МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Кваліфікаційна наукова
праця на правах рукопису

ТУШИЧ АЛІНА МИКОЛАЇВНА

УДК 004.032.26

ДИСЕРТАЦІЯ

МЕТОДИКА ПОБУДОВИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ АНАЛІЗУ ДАНИХ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

Спеціальність 123 «Комп'ютерна інженерія»
Галузь знань 12 «Інформаційні технології»
Подається на здобуття наукового ступеня
доктора філософії

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

Науковий керівник: СТОРЧАК Каміла Павлівна, доктор технічних наук, професор

Київ – 2021

АНОТАЦІЯ

Тушич А.М. Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – комп'ютерна інженерія (галузь знань 12 – інформаційні технології). – Державний університет телекомунікацій Міністерства освіти і науки України, Київ, 2021.

Дисертаційна робота присвячена актуальній науковій задачі розробки методики побудови інтелектуальної системи аналізу даних на основі нейронних мереж. Тематика дисертаційного дослідження відповідає тимчасовому стандарту та фаховим компетентностям освітньо-наукової програми підготовки докторів філософії з комп'ютерної інженерії Державного університету телекомунікацій Міністерства освіти і науки України, а саме: фундаментальним науковим дослідженням теоретико-методологічних, науково-методичних та прикладних засад підвищення ефективності інноваційної та виробничої діяльності підприємства, а також вдосконаленню процесу забезпечення впровадження новітніх інформаційних технологій на об'єктах інформаційної діяльності.

Для забезпечення заданих високих вимог до кількісних і якісних показників обробки інформації, що є процесом інформаційної взаємодії технологічних процесів сучасних підприємств і організацій, необхідна реалізація інтелектуальної системи аналізу даних.

Для досягнення цього завдання необхідно спроектувати таку систему, яка буде стійкою до відмов, мати можливість до автоматичного відновлення, бути байдужою до потрапляння зашумлених даних та показувати результати у прийнятному для користувача вигляді.

У роботі проведено порівняльний аналіз технологій, які вирішують задачу аналізу накопичених даних, що отримують підприємства у процесі своєї діяльності. Однак розробці методів і програм для аналізу даних, які можуть ідентифікувати потенційно корисну, але неявну інформацію, приділяється не багато уваги. Отримання цієї інформації може дати життєво важливий імпульс науковим дослідженням в інших областях. Цей нетривіальний витяг неявної, раніше невідомої і потенційно корисної інформації з великих баз даних відомий як інтелектуальний аналіз даних або виявлення знань.

Одна з вимог до інтелектуальних систем виявлення знань – ефективність і масштабованість. Робота з дуже великими базами даних вимагає ефективних алгоритмів, а неточність і часто неповнота даних створює додаткові проблеми при отриманні прихованих шаблонів. Навчені нейронні мережі здатні генерувати приховані знання з даних: створюється можливість прогнозування, класифікації та розпізнавання образів, але їх логічна структура з традиційним підходом залишається прихованою від користувача.

У дисертаційній роботі сформульовано вимоги до сучасних систем аналізу даних для обґрунтування вибору математичного апарату ядра аналітичної системи, розроблено новий метод нелінійної нормалізації даних, який полягає у поступовій реалізації перетворень над даними із змінним типом нелінійності, досліджено, які нейронні мережі мають перевагу над традиційними методами математичної статистики та технічного аналізу для визначеної задачі, обрано оптимальну топологію і параметри обраної технології, розроблено новий підхід до явного отримання правил від навченої нейронної мережі, який полягає у групуванні вхідних параметрів та активності нейронів, розроблено інформаційну технологію визначення пошуку правил функціонування досліджуваного об’єкта в явній формі на основі обробки ряду емпіричних даних за допомогою апарату нейронної мережі.

Метою даної дисертації є оптимізація процесу виявлення прихованих закономірностей, що містяться в базах даних за допомогою вдосконаленої методики обробки даних.

