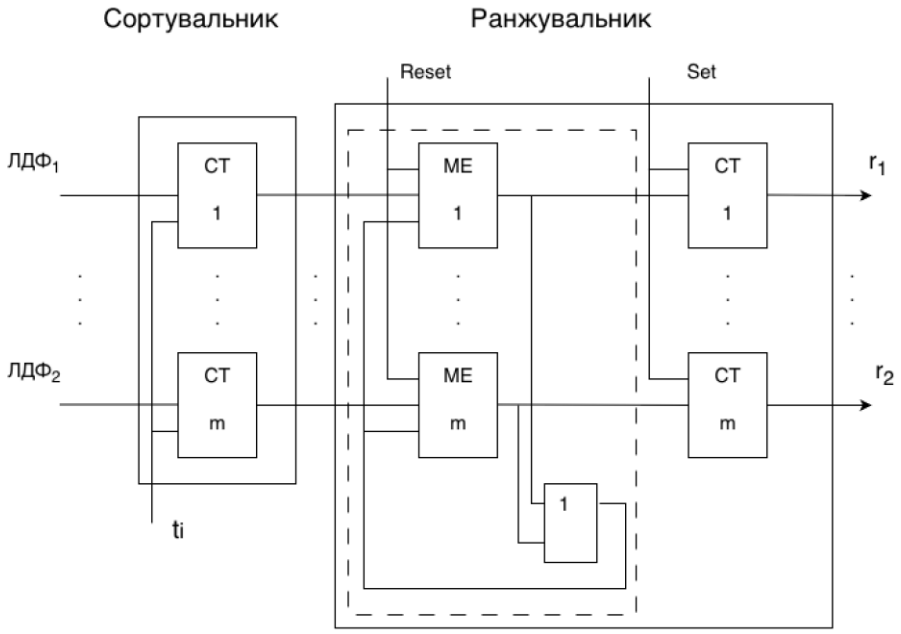


Рисунок 3.12 – Функціональна схема сортувальника та ранжувальника

У схемі показано крім та виходів рангів входи керування, а саме, вхід тактових імпульсів TI, вхід початкового встановлення Reset, вхід початкового стану Set, вхід дозволу Enable для елементів маски ME, а також виходи ознак нуля .

Максимізатор (рис. 3.12) має регулярну структуру, що містить n каналів оброблення, де n – розмірність числового масиву. Кожний канал містить лічильник обчислювача, елемент маски ME та лічильник пам’яті рангів. Крім того, всі вузли у кожному каналі мають регулярні та локальні зв’язки.

Разом з тим, сортувальник, аналізатор з елементами маски ME та пам’ять рагів мають регулярну структуру, оскільки містять однотипні базові вузли: лічильники та елементи маски ME. Виключення складає елемент АБО в аналізаторі реакцій, який необхідний для дозволу процесу маскування.

Отже, регулярну структуру максимізатора (рис.3.12) як у горизонтальному, так і вертикальному напрямках достатньо просто розташувати у мікросхемі ПЛІС типу FPGA, а також реалізувати просте модульне нарощування кількості входів, тобто вхідного числового масиву.

На рис. 3.13 представленно функціональну схему елемента маски ME [67]. На початку роботи RS-тригер встановлюється в одиничний стан за відповідним входом Reset. В процесі спрацювання елемента маски задіяно вхід ознаки нуля Z_i, який приходить від відповідного лічильника сортувальника та вхід дозволу Enable від елемента АБО.

RS-тригер спрацьовує (обнуляється) у момент обнуління відповідного лічильника сортувальника та формує сигнал ознаки маскування Mask відповідного лічильника ранжувальника. Цей сигнал маскує ранг відповідного лічильника сортувальника, який обнулився, і враховується елементом АБО при формуванні сигналу дозволу Enable.

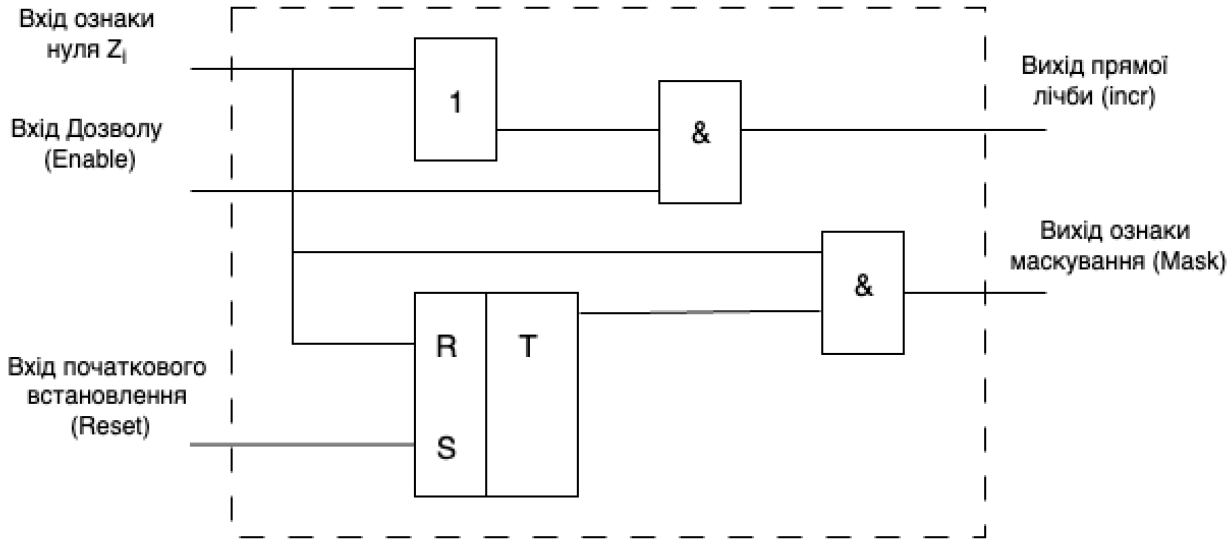


Рисунок 3.13 – Функціональна схема елемента маски

В іншому випадку в результаті спрацювання «відкритого» елемента маски ME формується одиничний сигнал на виході прямої лічби (інкремента) Incr, який подається на вхід лічби відповідного лічильника ранжувальника. Сигнал на виході ознаки маскування елемента маски ME формується за умови, якщо обнуляється відповідний лічильник сортувальника [20].

Структуру класифікатора об’єктів без навчального блока (рис.3.10) можна представити у нейромережному вигляді таким чином (рис. 3.14).

Порівняльний аналіз обох структур класифікаторів рис. 1.7 та рис. 3.14 показав таке.

По-перше, частина обох нейромережних класифікаторів має вигляд одношарового персептрона з лінійною функцією активації для формування *m* ЛДФ і залишається незмінною. Це відповідно вхідний шар та перший прихований шар відповідних елементів.

По-друге, вихідний шар у класифікатора–прототипа (рис. 1.7) виконує аналогічні функції як другий прихований шар у запропонованого класифікатора

(рис. 3.14), тому введено вихідний шар, що містить (m+1) нейроподібних елементів для формування рангів $r_1, \dots, r_m$.

По-третє, у другому прихованому шарі запропонованого класифікатора (рис. 3.14) реалізовано принцип конкуренції типу WTA в процесі сортування із задіянням паралельної операції декремента замість нейромережного підходу із задіянням від'ємних латеральних з'язків у вихідному шарі класифікатора-прототипу (рис. 1.7). В результаті не використовуються зворотні зв'язки у кількості m(m-1) як у класифікатора-прототипу (рис. 1.7).

У-четверте, у вихідному шарі запропонованого класифікатора (рис. 3.14) задіяно зворотній зв'язок через (m+1) нейроподібний елемент для маскування у процесі формування рангів $r_1, \dots, r_m$.

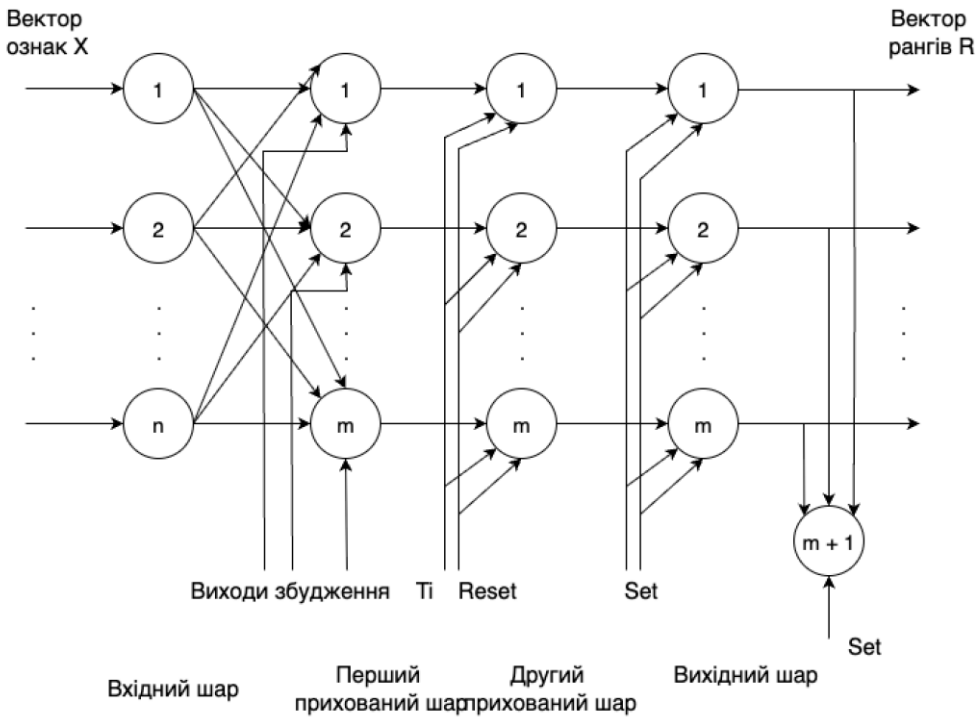


Рисунок 3.14 – Структурна схема нейроподібного класифікатора

3.3 Аналіз характеристик класифікатора з ранжуванням результатів класифікації

Серед базових показників ефективності будь-яких обчислювальних засобів важливими є часові та апаратні витрати, які необхідні для реалізації поставленних завдань. При цьому особливості методів оброблення даних вимагають знаходження компромісу між цими двома показниками.

Для обох розглянутих методів та засобів класифікації об’єктів важливою рисою є реалізаціє вирішального правила (1.5). Тому при аналізі загальних часових та апаратних витрат акцент буде зроблено саме на варіанти реалізації структури та принципу функціонування максимізатора у складі класифікатора об’єктів (рис. 3.2).

Для 1-го варіанта побудови класифікатора об’єктів (рис. 3.4), який реалізує метод класифікації з нормалізацією елементів ЛДФ, часові характеристики доцільно визначити, орієнтуючись на діаграму цього процесу на рис. 2.5.

Отже, не враховуючи формування початкової матриці $\boldsymbol{A}$ формувачем ЛДФ, тривалість процесу оброблення матриці $\boldsymbol{A}^0$ та формування вихідних сигналів має вигляд:

$$T_1 = T_{nor} + T_{sub} + T_{pr} + T_R\,, \tag{3.2}$$

де T_{nor} – час формування матриці зміщення $\boldsymbol{B}$; T_{sub} – час віднімання двох матриць $\boldsymbol{A}$ і $\boldsymbol{B}$; T_{pr} – час оброблення елементів матриці $\boldsymbol{A}^0$; T_R – час формування результату класифікації.

При аналізу формули (3.2) інтерес представляє тривалість процесу T_R що можна подати таким чином:

$$T_{pr} = (t_{ct} + t_{sh})N_{cep}\,, \tag{3.3}$$

де t_{ct} – час лічби на зменшення (операція декремента); t_{sh} - час зсуву на один двійковий розряд; N_{cep} – середнє значення кількості циклів оброблення, причому N_{cep} визначається як *O(mxn)* [66, 74].

Отже, найбільша тривалість процесу класифікаці з нормалізацією елементів ЛДФ напряму залежить від розмірності *mxn* матриці $\boldsymbol{A}^0$. Це пов'язано з тим, що в основі процесу методу оброблення даних за РЗ лежить принцип поступового вилучення з обнуленням найменших за значенням чисел масиву. Цей процес фактично представляє собою відсортування елементів числового масиву за зростанням їх значень [34], тобто є аналогічним класичному методу сортування "бульбашка" [2], але без збереження числових значень.

Відомо, що тривалість процесу класичного сортування "бульбашка" залежить від розмірності n масиву чисел, а саме $O(n^2)$ [2,63]. Але тривалість сортування з використанням методу РЗ із задіянням операції декремента напряму залежить від максимального значення одного з елементів числового масиву (табл. 2.2), тобто $O(a_{max})$.

Апаратні витрати класифікатора (рис. 3.4) також залежать від апаратних витрат на реалізацію матричного обчислювача як базового вузла у блоці класифікації, а саме:

$$K_2 = K_{nor} + K_{pr} + K_{lg} + K_R\,, \tag{3.4}$$

де K_{nor} – складність вузла нормалізації; K_{pr} – складність матричного обчислювача;

K_{lg} – складність формувача вихідних сигналів; K_R – складність пам'яті результатів.

Разом з тим, враховуючи структуру ПЕ матричного обчислювача (рис. 3.7), складність всього матричного обчислювача можна подати так:

$$K_{pr} = (K_{MX} + K_{CT} + K_{CE})mn\,, \tag{3.5}$$

де K_{MX} – складність мультиплексора МХ; K_{CT} – складність лічильника СТ; K_{CE} – складність елемента керування СЕ.

Разом з тим, необхідно відмітити регулярність структури матричного обчислювача (рис. 3.6) та нескладність його ПЕ (рис. 3.7). А це, у свою чергу, робить його придатним до розташування у мікросхемі ПЛІС середнього класу складності. Це підтверджують результати розміщення матричного обчислювача на базі ПЛІС сімейства MAX2 фірми Altera [64].

Реалізація на ПЛІС MAX2 серії EPM2210F256A5 одного рядка матричного обчислювача з трьома ПЕ з паралельним введеням даних у ПЕ має такі показники:

кількість задіяних логічних елементів < 1%; кількість задіяних виведень - 7%.

Ці дані свідчать про те, що у такого типу ПЛІС можна розмістити матричний обчислювач достатньо великих розмірів.

Для 2-го варіанта побудови класифікатора об’єктів (рис. 3.10), який реалізує процес класифікації з ранжуванням результатів, часові характеристики без врахування часу на формування ЛДФ, можна визначити таким чином:

$$T_2 = T_S + T_R\,, \tag{3.6}$$

де T_S – час сортування числових значень ЛДФ; T_R – час формування результату;

Основну роль тут відіграє перша складова T_S, а саме:

$$T_S = (2^q - 1)t_{ct}\,, \tag{3.7}$$

де q – розрядність даних.

При цьому, час T_R, враховуючи структуру аналізатора реакцій та памяті рангів (рис. 3.12) і (рис. 3.13), має такий вигляд:

$$T_R = t_{lg} + t_{ct}\,, \tag{3.8}$$

де t_{lg} – час спрацювання елементів аналізатора реакцій.

Відповідно апаратна складність максимізатора (рис. 3.10) становить:

$$K_2 = (2K_{CT} + K_{ME})n + k_{lg}\,, \tag{3.9}$$

де K_{ME} – складність елементів маски; k_{lg} – складність логічного елемента АБО; n – розмірність числового масиву.

Таким чином, найбільший вклад в апаратні витрати максимізатора (рис.3.10)
вносить апаратна складність лічильників, що реалізують швидкісні операції декремента/інкремента. Але логічна частина максимізатора відрізняється простотою та регулярністю зв'язків, що є важливим фактором для застосування перспективної елементної бази - ПЛІС. Крім того, застосування ПЛІС з орієнтовним часом спрацювання базових вузлів в таких мікросхемах у наносекундному діапазоні [75] дозволяє отримати кінцевий результат у мілісекундному діапазоні.

3.4 Реалізаційні моделі нейроподібних класифікаторів об'єктів з розширеними функціональними можливостями

Враховуючи напрям дослідження даної роботи, а саме медичне діагностування та системи автономного керування мобільних роботів (рис. 1.2) має сенс показати місце, яке займає класифікація об'єктів, як одна з найпоширеніших процедур розпізнавання образів [3-6].

У табл 3.2 показано рівні оброблення, аналізу та розпізнавання візуальної інформації, де показано всі процеси, що передують класифікації об’єктів, а також представлено процес ранжування результатів класифікації, який сприяє прийняттю відповідного рішення в експертних системах різного призначення.

Таблиця 3.2

Рівні оброблення, аналізу та розпізнавання візуальної інформації

Характер процесу	Цільове призначення рівня
Попереднє оброблення (препроцесування)	Покращення якості зображення, збільшення відношення сигнал/шум
Загальний і детальний аналіз (сегментація)	Аналіз зображень об’єктів
Виділення й аналіз ознак (кореляція)	Ідентифікація зображення
Класифікація об’єктів	Розпізнавання об’єктів, віднесення об’єкта до певного класу
Ранжування результатів	Прийняття вірогідного рішення

У табл 3.3 представлено приклади застосування таких асоціативних операцій, як сортування та ранжування числових даних, причому найбільш цікавий момент у табл 3.3 пов’язаний саме із засобами реалізації даних операцій. Так для нейромережної класифікації об’єктів для реалізації механізму конкуренції типу «1 з N» пропонується задіяти сортувальник, а для підсистеми підтримки прийняття рішень важливою складовою є ранжувальник.

Разом з тим, для систем автономного керування мобільного робота для сприйняття інформації та для розпізнавання об’єктів необхідною умовою є застосування класифікатора об’єктів з розширеними функціональними можливостями, що передбачає використання як сортувальника, так і ранжувальника. Це пов’язано з функціональними особливостями системи

автономного керування мобільного робота, а саме з реалізацією як процедури класифікації об’єктів, так і з адаптацією у навколишньому середовищі [76,77].

Таблиця 3.3

Прикладний аспект застосування асоціативних операцій

Область застосування	Функціональне призначення	Базові операцій	Засоби реалізації
Нейромережна класифікація обєктів	Реалізація механізму конкуренції «1 з N»	Визначення максимального числового значення та його місцерозташування	Сортувальник
Підсистема підтримки прийняття рішень в експертній системі	Визначення рангів входження об’єктів до класів розпізнавання	Ранжування даних як результат сортування	Ранжувальник
Система автономного керування мобільного робота	Сприйняття інформації Розпізнавання об’єктів	Класифікація, адаптація	Класифікатор: сортувальник + ранжувальник

Про важливість класифікації об’єктів з можливістю ранжування результатів свідчить необхідність їх входження в інтелектуальні навігаційні апаратні засоби у складі базової архітектури мобільної робототехнічної платформи (МРТП) (рис.3.15), в якій організовано інтелектуальну систему керування рухом і захистом передачі даних [15].

Саме класифікація об’єктів з ранжуванням дозволяє реалізовувати навігацію та ухилення від перешкод в процесі руху мобільного робота [78]. При цьому бажано забезпечити при автономному керувані мобільного робота таких характеристик, як надійність у складних умовах, компактність та швидкодія

апартної реалізації, заборона несанкціонованого доступу до процесу керування [76].

Все це може забезпечити, по-перше, апаратна реалізація базових вузлів блока керування, а по друге, орієнтація на використання програмованих логісчних ІС (ПЛІС) з різними апаратними та функціональними можливостями [75,76].

Рисунок 3.15 – Базова архітектура МРТП з інтелектуальною системою управління рухом і захистом передачі даних [15]

Отже, запропонований підхід до проектування та реалізації базових вузлів системи інтелектуального з керування мобільними роботами дозволяють зняти деякі ключові технічні виклики.

3.5 Систолічний підхід до паралельного оброблення одно- та двовимірних масивів даних

Звертаючи увагу на ключові технічні показники обчислювальних засобів, а саме на швидкодію та апаратні витрати [73], які знаходяться у технічному протиріччі, можна вказати на можливість компромісу між цими показниками.

Цього можна досягти, застосовуючи систолічний підхід до оброблення як одновимірних, так і двовимірних масивів даних [62], орієнтуючись на використання ПЛІС [75]. Саме мікросхеми ПЛІС забезпечують при спацюванні надійність, компактність апаратної реалізації та недоступність несанкціонованого доступу до програмного забезпечення [75].

При цьому, при розташуванні у мікросхемі ПЛІС обчислювальних засобів до них висуваються такі вимоги, як максимальний паралелізм оброблення масивів даних та висока регулярність структури. Саме такий підхід до оброблення значних масивів даних забезпечує систолічний метод, що реалізується на адекватній систолічній структурі [61, 64].

Систолічні методи оброблення даних являють собою клас паралельних алгоритмів і апаратних структур, що забезпечують ефективну організацію потоків даних між обчислювальними елементами.

В основі принципу систолічності лежить ритмічна передача інформації між елементами мережі, що нагадує фізіологічний процес руху крові під час систоли серця. Кожен обчислювальний елемент отримує, обробляє та передає дані далі у суворо визначеному тактовому режимі, дозволяє мінімізувати простої та підвищити продуктивність.

Систолічні структури виникли як відповідь на потребу у високопродуктивних обчисленнях, зокрема у задачах лінійної алгебри, цифрової обробки сигналів, оброблення потокових даних та інженерних задачах

оптимізації. Основна причина їх популярності полягає у відсутності необхідності розподіляти складні алгоритми між ядрами, як це відбувається у класичних мультипроцесорних архітектурах. Натомість структура визначається статично, а потоки даних - циклічно, що значно спрощує апаратну реалізацію.

Один із головних принципів побудови систолічних масивів - регулярність. Це означає, що всі обчислювальні елементи є однаковими, мають однаковий набір операцій та взаємодіють з сусідами у передбачуваних напрямках. Такий підхід не лише зменшує апаратну складність, але і робить можливим ефективний синтез схем у FPGA або ASIC без необхідності оптимізації під кожну конкретну задачу.

Ключовим аспектом систолічних структур є локальність обчислень. Дані зберігаються у регістрах обчислювального елементу, а передаються між елементами лише у разі потреби. Таким чином, мінімізується доступ до пам’яті, що у традиційних архітектурах є вузьким місцем. Систолічний підхід природним чином масштабуються: збільшення кількості елементів прямо пропорційно збільшує продуктивність.

Одновимірні систолічні масиви використовуються у задачах, де дані обробляються послідовно, наприклад, при обчисленні згорток, фільтрації сигналів, виконанні рекурсивних операцій. Двовимірні масиви дозволяють виконувати операції над матрицями та зображеннями, забезпечуючи рівень паралелізму, недосяжний у класичних системах.

Однією з найвідоміших реалізацій є двовимірна систолічна решітка для множення матриць, запропонована Кунгом. У ній елементи розташовані у вигляді прямокутної матриці, а дані переміщуються по горизонталі та вертикалі. Кожен елемент виконує базову операцію множення-з-накопиченням. Ця структура стала фундаментом для створення сучасних прискорювачів штучного інтелекту.

У сучасних апаратних прискорювачах глибинного навчання систолічні масиви відіграють ключову роль. Наприклад, Tensor Processing Unit (TPU), створений Google, містить великий двовимірний систолічний масив, який виконує мільярди операцій множення матриць на секунду. Принцип роботи TPU заснований на постійному русі даних між елементами та відсутності необхідності повторного звернення до зовнішньої пам’яті.

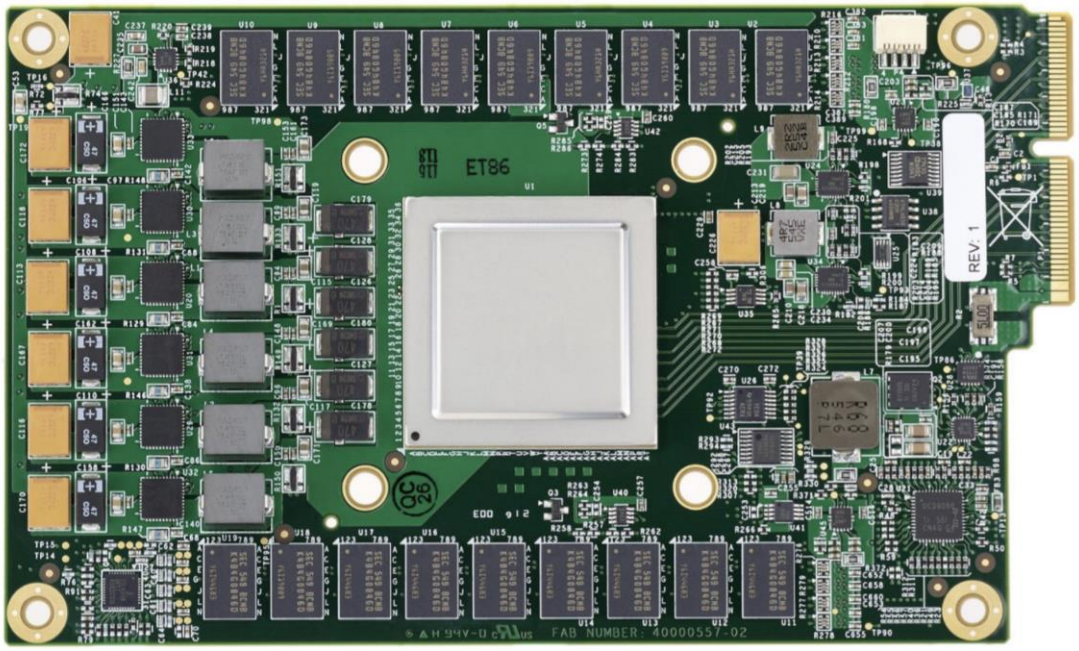


Рисунок 3.16 - Фізичний вигляд Tensor Processing Unit

Процес синтезу систолічної структури починається з аналізу математичної моделі задачі. Будується граф потоків даних, визначаються точки обміну інформацією. Далі обирається топологія масиву (лінійна, двовимірна, трикутна), після чого створюється апаратна схема. Важливо забезпечити, щоб усі елементи отримували дані у потрібні моменти часу, інакше структура працюватиме некоректно.

Систолічні методи також використовуються у телекомунікаційних системах, де потрібно виконувати фільтрацію сигналів у реальному часі.

Зокрема, фільтри Фір та Іір можуть бути реалізовані у вигляді лінійного систолічного масиву, де кожен елемент відповідає за множення на коефіцієнт фільтра.

Переваги систолічних структур включають високу енергоефективність, мінімальне споживання пам’яті, латентність та масштабованість. Недоліки - статичність структури та складність адаптації до задач із нерегулярними потоками даних. Однак у задачах, де обчислення мають регулярний характер, систолічні структури залишаються одним із найпродуктивніших підходів.

3.6 Висновки до третього розділу

1. Запропоновано реалізаційну модель систолічної матричної структури у складі 1-го варіанта класифікатора об’єктів для оброблення елементів ЛДФ з нормалізацією за методом різницевих зрізів, що забезпечує високий рівень паралелізму оброблення елементів матриці по її стовпцях і рядках.
2. Наведений 1-й варіант нейроподібного класифікатор об’єктів може бути реалізований у таких системах штучного інтелекту, де застосовано принцип самоорганізації нейробіологічних систем, коли просторове положення вихідних нейронів у топологічній мапі відповідає конкретній області ознак даних, виділених із вхідного простору.
3. Запропоновано структуру максимізатора у складі другого варіанта класифікатора об’єктів, яка має регулярність побудови як у горизонтальному, так і вертикальному напрямках, що дозволить ефективно її розмістити у мікросхемі ПЛІС типу FPGA з можливістю модульного нарощування.