Для досягнення цієї мети необхідно вирішити такі *задачі*:

1. Визначити вимоги до сучасних аналітичних систем, виходячи з представлених вимог, для обґрунтування вибору математичного апарату ядра аналітичної системи.
2. Розробити метод попередньої обробки даних, що аналізуються.
3. Провести дослідження на основі побудови алгоритму аналізу даних, використовуючи обрану нейромережеву технологію.
4. Обрати оптимальну топологію і параметри обраної топології.
5. Розробити алгоритм виявлення знань в аналізованих даних.
6. Розробити програмну реалізацію інформаційної технології для імітаційного моделювання процесів системи.

Методи дослідження. Результати проведених і представлених в дисертації досліджень отримані з використанням методів системного аналізу, теорії інформації, теорії ймовірностей, методів статистичного аналізу, комбінаторики, нейромережевого моделювання, кластерного аналізу, оптимізації, формалізації та методів динамічного програмування.

Наукова новизна дисертації:

1. Вперше розроблено метод нелінійної нормалізації даних, який ґрунтується на послідовному виконанні перетворень змінного типу нелінійності.
2. Удосконалено алгоритм навчання нейронної мережі, який відрізняється від існуючих застосуванням методу задання адаптивних параметрів.

3. Набув подальшого розвитку алгоритм виявлення принципів роботи тестової системи на навченій нейронній мережі, який відрізняється від існуючих групуванням вхідних параметрів і активності нейронів мережі.

4. Розроблено інформаційну технологію визначення закономірностей в накопичених даних на основі методу нелінійної нормалізації, методик навчання та виявлення принципів роботи системи.

Практична цінність роботи полягає в розробці формальної методології, яка дозволяє використовувати її в багатьох організаціях. Дисертація містить конкретні практичні рекомендації з використання запропонованої в ній методології і прийомів в різних прикладних областях. Впровадження програмного забезпечення аналітичної системи, створеного в рамках дисертації, може бути використано для автоматизації аналітичної роботи.

Застосування результатів роботи. Результати наукових досліджень були використані на кафедрі інформаційних систем та технологій Навчально-наукового інституту інформаційних технологій Державного університету телекомунікацій під час виконання науково-дослідної роботи на тему «Розробка системи активного управління чергою пакетів в мережах TCP/IP з використанням REM-регуляторів» (№ 0119U101284, ДУТ, м. Київ), госпдоговірних робіт «Аналіз ринку та дослідження перспектив розвитку комп'ютерно-інформаційного обладнання» (ДУТ, м. Київ) та «Дослідження обробки траєкторної інформації в вимірювально-обчислювальних системах» (ДУТ, м. Київ), впроваджені у виробничий процес на підприємствах ТОВ «ХУАВЕЙ Україна», ТОВ «ПІПЛ УА», ТОВ «Ай Ті Джи» та в навчальному процесі Державного університету телекомунікацій (Київ).

Основні положення для захисту:

1. Вперше розроблено метод нелінійної нормалізації даних, який ґрунтується на послідовному виконанні перетворень змінного типу нелінійності.

2. Удосконалено алгоритм навчання нейронної мережі, який відрізняється від існуючих застосуванням методу задання адаптивних параметрів.

3. Набув подальшого розвитку алгоритм виявлення принципів роботи тестової системи на навченій нейронній мережі, який відрізняється від існуючих групуванням вхідних параметрів і активності нейронів мережі.

4. Розроблено інформаційну технологію визначення закономірностей в накопичених даних на основі методу нелінійної нормалізації, методик навчання та виявлення принципів роботи системи.

Тематика дисертаційної роботи і отримані результати безпосередньо відповідають пріоритетності розвитку інформаційних та комунікаційних технологій в Україні до 2021 р. згідно із Законом України «Про пріоритетні напрями розвитку науки і техніки», від 11.07.2001 № 2623-III, зі змінами внесеними згідно із Законом України «Про наукову та науково-технічну діяльність» від 26.11.2015 № 848-VIII. Дисертаційна робота виконана відповідно до планів наукової і науково-технічної діяльності Державного університету телекомунікацій і є частиною досліджень в рамках науково-дослідної роботи «Розробка системи активного управління чергою пакетів в мережах TCP/IP з використанням REM-регуляторів» (№ 0119U101284, ДУТ, м. Київ), а також госпдоговірних робіт «Аналіз ринку та дослідження перспектив розвитку комп’ютерно-інформаційного обладнання» (ДУТ, м. Київ) та «Дослідження обробки траєкторної інформації в вимірювально-обчислювальних системах» (ДУТ, м. Київ).