4. Таким чином, можлива організація топографічної мапи, яка відрізняється від відомої самоорганізаційної мапи ознак Кохонена за критерієм відповідності і метрики як кількісної міри схожості, а також за принципом функціонування. Але при цьому досягається однакова мета, а саме, реалізація процедури класифікації об’єктів та отримання результатів структурного (топологічного) представлення вхідних даних у вигляді векторів ознак.
5. Аналіз базових показників часових та апаратних витрат для обох типів нейроподібних класифікаторів показав, що застосування систолічного підходу до оброблення матриці елементів ЛДФ, а також паралельне задіяння швидкісних операцій декремента/інкремента забезпечують їх адекватну реалізацію на одно – та двовимірних структурах. А це, у свою чергу, дозволяє їх ефективне розміщення у мікросхемах ПЛІС з можливістю спрацювання у мілісекундному діапазані.

РОЗДІЛ 4
ПРОГРАМНІ ЗАСОБИ ДЛЯ МОДЕЛЮВАННЯ ПРОЦЕСІВ КЛАСИФІКАЦІЇ

4.1 Вихідні дані для імітаційного моделювання процесу класифікації на прикладі медичного діагностування захворювань

Для імітаційного моделювання процесу класифікації з ранжуванням результатів обрано приклад медичного діагностування захворювань на апендицит, а саме [32]:

1) гангренозний;
2) флегмонозний;
3) катаральний;
4) інша патологія живота.

Для цього використано 8 базових симптомів x1,…x8 (табл. 4.1) . Для створення таблиці навчальної вибірки по всіх діагнозах задіяно результати 103 історій хвороби пацієнтів з підтвердженими діагнозами.

У табл. 4.2 наведено частину матриці навчальної вибірки, де наведено по 5 прикладів на кожний з чотирьох діагнозів. Для імітаційного моделювання задіяно коефіцієнти чотирьох лінійних дискримінантних функцій (ЛДФ), які розраховано за допомогою ППП Statistica Використані ЛДФ мають такий вигляд:

$$
\begin{aligned}
\text{ЛДФ1} &= -63.0 + 9.8x1 + 3.6x2 + 7.8x3 + 5.2x4 + 14.3x6 + 11.8x7 + 11.3x8 \\
\text{ЛДФ2} &= -57.4 + 8.3x1 + 4.9x2 + 6.2x3 + 4.3x4 + 13.5x6 + 11.7x7 + 10.6x8 \\
\text{ЛДФ3} &= -49.6 + 9.4x1 + 4.7x2 + 5.5x3 + 3.0x4 + 12.3x6 + 12.0x7 + 8.3x8 \quad (4.1) \\
\text{ЛДФ4} &= -23.0 + 6.3x1 + 2.5x2 + 5.3x3 + 2.8x4 + 7.8x6 + 7.0x7 + 5.8x8
\end{aligned}
$$

Таблиця 4.1

Симптоми апендициту та їх кодування

Симптом	Найменування	Степені та коди симптомів
x1	Болі в правій повздовжній області	1 - незначне 2 - виражене
x2	Тривалість болей у правій повздовжній області	1 – більше двох діб 2 – 25-48 год 3 – 13-24 год 4 – до 12 год
x3	Частота пульсу	1 – до 80 2 – 81-100 3 – більше 100 уд/хв
x4	Лейкоцити крові	1 – до 8 2 – 8-14 3 – більше 14 тис/мкл
x5	Змінення язика	0 – необложений 1 – обложений
x6	Симптом Щіткена-Блюмберга	0 – відсутній 2 – присутній
x7	Симптоп Ровзинга	0 – відсутній 2 – присутній
x8	Захисне м’язове напруження	0 – відсутній 2 – присутній

Таблиця 4.2

Фрагмент таблиці навчальної вибірки

Index	x1	x2	x3	x4	x5	x6	x7	x8
1	2	3	1	2	1	2	2	2
	2	2	2	2	1	2	0	2
	2	3	1	3	1	2	2	2
	2	2	3	1	1	0	2	2
	2	3	2	2	1	2	2	0
2	2	3	1	2	1	2	2	2
	1	4	2	1	0	2	0	2
	2	3	1	3	1	0	2	2
	1	4	2	2	1	2	2	2
	2	4	1	2	0	2	2	0
3	1	3	1	2	1	0	2	2
	2	4	1	1	0	2	0	0
	2	3	1	2	1	0	2	2
	2	4	2	2	1	2	0	0
	1	2	1	1	0	0	2	2
4	1	2	1	1	0	0	0	0
	1	1	2	1	0	0	0	0
	1	3	1	1	1	0	0	0
	2	1	1	2	0	0	0	0
	1	2	1	1	0	0	0	0

У Додатку Б предсавлено вихідні дані в якості навчальної вибірки для процесу класифікації з ранжуванням.

4.2 Опис програмного модуля обчислення ЛДФ

Програмний модуль потрібно розмістити на комп’ютері з будь-якою операційною системою, де можна встановити та налаштувати Visual Studio Code та Python.

Програмний модуль реалізовано мовою програмування Python. Основна мета - визначення максимального значеня серед лінійних дискримінантних функцій (ЛДФ) для чотирьох варіантів математичних моделей на основі вхідних значень ознак x1,…,x8. Модуль не вимагає сторонніх бібліотек і може бути запущений на будь-якій операційній системі з Python 3.6 або вище.

Програма є консольною, результат обчислення виводиться у терміналі. У базовій версії значення ознак x1,…,x8 задаються вручну всередині скрипта.

Програма обчислює значення чотирьох варіантів лінійних дискримінантних функцій (ldf1, ldf2, ldf3, ldf4) на основі заданих коефіцієнтів та вхідних ознак. Після цього визначається максимальне з них і виводиться результат у вигляді рангів всіх ЛДФ. Програма дозволяє швидко порівняти різні моделі класифікації та визначити найбільш вірогідний результат класифікації через ранги ЛДФ.

Основні системні вимоги до програми: операційна система Windows (Microsoft Windows 11,10,8,7 (64-bit)), Linux (GNOME або KDE), MacOS(Apple macOS 10.8.5 або новіше) або будь-яка інша, яка дозволить розмістити програму, написану на Python, з наявністю підключення клавіатури та миші; процесор – x86-65 Intel або AMD або ARM (для Apple); оперативна пам’ять – не менше 2 Гб, рекомендується 4 Гб, мінімальна роздільна здатність екрану – 1280 x 800, жорсткий диск – від 128 Гб.

Вимоги пристроїв користувачів: операційна система Windows (Microsoft Windows 11,10,8,7 (64-bit)), Linux (GNOME або KDE), MacOS(Apple macOS 10.8.5 або новіше)

Програма зберігається у вигляді одного файлу, наприклад main.py. Запуск здійснюється командою python main.py

Вихідні дані: обчислені значення ldf1, ldf2, ldf3, ldf4 та визначення максимального з них через їх ранги.

Опис роботи із програмою

1. Користувач відкриває файл main.py у редакторі або терміналі.
2. Запускає програму командою python main.py.
3. У терміналі відображається повідомлення з максимальним значенням ЛДФ серед чотирьох варіантів з формуванням їх рангів.
4. У разі потреби користувач може змінити значення змінних x1, x2, ..., x8 у коді програми.

Даний програмний модуль реалізовано мовою програмування Python та призначений для обчислення значень лінійних дискримінантних функцій (ЛДФ) для декількох груп даних, порівняння результатів та аналізу точності класифікації. Програма використовується у дослідженні методів асоціативної обробки даних для задач медичної діагностики.

Структура програми складається з головного файлу main.py та чотирьох допоміжних модулів, що містять навчальні вибірки: first_group, second_group, third_group, fourth_group. Кожна група представлена списком словників, де кожен словник описує один об’єкт з параметрами x1…x8.

Імпорт навчальних груп (рис. 4.1), фрагмент коду для імпорту навчальних вибірок:

```
from groups.first import first_group
from groups.second import second_group
from groups.third import third_group
from groups.fourth import fourth_group


learn_info = {
    '1': first_group,
    '2': second_group,
```

```
    '3': third_group,
    '4': fourth_group,
}
```

Рисунок 4.1 - Структура каталогів

Основна функція check_ldf() є ключовим обчислювальним компонентом програми. Вона приймає значення параметрів x1…x8, а також індекси групи та рядка. На основі цих даних обчислюються значення чотирьох лінійних дискримінантних функцій (ЛДФ).

```
def check_ldf(group_index, row_index, x1, x2, x3, x4, x6, x7, x8):
    ldf1 = round(-63.0 + 9.8*x1 + 3.6*x2 + 7.8*x3 + 5.2*x4 + 14.3*x6 + 11.8*x7 +
11.3*x8, 2)
    ldf2 = round(-57.4 + 8.3*x1 + 4.9*x2 + 6.2*x3 + 4.3*x4 + 13.5*x6 + 11.7*x7 +
10.6*x8, 2)
    ldf3 = round(-49.6 + 9.4*x1 + 4.7*x2 + 5.5*x3 + 3.0*x4 + 12.3*x6 + 12.0*x7 +
8.3*x8, 2)
    ldf4 = round(-23 + 6.3*x1 + 2.5*x2 + 5.3*x3 + 2.8*x4 + 7.8*x6 + 7.0*x7 +
```

```
5.8*x8, 2)

    ldfs = [
        {'value': ldf1, 'name': 'ldf1'},
        {'value': ldf2, 'name': 'ldf2'},
        {'value': ldf3, 'name': 'ldf3'},
        {'value': ldf4, 'name': 'ldf4'},
    ]

    sorted_ldfs = sorted(ldfs, key=lambda item: item['value'])
    ranked = []

    for i, item in enumerate(sorted_ldfs):
        ranked.append({
            'ldfValue': item['value'],
            'ldfName': item['name'],
            'rang': i + 1
        })
    return ranked
```

Кожна ЛДФ є лінійною комбінацією ознак та відповідних коефіцієнтів. Після обчислення значення ЛДФ сортуються за зростанням та кожному присвоюється ранг.

Обробка даних у циклі, основна частина програми проходить по всіх групах та всіх елементах усередині групи. Для кожного прикладу викликається check_ldf(), а результати накопичуються.

```
for group_index, group_data in learn_info.items():
    for row_index, item in enumerate(group_data):
        x1 = item['x1']
        x2 = item['x2']
        x3 = item['x3']
        x4 = item['x4']
        x6 = item['x6']
        x7 = item['x7']
        x8 = item['x8']

        handled_ldf_arr = check_ldf(group_index, row_index, x1, x2, x3, x4, x6, x7, x8)
        result.append(handled_ldf_arr)
        print(f"groupIndex: {group_index} rowIndex: {row_index} {handled_ldf_arr}")
```

Аналіз точності класифікації на завершальному етапі програма порівнює, чи є максимальна ЛДФ тим самим, що відповідає реальній групі об’єкта. Це дозволяє оцінити відсоток коректної класифікації. Результати виконання програми показано на (рис. 4.2).

```
for group_index, group_data in learn_info.items():
    max_ldf_count = 0
    for row_index, item in enumerate(group_data):
        handled_ldf_arr = check_ldf(group_index, row_index, ...)
        sorted_array = sorted(handled_ldf_arr, key=lambda item: item['ldfValue'],
reverse=True)
        max_ldf = sorted_array[0]

        if max_ldf['ldfName'] == f"ldf{group_index}":
```

```
        max_ldf_count += 1

    print(f"GROUP_{group_index}", max_ldf_count, len(group_data))
    print(f"Accuracy: {(max_ldf_count / len(group_data)) * 100:.2f}%")
```

```
dmytrokbackendbs@Dmytros-MacBook-Pro py % python3 main.py
x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 0 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldf
Value': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 2, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 1 [{'ldfValue': 41.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 38.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldf
Value': 38.0, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 36.8, 'ldfName': 'ldf3', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 2 [{'ldfValue': 65.6, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 64.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldf
Value': 63.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 52.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 3, x4: 1, x6: 0, x7: 2, x8: 2
groupIndex: 1 rowIndex: 3 [{'ldfValue': 38.9, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 38.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldf
Value': 38.6, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 36.5, 'ldfName': 'ldf2', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 1 rowIndex: 4 [{'ldfValue': 48.9, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 45.6, 'ldfName': 'ldf1', 'rang': 3}, {'ldf
Value': 45.3, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 42.9, 'ldfName': 'ldf4', 'rang': 1}]
```

Рисунок 4.2 – Детальні результати моделювання

4.3 Опис програмного модуля для медичного діагностування захворювань за ЛДФ з графічним інтерфейсом

Даний програмний модуль реалізує інтелектуальну систему класифікації станів апендициту на основі лінійних дискримінантних функцій (LDF). Програма створена з використанням бібліотеки PyQt6, що забезпечує графічний інтерфейс користувача (GUI) у середовищі macOS, Windows та Linux. Користувач вводить значення восьми діагностичних ознак X1–X8, після чого програма обчислює значення чотирьох ЛДФ та визначає їх ранги.

Програма призначена для класифікації клінічних випадків на основі дискримінантних моделей. Вона дозволяє здійснювати оперативне експертне оцінювання стану пацієнта за допомогою восьми ключових ознак. Результати представлено у формі ранжованих LDF, що характеризують ступінь приналежності до певного клінічного стану.

Програмний код складається з трьох основних частин:

1. Функція calculate_ldf_values() - обчислення значень ЛДФ та ранжування.

2. Клас AppendicitisApp - графічний інтерфейс користувача.

3. Функція main() - запуск та коректне завершення програми.

4. Функція calculate_ldf_values()

Функція обчислює значення чотирьох лінійних дискримінантних функцій (LDF1–LDF4) на основі введених користувачем значень X1–X8. Кожна з формул є результатом статистичного аналізу та має вигляд лінійної моделі. Всі обчислення здійснюються з округленням до двох знаків після коми. Після цього результати сортуються за зростанням, а кожній ЛДФ присвоюється відповідний ранг.

```
def calculate_ldf_values(x1, x2, x3, x4, x5, x6, x7, x8):
    ldf1 = round(-63.0 + 9.8*x1 + 3.6*x2 + 7.8*x3 + 5.2*x4 + 14.3*x6 + 11.8*x7 + 11.3*x8, 2)
    ldf2 = round(-57.4 + 8.3*x1 + 4.9*x2 + 6.2*x3 + 4.3*x4 + 13.5*x6 + 11.7*x7 + 10.6*x8, 2)
    ldf3 = round(-49.6 + 9.4*x1 + 4.7*x2 + 5.5*x3 + 3.0*x4 + 12.3*x6 + 12.0*x7 + 8.3*x8, 2)
    ldf4 = round(-23.0 + 6.3*x1 + 2.5*x2 + 5.3*x3 + 2.8*x4 + 7.8*x6 + 7.0*x7 + 5.8*x8, 2)
```

```
    result = [
        {'name': 'Гангренозний', 'code': 1, 'value': ldf1},
        {'name': 'Флегмонозний', 'code': 2, 'value': ldf2},
        {'name': 'Катаральний',  'code': 3, 'value': ldf3},
        {'name': 'Інше',         'code': 4, 'value': ldf4},
    ]

    result_sorted = sorted(result, key=lambda x: x['value'], reverse=False)

    for i, item in enumerate(result_sorted):
        item['rank'] = i + 1

    return result_sorted
```

Клас AppendicitisApp відповідає за побудову графічного вікна, розміщення елементів інтерфейсу та обробку взаємодії користувача. Вікно містить поля для введення параметрів X1–X8, кнопку «Класифікувати» та чотири мітки для відображення результатів ранжування.

```
class AppendicitisApp(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Класифікація (LDF)")
        self.setGeometry(300, 200, 700, 400)
```

```
        layout = QGridLayout()

        labels = ["X1","X2","X3","X4","X5","X6","X7","X8"]
        self.inputs = []

        for i, lbl in enumerate(labels):
            label = QLabel(lbl)
            label.setFont(QFont("Arial", 12))
            layout.addWidget(label, 0, i)

            inp = QLineEdit()
            layout.addWidget(inp, 1, i)
            self.inputs.append(inp)
```

Функція main() створює застосунок PyQt6, ініціалізує головне вікно та забезпечує коректне завершення програми навіть у випадку аварійного закриття.

```
def main():
    app = QApplication(sys.argv)
    window = AppendicitisApp()
    window.show()

    try:
        app.exec()
    finally:
```

```
    app.quit()
    sys.exit(0)
```

Клас AppendicitisApp відповідає за побудову графічного інтерфейсу користувача та логіку взаємодії. При ініціалізації створюється основне вікно, розміщується сітка елементів, додаються текстові поля для введення параметрів X1…X8.

1. Ініціалізація - в конструкторі встановлюються заголовок, розміри вікна та створюється QGridLayout для впорядкування віджетів. Кожен параметр X представлений QLabel і QLineEdit, що дозволяє користувачу вводити значення.

2. Кнопка "Класифікувати" - при натисканні викликається метод on_classify, який зчитує значення полів, виконує перевірку на коректність і передає дані у функцію calculate_ldf_values.

3. Відображення результатів - Після розрахунку ЛДФ результати відображаються у відповідних QLabel. Кожен результат містить назву діагнозу та ранг.

4. Обробка помилок - якщо користувач вводить некоректні дані, виводиться QMessageBox із повідомленням про помилку. Це забезпечує надійність і захист від некоректного введення.

5. Завершення програми - метод main забезпечує запуск QApplication та коректне завершення процесу через блок finally, гарантуючи, що GUI повністю закриється після завершення роботи.

Результат виконання програми з графічним інтерфейсом зображено на рис. 4.3.

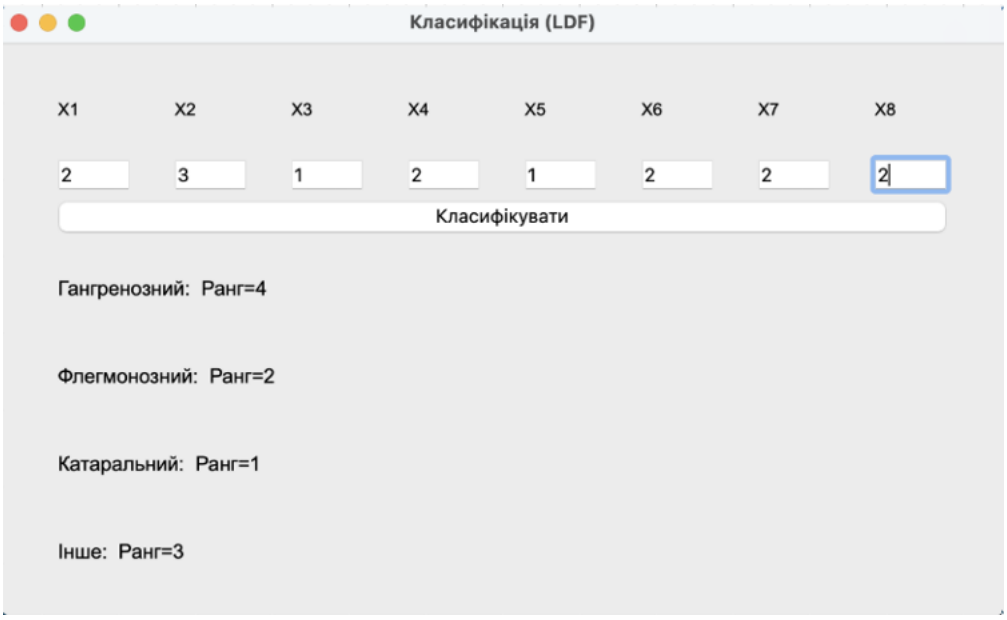


Рисунок 4.3 – Результат виконання програми з графічним інтерфейсом

У Додатку В наведено лістинг програми для імітаційного моделювання процесу класифікації з ранжуванням.

У Додатку Д представлено результати виконання програми для імітаційного моделювання процесу класифікації з ранжуванням.

4.4 Аналіз результатів моделювання

За результатами імітаційного моделювання процесу класифікації з ранжуванням на прикладі медичного діагностування захворювань на апендицит існує можливість визначення такого показника ефективності як чутливість методу діагностування як чутливість[30] у табл. 4.3.

Чутливість – це відносна частота віднесення істинно хворого до класу хворих, тобто відношення кількості правильних діагнозів до загальної кількості визначенних діагнозів.

тим, що симптоми для цих випадків гострого апендициту значно перекриваються. Для підвищення точності діагностування можна застосувати

або розширення списку симптомів, або повторний аналіз результатів дослідження, або збільшення навчальної вибірки для уточнення коефіцієнтів ЛДФ.

Таблиця 4.3

Показники ефективності методу діагностування

Номер діагнозу	Назва	Чутливість
1	Гангренозний	71.43%
2	Флегмонозний	44.00%
3	Катаральний	50.00%
4	Інша патологія живота	100.00%

Невисока точність діагностування для 2-го та 3-го діагнозів пояснюється

Таблиця 4.4

Перевірочні дані

Діагноз	Симптоми							
	x1	x2	x3	x4	x5	x6	x7	x8
y_1	2	3	1	2	1	2	2	2
y_2	1	4	2	1	0	2	0	2
y_3	2	3	1	2	1	0	2	2
y_4	1	2	1	1	0	0	0	0

За розробленою блок-схемою алгоритму (рис.4.1) виконано класифікацію з ранжуванням із застосуванням списку закодованих симптомів (табл. 4.1) та сформованими ЛДФ (4.1) для діагностування чотирьох діагнозів

апендициту.Ранжування діагнозів виконувалось таким способом: 4 – найбільш імовірний, 1 – найменш імоворівний ранг діагнозів. У табл. 4.4 наведено один з варіантів набору симптомів для чотирьох діагнозів y_1, y_2, y_3, y_4.

Результати імітаційного моделювання процесу класифікації з ранжуванням для даних табл.4.4 наведено на рис.4.2

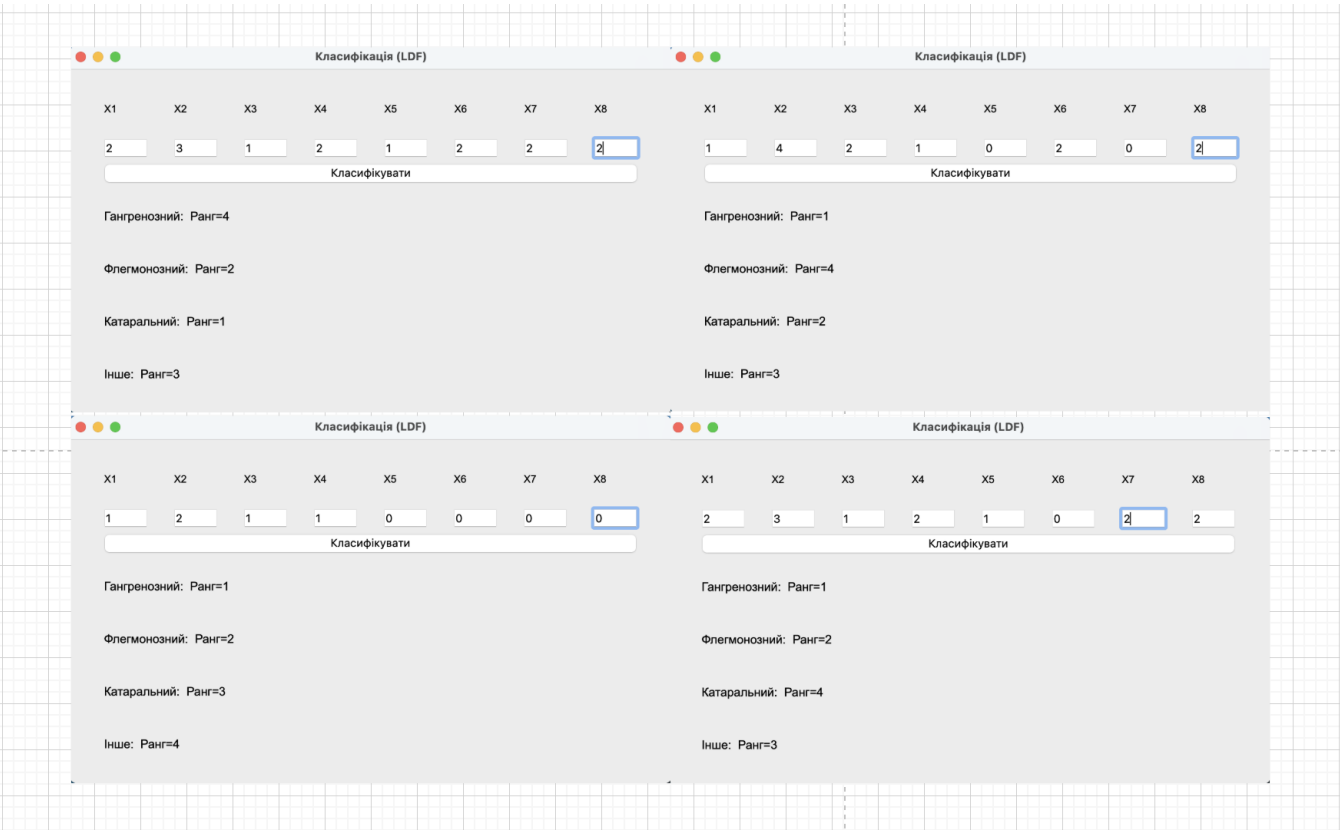


Рисунок 4.2 – Рузультати імітаційного моделювання

4.5 Висновки до четвертого розділу

1. Для імітаційного моделювання обрано як навчальні та тестові вибірки 103 показники медичного діагностування захворювань на апендицит із доведених 4-х діагнозів для 8 симптомів. При цьому задіяно 4 визначені ЛДФ з відповідними коефіцієнтами для кожного із симптомів

2. Розроблено програму моделювання процесу класифікації з ранжуванням результатів на мові програмування Python, що дозволяє визначити не тільки ЛДФ з максимальним значенням, але й сформувати ранги всіх ЛДФ.

3. Аналіз результатів імітаційного моделювання показав, що такий показник точності діагностування за вирішальним правилом дискримінантного аналізу, як чутливість, яка має такі значення для кожного з 4-х діагнозів: 1-й – 71.43%, 2-й – 44.00%, 3-й – 50.00%, 4-й – 100.00%.

4. Результати імітаційного моделювання процесу класифікації об’єктів з формуванням рангів підтверджують правильність алгоритму функціонування нейромережного класифікатора, що свідчить про доцільність його застосування у складі підсистеми підтримки прийняття рішень в інтелектуальних системах медичного діагностування.

ВИСНОВКИ

1. Аналіз застосувань методів та засобів штучного інтелекту показав, що існує необхідність у нових підходах до реалізації функціональних та структурних базисів побудови класифікаторів об’єктів, які входять до складу підсистем підтримки прийняття рішень в експертних системах різного призначення.
2. Удосконалено матричний метод оброблення елементів ЛДФ з урахуванням коефіцієнтів зміщення в матричному вигляді, що дозволяє реалізувати відомий метод різницево-зрізового оброблення матриці елементів ЛДФ з паралельним обробленням по стовпцях та рядках систолічної матриці.
3. Запропоновано альтернативний підхід до реалізації механізму конкуренції типу WTA у максимізаторі на базі паралельного задіяння швидкісної операції декремента, що прискорює обчислювальний процес визначення максимальної за значенням ЛДФ.
4. Запропоновано використання операції інкремента для реалізації ранжування результатів класифікації з одночасним застосуванням парної операції декремента/інкремента у кожному циклі оброблення, що дозволяє розширити функціональні можливості класифікатора об’єктів.
5. Запропоновано структуру та принцип функціонування нейроподібного класифікатора з позрізовим обробленням елементів ЛДФ, в якому враховано наявність коефіцієнтів зміщення у матричному вигляді, що дозволяє використати матричне оброблення елементів ЛДФ на систолічній матриці.
6. Запропоновано структуру та принцип функціонування нейроподібного класифікатора з ранжуванням результатів класифікації, в якому реалізованно механізм конкуренції серед ЛДФ за принципом альтернативного сортування,

що дозволяє задіяти швидкісні операції декремента/інкремента відповідно до сортуваня/ранжування елементів числового масиву.

7. Апаратна реалізація запропонованих структур класифікаторів об’єктів обох типів на ПЛІС представляє інтерес як апаратний прискорювач для адаптивних систем класифікації із спрацюванням у реальному часі, зокрема у блоці сприйняття та аналізу зображень об’єктів у системах автономного керування мобільних роботів.

8. Розроблено програму моделювання процесу класифікації з ранжуванням результатів на мові програмування Python, що дозволяє визначити не тільки ЛДФ з максимальним значенням, але й сформувати ранги всіх ЛДФ. Це, у свою чергу, значно полегшить спеціалісту прийняти остаточне рішення за рахунок аналізу найбільшого та найближчих до нього сформованих рангів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. R. Sedgewick, *Algorithms in C++: Part 1-4. Fundamentals, Data Structures, Sorting, Searching.* 3rd ed.. Addison-Wesley, Longman, 1999.
2. J. D. Knuth, *The Art of Computer Programming, V.3. Sorting and Searching*, 2nd ed. Addison - Wesley Professional, 1998. 800 p.
3. M.T. Jones, *Al Application Programming*. Charles River Media, 2003.
4. S. Osowski, Sieci neuronowe do przetwarzania informacji. Poland: Oficiyna Wydawnicza Politechniki Warszawskiej, 2013. 490 s.
5. R. Callan, *The Essence of Neural Networks*. Prentice Hall Europe, NY, 1999. 232c.
6. S. Haykin, *Neural Networks and Learning Machines.* Upper Saddle River, *2009.*
7. T. Kohonen, *Self-Organization and Associative Memory*. Springer Berlin Heidelberg, 1989.
8. Е.М. Куссуль, *Асоціативні нейроподібні структури*. Київ, Україна: Наукова думка, 1990.
9. І.І. Цмоць, *Інформаційні технології та спеціалізовані засоби обробки сигналів і зображень в реальному часі*. Львів, Україна: Видавництво УАД, 2005. 228 с.
10. В. Вуйцік, О.З. Готра, та В.В. Григор'єв, *Експертні системи.* Львів, Україна: Ліга-Прес, 2006. 209 с.
11. Р.Ф. Ситник, *Системи прийняття рішень*. Київ, Україна: КНЕУ, 2004. 614 с.
12. Г. М. Гнатієнко, та В.Є. Снитюк, *Експертні технології прийняття рішень.* Київ, Україна: ТОВ "Маклаут", 2008.
13. T. Martyunik, B. Krupikivskyi, L. Kupenshtein, and V. Lukichov, "Neural network models of heteroassociative memory for the classification tasks*"*,

Radioelectronics and Computer Systems, №1 (102), с. 104-111. 2022. doi:10.32620/reks.2022.2.09.

14. Т. Б. Мартинюк, А. В. Кожем’яко, Л. М. Куперштейн, та О. С. Безкревний, “Операційно-елементний базис для інтелектуальних систем”, *Вісник Хмельницького національного університету. Технічні науки*, №6, с. 197-201. 2019.
15. І.Г. Цмоць, В.М. Теслюк, Ю.В. Опотяк, Р.В. Парцей, та Р.В. Зінько, “Базова архітектура мобільної робототехнічної платформи з інтелектуальною системою управління рухом і захистом передачі даних”, *Український журнал інформаційних технологій*, №2, Т.3, с. 74-80. 2021. https://doi.org/10.23939/ujit2021.02.074
16. О.Д. Азаров, Т.Б. Мартинюк, А.В. Кожем’яко, *Теоретичні та реалізаційні моделі комп'ютерних інтелектуальних систем.* Монографія. Вінниця, Україна: ВНТУ, 2024. 130с.
17. Т.Б. Мартинюк, Б.І. Круківський та О.А. М’якішев, “Особливості моделей нейроподібного класифікатора для розпізнавання обєктів”, *Вісник Вінницького національного політехнічного інституту*, №4(163), с. 53-63. 2022. https://doi.org/10.31649/1997-9266-2022-163-4-56-63.
18. D. Smith, J. Hall and K. Miyke, “The CAM 2000 Chip Architecture”. Rutgers University. [Електронний ресурс], Режим доступу. https://www.rutgers.edu/pub/technical-reports.
19. N.F. Kirichenko, A.M. Reznik, & S.P. Shchetenyuk, “Matrix pseudoinvertion in the Problem of Design of Associative Memory”, *Sybernetic and Systems Analysis*, №33, pp. 308-316. 2001.

20. Т.Б. Мартинюк, та Б.І. Круківський, “Модель паралельного сортувальника для асоціативного процесора”, *Вісник Вінницького політехнічного інституту*, №5, с. 49-55.2020. https://doi.org/10.31649/1997-9266-2020-152-5-49-55.
21. А.С. Васюра, Т.Б.Мартинюк, та Л.М. Куперштейн, *Методи та засоби нейроподібної обробки даних для систем керування*. Монографія. Вінниця, Україна: УНІВЕРСУМ - Вінниця, 2008. 175с.
22. P. Corke, *Robotic, Vision and Control*. Springer International Publishing. 2017.
23. I. Arents, & M. Greitans, “Smart industrial Robot Control Trends, Challenges and Opportunities within Manufacturing”, *Applies Sciences*, vol.12, issue 2.p.937. 2022.
24. Y. Pategera, “Main strategy for autonomous robotic controller design”, *Радіоелектроніка та інформатика*, вип. 4,с. 35-41. 2011.
25. В.Ф. Бордаченко, О.К. Колесницький, та С.А. Василецький, *Таймерні нейронні елементи та структури*. Монографія. Вінниця, Україна: УНІВЕРСУМ - Вінниця, 2004. 126с.
26. Т.Б. Мартинюк, А.В. Кожем’яко, Н.В. Фофанова, та О.М. Наконечний, “Адаптивний суматор для системи керування роботом”, *Оптико-електронні інформаційно-енергетичні технології*, №2(10), с. 96-104. 2005.
27. V.I. Osinsky, "Information conception of image perceptron at solid-state lighting”, *Semiconductor Physics, Quantum Electronics & Optoelectronics*, V.10, №3, p.30-43.2007.
28. О.В. Нестеренко, *Інтелектуальні системи підтримки прийняття рішень*. Навч. посібник. Київ, Українка: Нац. акад. управління, 2016.
29. О.Г. Аврунін, Є.В. Бодянський, М.В. Калашник, В.В. Семенець, та В.О. Філатов, *Сучасні інтелектуальні технології медичної діагностики*. Харків, Україна: ХНУРЕ, 2018.

30. Rangaraj M.Rangayyan, *Biomedical Signal Analysis*. John Wiley & Sous, 2015.

31. J. Brownlee, “Linear Discriminant Analysis for Machine Learning”. Available at: https://machinelearningmastery.com/linear-discriminant-analysis-for-learning

32. T.B. Martyniuk, L.M. Kupershtein, A.V. Medvid, A.V. Kozhemiako, and others, “Application of Discriminants Analysis Methods in Medical Diagnostics”, *Optical Fibers and Their Applications*, Proc. of SPIE 8698, 86980, 2012.

33. Т.Б. Мартинюк, та Д.О. Каташинський, “Особливості асоціативного оброблення даних в інтелектуальних системах”, *Оптико-електронні інформаційно-енергетичні технології,* №1(49), с.44-52.2025. DOI: https://doi.org/10.31649/1681-7893-2025-49-1-44-52

34. Т. Б. Мартинюк, *Рекурсивні алгоритми багатооперандної обробки інформації*. Монографія. Вінниця: «Універсум» Вінниця, 2000. – 216с.

35. Т. Б. Мартинюк, та В.В. Хом’юк, Методи та засоби паралельних перетворень векторних масивів даних: Монографія. Вінниця, Україна: УНІВЕРСУМ-Вінниця, 2005.

36. Т.Б. Мартинюк, Д.О. Каташинський, М.В. Микитюк, та М.О. Зайцев, “Особливості обчислювальних процесів на базі SM-перетворення”, Оптико-електронні інформаційно-енергетичні технології, №2(44), с.32-37. 2022. https://doi.org/10.31649/1681-7893-2022-44-2-32-37

37. І. Г. Цмоць, В. Я. Антонів, та В. О. Парубчак, “Паралельно-вертикальне сортування одновимірних даних методом злиття з використанням підрахунку”, *Збірник наукових праць. Інститут проблем моделювання в енергетиці,* Вип. 68, с. 92-100, 2013.

38. І. Г. Цмоць, та В. Я. Антонів, “Алгоритми та паралельні структури сортування даних методом вставки”, *Науковий вісник НЛТУ,* Вип. 261, с. 340-350, 2016.

39. W.K. Pratt, *Digital Image Processing:* PIKS Scientific Inside. 2006. – 480с.

DOI:10.1002/0470097434

40. Д.О. Каташинський, та Т.Б. Мартинюк, “Функціональні можливості оброблення даних на база SM-перетворення”, *Матеріали Всеукраїнської науково-практичної інтернет-конференції* «Молодь в науці: дослідження, проблеми, перспективи», Вінниця: ВНТУ, 2023. https://conferences.vntu.edu.ua/index.php/mn /mn2023/paper/view/17047
41. P.Jackson, *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc. 75 Arlington Street, Suite 300 Boston, 1998.
42. *Методи класифікації інформації*. [Електронний ресурс]. Режим доступу: http://helpstudenty.at.ua/publ/katalog_dlja_studenta/informacionnye_sistemy/metodi_klasifikaciji _informaciji/25-1-0-741. Дата звер-тання: Трав. 20, 2024.
43. С. М. Злепко, О. Л. Лаугс, К. С. Навроцька та С. В. Тимчик, «Автоматизований медичний комплекс для оцінювання здоров’я студентів», *Патент України A61B 5/00. № 101608 МПК(2008)*, 25.09.2015.
44. Т. Б. Мартинюк, та Я. В. Запетрук, «Нейромережевий підхід до медичної експрес-діагностики», *Вісник Вінницького політехнічного інституту*, № 6, с.37-44.2019,
45. Т. Б. Мартинюк, та А. В. Маслій, «Аналіз обчислювального процесу в нейромережевому класифікаторі», *Інформаційні технології та комп’ютерна інженерія*, № 3, с. 55-60. 2017.
46. L. Kupershtein, T. Martyniuk, O. Voitovych, M. Krentsin, “Neural network approach in the stroke diagnosis”, *Proc. of the 2016 IEEE 1st internatinal Conference on Data Mining and Processing, DSMP 1016, 23-27 Ang.*, 2016, pp, 138-141. DOI:10.1109/DSMP.2016.7583525
47. L. Kupershtein, T. Martyniuk, M. Krentsin, A. Kozhemiako, and other., “Neural expert decision support system for stroke diagnosis”, *Proc. of Conference*

Photonics Application in Astronomy, Communications, Industry, and High-Energy Physics Experiments, 2017, vol. 10445, no 1044531. DOI:10.1117/12.2280956

48. Лінійка світлодіодів. [Електронний ресурс]. Режим доступу: http://arduino.ua/prod1624-poloska-s-8-rgb-svetodiodami . Дата звертання: Квіт. 15, 2025.
49. Т. Б. Мартинюк, та Я. В. Запетрук, «Реалізаційні моделі базових вузлів нейромережевого класифікатора», *на Шостій міжнар. наук.-техн. конф. Оптоелектронні інформаційні технології «Фотоніка ОДС* - 2018», Вінниця, 2018, с. 27.
50. Г. Т. Олійник, І. В. Степанушко, та І. Б. Трегубенко, «Побудова класифікаторів в задачах біометричної ідентифікації та аутентифікації користувачів,» *Вісник Черкаського державного технічного університету*, № 1, с. 37-40, 2009.
51. А. С. Васюра, Т. Б. Мартинюк, та Л. М. Куперштейн, *Методи та засоби нейроподібної обробки даних для систем керування*. Вінниця, Україна: Универсум-Вінниця, 2008.
52. Т.Б. Мартинюк, М.Г. Тарановський, та Я.В. Запетрук, «Структурні особливості нейромережевого класифікатора», *Вісник Вінницького політехнічного інституту*, № 1, с.46-52.2020
53. ТБ Мартинюк, ДО Каташинський, МВ Микитюк, та МО Зайцев, «Особливості обчислювальних процесів на базі SM–перетворення», *Інформаційно-енергетичні технології*, № 2, с. 32-37. 2022.
54. R. Sedgewick, *Algorithmsin C++. Part 5: Graph Algorithms*. Pearson Education, inc., 2002.
55. A. P. Tsarev, *Algorithmic Models and Structures of High-Performance Processors of Digital Signal Processing*. Szczecin, Poland: Informa, 2000.

56. Я. М. Николайчук, *Теорія джерел інформації*. Тернопіль, Україна: ТзОВ “Термо-Граф”, 2010.
57. Т.Б. Мартинюк, А.В. Кожем’яко, І.В. Булига, та Д.О. Каташинський, «Графічна модель пралельної асоціативної обробки числових даних», *Наукові праці ВНТУ*, №4, с.1-5.2024. https://praci.vntu.edu.ua/index.php/praci/article/view/769/741
58. І.В. Булига, Т.Б. Мартинюк, та Д.О. Каташинський, «Особливості асоціативної обробки числових даних за різницевими зрізами», *Молодь в науці: дослідження, проблеми, перспективи*. 2025.
59. Т. Б. Мартинюк, А. В. Кожем’яко, Б. І. Круківський, та А. Г. Буда, «Асоціативні операції на базі різнцево-зрізової обробки даних», *Вісник Хмельницького національного університету. Технічні науки*, №4, с. 159-163, 2022. doi: 10.31891/2307 – 5732 – 2022-311-4-159-163
60. Т.Б. Мартинюк, Л. М. Куперштейн, та М.Д. Кренцін, «Особливості процесу класифікації об’єктів на базі дискримінантних функцій», *Математичні машини і системи,* №3, с, 81-87. 2021.
61. Т.Б. Мартинюк, Л. В. Крупельницькій, М.В. Микитюк, та М.О. Зайцев, «Систолічна структура матричного обчислювача для класифікатора об’єктів», *Electronic modeling*, V.43, №3, с, 36-46. 2021. http://ir.lib.vntu.edu.ua//handle/123456789/34183
62. Т.Б. Мартинюк, та А.В. Кожем’яко, *Систолічні структури для багатоопперантної обробки векторних даних*. Вінниця, Україна: УНІВЕРСУМ – Вінниця, 2008.
63. Т. Б. Мартинюк, та Б.І. Круківський, «Класифікаційний аналіз методів сортування», *Вісник Вінницького політехнічного інстетуту*, №3, с, 77-83. 2023. https://doi.org/10.31649/1997-9266-2023-168-3-77-83

64. Т.Б. Мартинюк, А.В. Кожем’яко, Л.В. Крупельницький, О.М. Перебейніс, О.С. Безкревний, «Реалізаційні моделі матричного обчислювача для класифікатора біомедичних даних». *Інформаційні технології та комп’ютерна інженерія*, №2(36), с, 43-51. 2016.

65. Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, «Особливості обчислювального процесу у матричному класифікаторі об’єктів», *Наукові праці ВНТУ*, №3, 2025.

66. Т.Б. Мартинюк, А.В. Медвідь, та О.М. Гуцол, «Моделювання процесу ранжування значень дискримінантних функцій», *Вісник Вінницького політехнічного інстетуту*, №5, с. 74-80. 2013.

67. Т.Б. Мартинюк, Л.В. Крупельницький, та Б.І. Круківський, «Регулярна обчислювальна структура для ранжування даних», *Інформаційні технології та комп’ютерна інженерія,* №3(52), с. 70-76. 2021. https://doi.org/10.31649/1999-9941-2021-52-3-70-76

68. Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, «Особливості процесу класифікації у контексті медичного діагностування», *Наукові праці ВНТУ*, №1 с. 1-6. 2025.

69. Т.Б. Мартинюк, та Я.В. Запетрук, «Класифікатор», *Патент України G06G7/00, №138749 МПК (2019.01)*, 10.12.2019.

70. Т.Б. Мартинюк, І.Д. Дорощенков, та О.М. Гуцол, «Схемотехнічні рішення базових блоків для класифікатора образів», *Вісник Вінницького політехнічного інстетуту*, №3, с. 132-141. 2012.

71. Т. Б. Мартинюк, А. В. Кожем’яко, Д. О. Каташинський, та І. В. Булига, «Структурні особливості нейроподібного класифікатора об’єктів», *Наукові праці ВНТУ*, №4, с. 1-7. 2023. https://doi.org/10.31649/2307-5376-2023-4-1-7

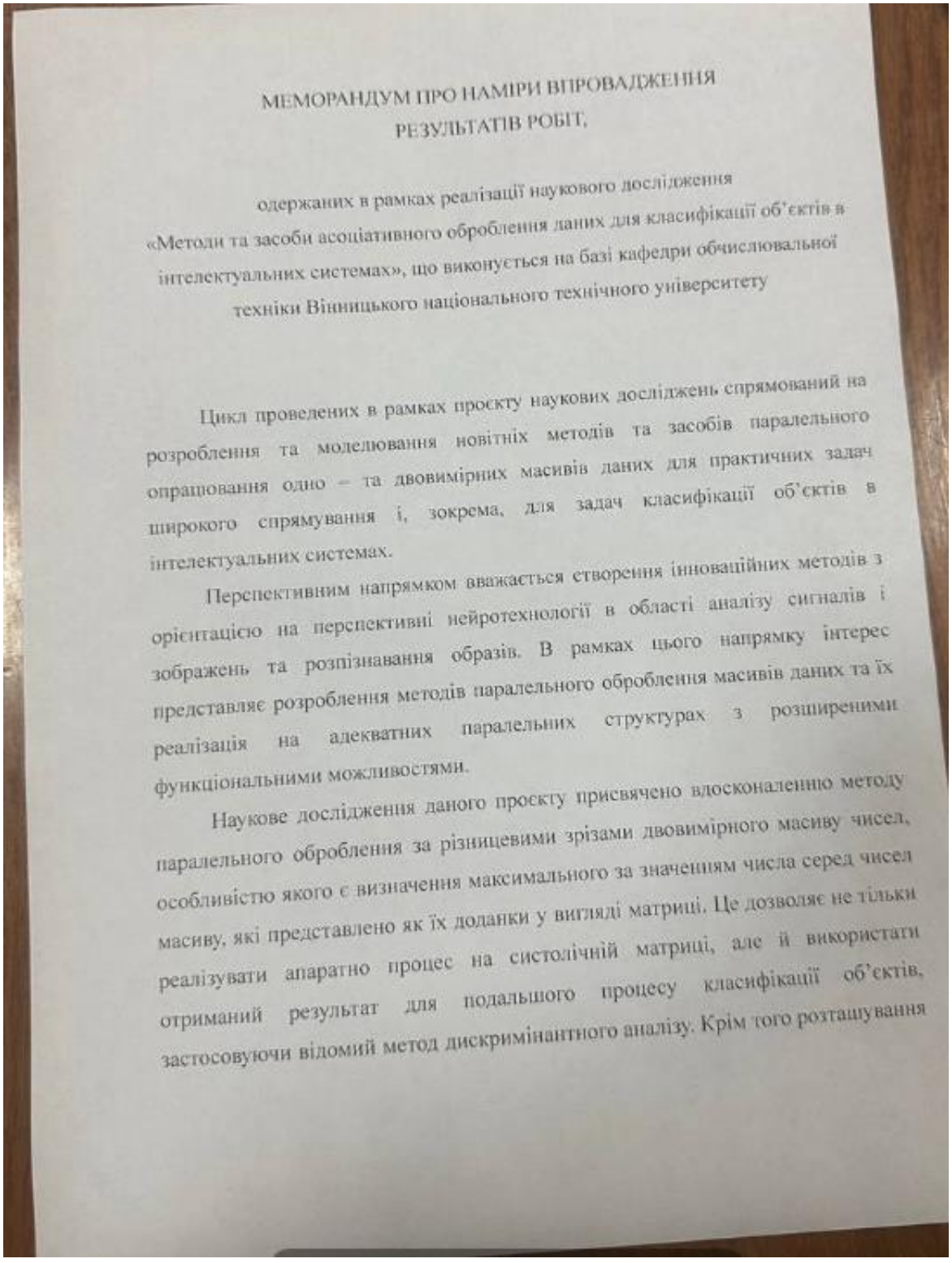
72. Д.О. Каташинський, Т. Б. Мартинюк, та І. В. Булига, «Модель двовимірного нейроподібного класифікатора», *Наукові праці ВНТУ*, №4, 2024.

73. С.О. Кравчук, та В.О. Шонін, *Основи комп'ютерної техніки*. *Компоненти, системи, мережі*. Навч посібник. Київ, Україна: ІВЦ «Видавництво політехніка», 2005.
74. Т.Б. Мартинюк, А.В. Кожем'яко, та Л.М. Куперштейн, «Властивості матриці елементів дискримінантних функцій», *Вісник Хмельницького національного університету. Технічні науки*, №3, с, 202-204. 2015.
75. D. M. Harris, L. H. Morgan, *Digital Design and Computer Architecture*, 3-d Edition. Kaufmann Publishers, 2015.
76. L. Tsmots, V. Teslyuk, & I. Vavruk, "Hardware and software tools for motion control of mobile robotic system", in 12th *Internatinal Conference "The Experience of Desining and Application of CAD systems in Microelectronics"*, CADSM 2013, 368 p.
77. A. Medina-Santiago, I. A. Morales-Rosales, C.A. Hernandes-Gracidas and others, "Reactive Obstacle – Avoidance Systems for Wheeled Mobile Robots Based on Artificial Intelligance", *Applied Sinece*, *11*(14), 6468. 2021 https://doi.org/10.3390/app11146468
78. L. Yang, J.Qi, D. Song and others, "Survey of robot 3D path planning algorithms", *J Control Sci Eng*, 5p. 2016 https://doi.org/10.1155/2016/7426913

ДОДАТКИ

Додаток А

Акти впропадження результатів та наміри про впровадження



МЕМОРАНДУМ ПРО НАМІРИ ВПРОВАДЖЕННЯ
РЕЗУЛЬТАТІВ РОБІТ,

одержаних в рамках реалізації наукового дослідження
«Методи та засоби асоціативного оброблення даних для класифікації об'єктів в інтелектуальних системах», що виконується на базі кафедри обчислювальної техніки Вінницького національного технічного університету

Цикл проведених в рамках проєкту наукових досліджень спрямований на розроблення та моделювання новітніх методів та засобів паралельного опрацювання одно – та двовимірних масивів даних для практичних задач широкого спрямування і, зокрема, для задач класифікації об'єктів в інтелектуальних системах.

Перспективним напрямком вважається створення інноваційних методів з орієнтацією на перспективні нейротехнології в області аналізу сигналів і зображень та розпізнавання образів. В рамках цього напрямку інтерес представляє розроблення методів паралельного оброблення масивів даних та їх реалізація на адекватних паралельних структурах з розширеними функціональними можливостями.

Наукове дослідження даного проєкту присвячено вдосконаленню методу паралельного оброблення за різницевими зрізами двовимірного масиву чисел, особливістю якого є визначення максимального за значенням числа серед чисел масиву, які представлено як їх доданки у вигляді матриці. Це дозволяє не тільки реалізувати апаратно процес на систолічній матриці, але й використати отриманий результат для подальшого процесу класифікації об'єктів, застосовуючи відомий метод дискримінантного аналізу. Крім того розташування

спроєктованої систолічної матриці у мікросхемі ПЛІС XC6SIX45T сімейства Spartan-6 показало спрацювання у мілісекундному діапазоні.

Другий варіант паралельного оброблення стосується одновимірного масиву чисел і також зорієнтований на отримання результату класифікації об'єктів. В цьому випадку застосовано паралельне оброблення векторного числового масиву за операцією декремента, що дозволило використати паряну операцію інкремента для формування рангів результатів класифікації. В результаті розширюються функціональні можливості класифікатора об'єктів. Крім того, по-перше, використання операції декремента представляє собою альтернативний підхід до механізму конкуренції нейронів WTA (Winner Takes All), а по-друге, структура такого нейроподібного класифікатора має високу регулярність через можливість використання лінійки реверсивних лічильників.

Сукупність результатів, отриманих у проєкті, можуть бути основою для розвитку методів та нейротехнологій в інтелектуальних системах, зокрема, для медичного діагностування у підсистемі підтримки прийняття рішень в експертних системах та у системах автономного керування мобільного робота через орієнтацію на перспективну елементну базу – ПЛІС.

Результати досліджень пройшли апробацію на X міжнародній науково-практичній конференції з оптико-електронних інформаційних технологій «Фотоніка – ODS 2025» (м. Вінниця) з подальшою можливістю публікації у наукометричній базі Scopus.

Директор ТОВ "Елефантелаб" Оксана БЕЛОКОНЬ

Додаток А

Акти впровадження результатів та наміри про впровадження

Проректор з науково-педагогічної роботи
та організації освітнього процесу ВНТУ
_____ Олександр ПЕТРОВ
09 2025р.

АКТ

про впровадження результатів в рамках виконання дисертаційних досліджень

КАТАШИНСЬКОГО ДМИТРА ОЛЕКСАНДРОВИЧА

за темою «Методи та засоби асоціативного оброблення даних для класифікації об'єктів в інтелектуальних системах», яка подана на здобуття ступеня доктора філософії з галузі знань 12 «Інформаційні технології» за спеціальністю 123 «Комп'ютерна інженерія»

Комісія у складі декана факультету інформаційних технологій та комп'ютерної інженерії, к.пед.н., доцента Кирилащук Світлани Анатоліївни, зав. кафедри обчислювальної техніки, д.т.н., професора Азарова Олексія Дмитровича та к.т.н., доцента кафедри обчислювальної техніки Богомолова Сергія Віталійовича склали цей акт про те, що протягом 2022-2025 років на базі кафедри обчислювальної техніки Вінницького національного технічного університету впроваджено результати дисертаційної роботи Каташинського Дмитра Олександровича у навчальному процесі, а саме запропоновано нові технічні рішення стосовно побудови структур нейроподібних класифікаторів

об'єктів із задіянням методів реалізації асоціативного оброблення масивів даних на базі альтернативного сортування із застосуванням швидкісних операцій декремента/інкремента.

Результати досліджень дозволили розширити функціональні можливості нейроподібних класифікаторів на базі дискримінантного аналізу даних за рахунок можливості ранжування результатів класифікації. Такий підхід особливо важливий при медичному діагностуванні захворювань, оскільки дозволяє, використовуючи отримані ранги діагнозів, проаналізувати найбільш вірогідні результати діагностування. Програмну реалізацію запропонованого методу класифікації з ранжуванням зорієнтовано на використання у підсистемі підтримки прийняття рішень у складі медичної експертної системи.

Результати досліджень впроваджено у навчальний процес кафедри обчислювальної техніки ВНТУ у дисципліни «Інтелектуальні системи, технології та нейрокомп'ютери» та «Методи штучного інтелекту та нейромережі» для студентів, які навчаються за освітньою програмою «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія».

Голова комісії: _____ Світлана КИРИЛАЩУК

Члени: _____ Олексій АЗАРОВ

_____ Сергій БОГОМОЛОВ

Додаток Б

Вихідні дані для імітаційного моделювання

```
first_group =  [
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 2,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 0,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 2,'x3': 3,'x4': 1,'x5': 1,'x6': 0,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 0 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 3,'x5': 0,'x6': 0,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 2,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 0,'x8': 2 },
    { 'x1': 2, 'x2': 4,'x3': 1,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 1, 'x2': 2,'x3': 2,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 1,'x3': 1,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 0 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 3,'x5': 0,'x6': 2,'x7': 0,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 2,'x5': 1,'x6': 0,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 4,'x3': 2,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 2,'x3': 1,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 3,'x4': 2,'x5': 1,'x6': 2,'x7': 0,'x8': 2 },
    { 'x1': 1, 'x2': 1,'x3': 2,'x4': 2,'x5': 0,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 1,'x3': 1,'x4': 3,'x5': 1,'x6': 0,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 3,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 2,'x3': 1,'x4': 2,'x5': 1,'x6': 2,'x7': 0,'x8': 2 },
```

```
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 2,'x5': 0,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 2,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 3,'x5': 1,'x6': 2,'x7': 2,'x8': 2 },
    { 'x1': 2, 'x2': 3,'x3': 1,'x4': 2,'x5': 1,'x6': 2,'x7': 2,'x8': 0 },
]
second_group =  [
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 2, 'x4': 1, 'x5': 0, 'x6': 2, 'x7': 0, 'x8': 2, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 3, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 2, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 3, 'x5': 0, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 3, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 3, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 2, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 3, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 2, 'x4': 1, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
```

```
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 3, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 3, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
]
third_group =  [
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 2, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 3, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 1, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 2, },
    { 'x1': 2, 'x2': 1, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 3, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 4, 'x3': 2, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 2, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 3, 'x5': 0, 'x6': 0, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
```

```
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 2, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 2, },
    { 'x1': 1, 'x2': 4, 'x3': 2, 'x4': 1, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 2, 'x2': 3, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 2, 'x8': 0, },
]
fourth_group =  [
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 1, 'x3': 2, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 1, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 1, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 1, 'x3': 2, 'x4': 1, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 1, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 2, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 1, 'x3': 1, 'x4': 2, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 2, },
    { 'x1': 1, 'x2': 1, 'x3': 2, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 2, 'x8': 0, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 3, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 1, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 4, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
```

```
    { 'x1': 1, 'x2': 1, 'x3': 1, 'x4': 2, 'x5': 1, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 1, 'x3': 2, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 2, 'x2': 1, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
    { 'x1': 1, 'x2': 2, 'x3': 1, 'x4': 1, 'x5': 0, 'x6': 0, 'x7': 0, 'x8': 0, },
]
```

C

Додаток В

Лістинг програми для імітаційного моделювання

```
# Import pre-defined groups of data from external modules.
# Each group is expected to be a list of dictionaries with numerical features (x1, x2, ..., x8).
from groups.first import first_group
from groups.second import second_group
from groups.third import third_group
from groups.fourth import fourth_group

# Dictionary that maps a group index (string key) to the corresponding dataset.
# This allows iteration through all groups in a unified way.
learn_info = {
    '1': first_group,
    '2': second_group,
    '3': third_group,
    '4': fourth_group,
}


def check_ldf(group_index, row_index, x1, x2, x3, x4, x6, x7, x8):
    """
    Calculate four discriminant function values (ldf1, ldf2, ldf3, ldf4)
    based on the provided input features (x1, x2, x3, x4, x6, x7, x8).
```

```
    Args:
        group_index (str): Identifier of the group the row belongs to.
        row_index (int): Row position within the group.
        x1..x8 (float): Feature values used in linear discriminant function formulas.

    Returns:
        list of dict: Each element contains the LDF value, its name,
        and ranking based on ascending order.
    """

    # Compute four different discriminant functions as weighted linear combinations of inputs.
    # Each function has its own coefficients.
    ldf1 = round(-63.0  + 9.8 * x1 + 3.6 * x2 + 7.8 * x3 + 5.2 * x4 + 14.3 * x6 + 11.8 * x7 + 11.3 * x8, 2)
    ldf2 = round(-57.4  + 8.3 * x1 + 4.9 * x2 + 6.2 * x3 + 4.3 * x4 + 13.5 * x6 + 11.7 * x7 + 10.6 * x8, 2)
    ldf3 = round(-49.6  + 9.4 * x1 + 4.7 * x2 + 5.5 * x3 + 3.0 * x4 + 12.3 * x6 + 12.0 * x7 +  8.3 * x8, 2)
    ldf4 = round(-23.0  + 6.3 * x1 + 2.5 * x2 + 5.3 * x3 + 2.8 * x4 +  7.8 * x6 +  7.0 * x7 +  5.8 * x8, 2)

    # Store LDF results in a list of dictionaries.
    ldfs = [
        {"value": ldf1, "name": "ldf1"},
        {"value": ldf2, "name": "ldf2"},
        {"value": ldf3, "name": "ldf3"},
```

```
        {"value": ldf4, "name": "ldf4"},
    ]

    # Sort LDF values in ascending order (lowest value first).
    # Change reverse=True if you want highest-first ordering.
    sorted_ldfs = sorted(ldfs, key=lambda item: item["value"], reverse=False)

    # Create a new list with ranking information.
    ranked = []
    for i, item in enumerate(sorted_ldfs):
        ranked.append({
            "ldfValue": item["value"],  # Numerical LDF value
            "ldfName": item["name"],    # Which LDF it corresponds to
            "rang": i + 1               # Rank (1 = lowest value, 4 = highest value)
        })

    return ranked



# Will store the results for all groups and rows
result = []

# Iterate through each group and each row within the group
for group_index, group_data in learn_info.items():
    for row_index, item in enumerate(group_data):
        # Extract feature values from the current row (dictionary)
        x1 = item["x1"]
```

```
        x2 = item["x2"]
        x3 = item["x3"]
        x4 = item["x4"]
        x6 = item["x6"]
        x7 = item["x7"]
        x8 = item["x8"]

        # Compute LDFs and rankings for this row
        handled_ldf_arr = check_ldf(group_index, row_index, x1, x2, x3, x4, x6, x7, x8)

        # Save results into global list
        result.append(handled_ldf_arr)

        # Print row details for debugging/logging purposes
        print(f"x1: {x1}, x2: {x2}, x3: {x3}, x4: {x4}, x6: {x6}, x7: {x7}, x8: {x8}")
        print(
            f"groupIndex: {group_index} rowIndex: {row_index} "
            f"{sorted(handled_ldf_arr, key=lambda item: item['ldfValue'],
reverse=True)}\n"
        )

result = []

# Iterate through each group and each row within that group
for group_index, group_data in learn_info.items():
    max_ldf_count = 0  # Counter for how many times ldf{group_index} is the highest
```

```
    for row_index, item in enumerate(group_data):
        # Extract feature values from the current item (dictionary)
        x1 = item["x1"]
        x2 = item["x2"]
        x3 = item["x3"]
        x4 = item["x4"]
        x6 = item["x6"]
        x7 = item["x7"]
        x8 = item["x8"]

        # Compute the LDF values and their ranking
        handled_ldf_arr = check_ldf(group_index, row_index, x1, x2, x3, x4, x6, x7, x8)
        result.append(handled_ldf_arr)

        # Sort LDFs by descending value to find the maximum one
        sorted_array = sorted(handled_ldf_arr, key=lambda item: item["ldfValue"],
reverse=True)
        max_ldf = sorted_array[0]

        # Check if the max LDF matches the expected one for this group
        if max_ldf["ldfName"] == f"ldf{group_index}":
            max_ldf_count += 1

    # Output the proportion of rows where ldf{group_index} was the top-ranked
    print(f"GROUP_{group_index} {(max_ldf_count / len(group_data)) * 100:.2f}%")
```

```
from PyQt6.QtWidgets import (
    QApplication, QWidget, QLabel, QGridLayout,
    QLineEdit, QPushButton, QMessageBox
)
from PyQt6.QtGui import QFont
import sys


def calculate_ldf_values(x1, x2, x3, x4, x5, x6, x7, x8):
    # LDF formulas
    ldf1 = round(-63.0  + 9.8 * x1 + 3.6 * x2 + 7.8 * x3 + 5.2 * x4 + 14.3 * x6 + 11.8 *
x7 + 11.3 * x8, 2)
    ldf2 = round(-57.4  + 8.3 * x1 + 4.9 * x2 + 6.2 * x3 + 4.3 * x4 + 13.5 * x6 + 11.7 *
x7 + 10.6 * x8, 2)
    ldf3 = round(-49.6  + 9.4 * x1 + 4.7 * x2 + 5.5 * x3 + 3.0 * x4 + 12.3 * x6 + 12.0 *
x7 +  8.3 * x8, 2)
    ldf4 = round(-23.0  + 6.3 * x1 + 2.5 * x2 + 5.3 * x3 + 2.8 * x4 +  7.8 * x6 +  7.0 *
x7 +  5.8 * x8, 2)

    # Store values
    result = [
        {"name": "Гангренозний", "code": 1, "value": ldf1},
        {"name": "Флегмонозний", "code": 2, "value": ldf2},
        {"name": "Катаральний",  "code": 3, "value": ldf3},
        {"name": "Інше",         "code": 4, "value": ldf4},
    ]

    # Sort by LDF descending (max → min)
    result_sorted = sorted(result, key=lambda x: x["value"], reverse=False)

    # Add ranks
    for i, item in enumerate(result_sorted):
        item["rank"] = i + 1

    return result_sorted


class AppendicitisApp(QWidget):
    def __init__(self):
        super().__init__()
```

```
        self.setWindowTitle("Класифікація (LDF)")
        self.setGeometry(300, 200, 700, 400)

        layout = QGridLayout()

        labels = ["X1","X2","X3","X4","X5","X6","X7","X8"]
        self.inputs = []

        # Input rows
        for i, lbl in enumerate(labels):
            label = QLabel(lbl)
            label.setFont(QFont("Arial", 12))
            layout.addWidget(label, 0, i)

            inp = QLineEdit()
            inp.setFixedWidth(50)
            layout.addWidget(inp, 1, i)

            self.inputs.append(inp)

        # Button
        btn = QPushButton("Класифікувати")
        btn.clicked.connect(self.on_classify)
        layout.addWidget(btn, 2, 0, 1, 8)

        # Result block labels
        self.result_labels = []
        diagnosis_titles = ["Гангренозний", "Флегмонозний", "Катаральний", "Інше"]

        for i, title in enumerate(diagnosis_titles):
            lbl = QLabel(f"{title}:  ---")
            lbl.setFont(QFont("Arial", 14))
            layout.addWidget(lbl, 3 + i, 0, 1, 8)
            self.result_labels.append(lbl)

        self.setLayout(layout)

    def on_classify(self):
        # Read values
```

```
        try:
            vals = [float(i.text()) for i in self.inputs]
        except ValueError:
            QMessageBox.critical(self, "Error", "All X values must be numbers!")
            return

        x1, x2, x3, x4, x5, x6, x7, x8 = vals

        # Compute results
        results = calculate_ldf_values(x1, x2, x3, x4, x5, x6, x7, x8)

        # Render into screen (same order as names)
        order = ["Гангренозний", "Флегмонозний", "Катаральний", "Інше"]

        for label in self.result_labels:
            label.setText(label.text().split(":")[0] + ":  ---")

        for r in results:
            for label in self.result_labels:
                if label.text().startswith(r["name"]):
                    label.setText(
                        f"{r['name']}:  Ранг={r['rank']}"
                    )


def main():
    app = QApplication(sys.argv)
    window = AppendicitisApp()
    window.show()

    try:
        app.exec()
    finally:
        app.quit()
        sys.exit(0)


if __name__ == "__main__":
    main()
```

Додаток Д

Результати викоонання програми для імітаційного моделювання

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 0 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 2, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 1 [{'ldfValue': 41.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 38.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 38.0, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 36.8, 'ldfName': 'ldf3', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 2 [{'ldfValue': 65.6, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 64.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 63.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 52.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 3, x4: 1, x6: 0, x7: 2, x8: 2
groupIndex: 1 rowIndex: 3 [{'ldfValue': 38.9, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 38.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 38.6, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 36.5, 'ldfName': 'ldf2', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 1 rowIndex: 4 [{'ldfValue': 48.9, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 45.6, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 45.3, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 42.9, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 0, x7: 2, x8: 2
groupIndex: 1 rowIndex: 5 [{'ldfValue': 38.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 37.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 36.4, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 2, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 6 [{'ldfValue': 41.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 38.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 38.0, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 36.8, 'ldfName': 'ldf3', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 7 [{'ldfValue': 69.5, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 69.2, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 67.7, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 2, x3: 2, x4: 3, x6: 2, x7: 2, x8: 2

groupIndex: 1 rowIndex: 8 [{'ldfValue': 60.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 57.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 54.4, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 48.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 9 [{'ldfValue': 68.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 66.5, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 65.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 3, x6: 2, x7: 2, x8: 0
groupIndex: 1 rowIndex: 10 [{'ldfValue': 37.0, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 35.8, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 35.4, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 33.6, 'ldfName': 'ldf2', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 11 [{'ldfValue': 68.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 66.5, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 65.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 12 [{'ldfValue': 42.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 41.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 39.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 38.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 0, x7: 2, x8: 2
groupIndex: 1 rowIndex: 13 [{'ldfValue': 40.9, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 39.6, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 39.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 38.9, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 14 [{'ldfValue': 71.8, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 71.4, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 70.2, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 57.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 15 [{'ldfValue': 62.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 59.7, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 58.3, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 3, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 16 [{'ldfValue': 52.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 49.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 47.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 45.8, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 1, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2

groupIndex: 1 rowIndex: 17 [{'ldfValue': 51.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 48.4, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 46.7, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 43.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 18 [{'ldfValue': 73.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 70.8, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 68.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 57.3, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 3, x6: 0, x7: 2, x8: 2
groupIndex: 1 rowIndex: 19 [{'ldfValue': 31.4, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 29.8, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 29.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 27.8, 'ldfName': 'ldf2', 'rang': 1}]

x1: 2, x2: 3, x3: 3, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 20 [{'ldfValue': 76.0, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 72.7, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 71.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 59.8, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 21 [{'ldfValue': 73.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 70.8, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 68.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 57.3, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 2, x3: 1, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 1 rowIndex: 22 [{'ldfValue': 33.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 32.7, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 32.0, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 31.3, 'ldfName': 'ldf3', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 23 [{'ldfValue': 68.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 66.5, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 65.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 24 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 1 rowIndex: 25 [{'ldfValue': 73.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 70.8, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 68.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 57.3, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2

groupIndex: 1 rowIndex: 26 [{'ldfValue': 65.6, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 64.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 63.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 52.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 1 rowIndex: 27 [{'ldfValue': 43.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 39.1, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.8, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 37.6, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 0 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 2, x4: 1, x6: 2, x7: 0, x8: 2
groupIndex: 2 rowIndex: 1 [{'ldfValue': 35.4, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 33.9, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 33.8, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 33.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 0, x7: 2, x8: 2
groupIndex: 2 rowIndex: 2 [{'ldfValue': 38.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 37.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 36.4, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 3 [{'ldfValue': 63.1, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 62.0, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 60.8, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 50.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 2 rowIndex: 4 [{'ldfValue': 48.1, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 44.0, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 41.4, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 40.1, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 2 rowIndex: 5 [{'ldfValue': 48.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 48.0, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 46.2, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 43.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 6 [{'ldfValue': 47.1, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 47.0, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 45.9, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 40.4, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 3, x6: 0, x7: 2, x8: 2

groupIndex: 2 rowIndex: 7 [{'ldfValue': 48.7, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 48.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 48.4, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 44.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 1, x6: 2, x7: 0, x8: 0
groupIndex: 2 rowIndex: 8 [{'ldfValue': 14.5, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 7.0, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 3.1, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -0.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 9 [{'ldfValue': 65.2, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 64.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 64.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 51.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 10 [{'ldfValue': 69.5, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 69.2, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 67.7, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 2 rowIndex: 11 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -19.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -24.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -27.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 12 [{'ldfValue': 65.6, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 64.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 63.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 52.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 1, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 13 [{'ldfValue': 52.6, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 52.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 49.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 42.6, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 0, x8: 0
groupIndex: 2 rowIndex: 14 [{'ldfValue': 26.1, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 24.1, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 20.6, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 17.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 0, x7: 2, x8: 2
groupIndex: 2 rowIndex: 15 [{'ldfValue': 35.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 33.6, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 33.3, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 31.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 4, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2

groupIndex: 2 rowIndex: 16 [{'ldfValue': 63.1, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 62.0, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 60.8, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 50.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 3, x6: 2, x7: 2, x8: 0
groupIndex: 2 rowIndex: 17 [{'ldfValue': 51.1, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 48.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 46.6, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 42.9, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 2 rowIndex: 18 [{'ldfValue': 36.9, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 36.8, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 36.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 35.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 2, x4: 1, x6: 0, x7: 2, x8: 2
groupIndex: 2 rowIndex: 19 [{'ldfValue': 33.2, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 32.3, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 31.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 28.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 20 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 21 [{'ldfValue': 65.2, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 64.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 64.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 51.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 22 [{'ldfValue': 71.8, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 71.4, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 70.2, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 57.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 3, x6: 0, x7: 2, x8: 2
groupIndex: 2 rowIndex: 23 [{'ldfValue': 48.7, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 48.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 48.4, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 44.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 2 rowIndex: 24 [{'ldfValue': 58.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 58.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 56.1, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 48.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 2, x6: 0, x7: 2, x8: 2

groupIndex: 3 rowIndex: 0 [{'ldfValue': 27.3, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 26.0, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 25.0, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 22.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 1, x6: 2, x7: 0, x8: 0
groupIndex: 3 rowIndex: 1 [{'ldfValue': 23.3, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 21.1, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 16.3, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 12.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 0, x7: 2, x8: 2
groupIndex: 3 rowIndex: 2 [{'ldfValue': 35.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 33.6, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 33.3, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 31.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 2, x6: 2, x7: 0, x8: 0
groupIndex: 3 rowIndex: 3 [{'ldfValue': 31.4, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 29.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 26.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 25.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 2, x8: 2
groupIndex: 3 rowIndex: 4 [{'ldfValue': 22.0, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 18.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 15.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 13.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 5 [{'ldfValue': 46.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 43.4, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 43.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 40.4, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 3 rowIndex: 6 [{'ldfValue': 65.2, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 64.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 64.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 51.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 1, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 7 [{'ldfValue': 31.0, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 29.8, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 25.4, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 25.0, 'ldfName': 'ldf2', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 2, x6: 0, x7: 0, x8: 2
groupIndex: 3 rowIndex: 8 [{'ldfValue': 15.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 6.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 6.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 2.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 1, x3: 2, x4: 2, x6: 2, x7: 2, x8: 0

groupIndex: 3 rowIndex: 9 [{'ldfValue': 39.5, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 38.4, 'ldfName': 'ldf1', 'rang': 3}, {'ldfValue': 37.9, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 35.5, 'ldfName': 'ldf2', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 1, x6: 2, x7: 0, x8: 2
groupIndex: 3 rowIndex: 10 [{'ldfValue': 33.0, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 32.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 32.4, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 31.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 3 rowIndex: 11 [{'ldfValue': 10.5, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -0.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -6.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -10.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 1, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 12 [{'ldfValue': 31.0, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 28.5, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 26.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 22.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 0, x7: 2, x8: 2
groupIndex: 3 rowIndex: 13 [{'ldfValue': 40.1, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 38.2, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 36.1, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 35.4, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 2, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 3 rowIndex: 14 [{'ldfValue': 68.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 66.5, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 65.5, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 54.5, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 1, x6: 0, x7: 2, x8: 0
groupIndex: 3 rowIndex: 15 [{'ldfValue': 15.4, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 11.1, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 4.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -2.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 16 [{'ldfValue': 43.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 39.1, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.8, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 37.6, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 2, x4: 2, x6: 2, x7: 0, x8: 2
groupIndex: 3 rowIndex: 17 [{'ldfValue': 48.2, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 48.0, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 46.2, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 43.0, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 3, x6: 0, x7: 2, x8: 2

groupIndex: 3 rowIndex: 18 [{'ldfValue': 38.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 37.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 36.4, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 3 rowIndex: 19 [{'ldfValue': 10.5, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -0.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -6.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -10.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 1, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 20 [{'ldfValue': 31.0, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 28.5, 'ldfName': 'ldf4', 'rang': 3}, {'ldfValue': 26.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 22.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 3 rowIndex: 21 [{'ldfValue': 60.4, 'ldfName': 'ldf1', 'rang': 4}, {'ldfValue': 60.3, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 60.0, 'ldfName': 'ldf3', 'rang': 2}, {'ldfValue': 49.2, 'ldfName': 'ldf4', 'rang': 1}]

x1: 2, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 2
groupIndex: 3 rowIndex: 22 [{'ldfValue': 65.2, 'ldfName': 'ldf2', 'rang': 4}, {'ldfValue': 64.7, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 64.0, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 51.7, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 2, x4: 1, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 23 [{'ldfValue': 41.2, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 37.6, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 36.3, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 34.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 3, x3: 1, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 24 [{'ldfValue': 43.4, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 39.1, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 37.8, 'ldfName': 'ldf1', 'rang': 2}, {'ldfValue': 37.6, 'ldfName': 'ldf4', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 2, x6: 2, x7: 2, x8: 0
groupIndex: 3 rowIndex: 25 [{'ldfValue': 38.7, 'ldfName': 'ldf3', 'rang': 4}, {'ldfValue': 35.7, 'ldfName': 'ldf2', 'rang': 3}, {'ldfValue': 33.8, 'ldfName': 'ldf4', 'rang': 2}, {'ldfValue': 31.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 0 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 2, x4: 1, x6: 0, x7: 0, x8: 0

groupIndex: 4 rowIndex: 1 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -21.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -27.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -28.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 2 [{'ldfValue': -1.1, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -17.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -23.9, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -29.4, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 3 [{'ldfValue': 3.0, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -14.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -21.1, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -21.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 4 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 5 [{'ldfValue': -6.1, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -27.0, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -33.7, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -36.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 6 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 2, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 7 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -21.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -27.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -28.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 8 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -19.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -24.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -27.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 9 [{'ldfValue': 0.2, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -17.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -25.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -26.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 2, x6: 2, x7: 0, x8: 0

groupIndex: 4 rowIndex: 10 [{'ldfValue': 14.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 5.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': 2.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': 0.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 11 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -19.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -24.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -27.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 1, x4: 2, x6: 0, x7: 0, x8: 2
groupIndex: 4 rowIndex: 12 [{'ldfValue': 8.3, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -7.4, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -8.2, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -8.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 2, x4: 1, x6: 0, x7: 2, x8: 0
groupIndex: 4 rowIndex: 13 [{'ldfValue': 13.2, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': 2.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -4.1, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -5.2, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 14 [{'ldfValue': 1.4, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -12.9, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -19.0, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -25.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 3, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 15 [{'ldfValue': -1.1, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -17.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -23.9, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -29.4, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 16 [{'ldfValue': 3.0, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -14.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -21.1, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -21.6, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 4, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 17 [{'ldfValue': 1.4, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -12.9, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -19.0, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -25.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 18 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 1, x4: 2, x6: 0, x7: 0, x8: 0

groupIndex: 4 rowIndex: 19 [{'ldfValue': -3.3, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -24.0, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -29.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -31.4, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 20 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 1, x3: 2, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 21 [{'ldfValue': -0.8, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -21.5, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -27.5, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -28.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 2, x2: 1, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 22 [{'ldfValue': 0.2, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -17.6, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -25.4, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -26.8, 'ldfName': 'ldf1', 'rang': 1}]

x1: 1, x2: 2, x3: 1, x4: 1, x6: 0, x7: 0, x8: 0
groupIndex: 4 rowIndex: 23 [{'ldfValue': -3.6, 'ldfName': 'ldf4', 'rang': 4}, {'ldfValue': -22.3, 'ldfName': 'ldf3', 'rang': 3}, {'ldfValue': -28.8, 'ldfName': 'ldf2', 'rang': 2}, {'ldfValue': -33.0, 'ldfName': 'ldf1', 'rang': 1}]

GROUP_1 71.43%
GROUP_2 44.00%
GROUP_3 50.00%
GROUP_4 100.00%

Додаток Ж

Список опублікованих праць за темою дисертації

Статті у журналах, що включенні до переліку наукових фахових видань України

12 Т.Б. Мартинюк, Д.О. Каташинський, М.В. Микитюк, та М.О. Зайцев, “Особливості обчислювальних процесів на базі SM-перетворення”, *Оптико-електронні інформаційно-енергетичні технології*, №2(44), с.32-37. 2022. https://doi.org/10.31649/1681-7893-2022-44-2-32-37

13 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, “Структурні особливості нейроподібного класифікатора об’єктів”, *Наукові праці ВНТУ*, №4, с.1-7. 2023. https://doi.org/10.31649/2307-5376-2023-4-1-7

14 Т.Б. Мартинюк, А.В. Кожем’яко, І.В. Булига, та Д.О. Каташинський, “Графічна модель паралельної асоціативної обробки числових даних”, *Наукові праці ВНТУ*, №4, с.58-62. 2024. https://doi.org/10.31649/2307-5376-2024-4-58-62

15 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, “Особливості процесу класифікації у контексті медичного моделювання”, *Наукові праці ВНТУ*, №1, с.80-85. 2025. https://doi.org/10.31649/2307-5376-2025-1-80-85

16 Т.Б. Мартинюк, Д.О. Каташинський, “Особливості асоціативного оброблення даних в інтелектуальних системах”, *Оптико-електронні інформаційно-енергетичні технології*, №1 , с.44-52. 2025. https://doi.org/10.31649/1681-7893-2025-49-1-44-52

17 Т.Б. Мартинюк, А.В. Кожем’яко, Д.О. Каташинський, та І.В. Булига, «Особливості обчислювального процесу у матричному класифікаторі об’єктів», *Наукові праці ВНТУ*, №3, 2025. https://praci.vntu.edu.ua/index.php/praci/article/view/850

Публікації в матеріалах конференцій, тезах доповідей

18 Д. О. Каташинський, «Комп’ютерна система для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу попередніх продаж», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»,* Вінниця: ВНТУ, 2021.
https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/view/13180

19 О. Каташинський, та Т.Б. Мартинюк, “Функціональні можливості оброблення даних на база SM-перетворення”, *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2023. https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/17047

20 Д.О. Каташинський, Т.Б. Мартинюк, та І.В. Булига, «Модель двовимірного нейроподібного класифікатора», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2024. https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21168

21 Т.Б. Мартинюк, М. А. Очкуров, Д.О. Каташинський, та І.В. Булига, «Структурні та функціональні особливості класифікаторів об’єктів», на *Міжнародній науково-практичній конференції з оптико-електронних технологій* “ФОТОНІКА − ODS 2025”. Вінниця, 2025. https://conferences.vntu.edu.ua/index.php/ods/ods_2025/paper/view/24668

22 І. В. Булига, Т.Б. Мартинюк, та Д.О. Каташинський, «Особливості асоціативної обробки числових даних за різницевими зрізами», *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця: ВНТУ, 2025. https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/48386/24839.pdf