Ключові слова: інформаційні технології, штучний інтелект, нейронні мережі, кластеризація, інтелектуальний аналіз даних, виявлення знань, великі дані.

ANNOTATION

Tushych A.M. Methods for building an intelligent data analysis system based on neural networks. – Qualifying scientific work as a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 – computer engineering (area of knowledge 12 – information technology). – State University of Telecommunications of the Ministry of Education and Science of Ukraine, Kyiv, 2021.

The dissertation work is devoted to the urgent scientific task of developing a methodology for constructing an intelligent data analysis system based on neural networks. The topic of the dissertation research corresponds to the temporary standard and professional competence of the educational and scientific program for the training of doctors of philosophy in computer engineering of the State University of Telecommunications of the Ministry of Education and Science of Ukraine, namely: fundamental scientific research of theoretical and methodological, scientific and methodological and applied foundations of increasing the efficiency of innovative and production activities enterprises, as well as improving the process of ensuring the introduction of the latest information technologies at the objects of information activities.

To ensure the specified high requirements for quantitative and qualitative indicators of information processing, is the process of information interaction of technological processes of modern enterprises and organizations, it is necessary to implement an intelligent data analysis system.

To achieve this goal, it is necessary to design a system that will be resilient to failures, be able to automatically recover, be indifferent to noisy data and show the results in a form acceptable to the user.

The paper provides a comparative analysis of technologies that solve the problem of analyzing the accumulated data that enterprises receive in the course of their activities. However, little attention has been paid to the development of methods and

programs for data analysis that can identify potentially useful but implicit information. Obtaining this information can provide a vital impetus for research in other areas. This nontrivial extraction of implicit, previously unknown and potentially useful information from large databases is known as data mining or knowledge discovery.

One of the requirements for intelligent knowledge discovery systems is efficiency and scalability. Working with very large databases requires efficient algorithms, and inaccuracies and often incomplete data creates additional problems when retrieving hidden patterns. Taught neural networks are able to generate hidden knowledge from data: the ability to predict, classify and recognize patterns is created, but their logical structure with the traditional approach remains hidden from the user.

The dissertation work formulates the requirements for modern data analysis systems to justify the choice of the mathematical apparatus of the core of the analytical system, a new method of nonlinear data normalization is developed, which consists in the gradual implementation of transformations on data with a variable type of nonlinearity, it is investigated which neural networks have an advantage over traditional methods of mathematical statistics and technical analysis for a specific task, the optimal topology and parameters of the selected technology have been chosen, a new approach has been developed to explicitly obtaining rules from a trained neural network, consists in grouping input parameters and neuron activity, an information technology has been developed for determining the search for the rules of functioning of the object under study in an explicit form on based on processing a number of empirical data using a neural network apparatus.

The *purpose* of this dissertation is to optimize the process of revealing hidden patterns contained in databases using improved data processing techniques.

To achieve this goal, it is necessary to solve the following *tasks*:

1. Determine the requirements for modern analytical systems, based on the presented requirements, to justify the choice of the mathematical apparatus of the core of the analytical system.

2. Develop a method for preprocessing the data being analyzed.

3. Conduct a study based on the construction of a data analysis algorithm using the selected neural network technology.

4. Select the optimal topology and parameters of the selected topology.

5. Develop an algorithm for identifying knowledge in the analyzed data.

6. Develop a software implementation of information technology for simulation of system processes.

Research methods. The results of the research carried out and presented in the dissertation were obtained using the methods of systems analysis, information theory, probability theory, methods of statistical analysis, combinatorics, neural network modeling, cluster analysis, optimization, formalization and dynamic programming methods.

Scientific novelty of the thesis:

1. For the first time, a method of nonlinear data normalization based on the sequential execution of transformations of a variable type of nonlinearity has been developed.

2. The algorithm for learning the neural network has been improved, it differs from the existing ones by the application of the method for setting adaptive parameters.

3. Entered the further development of the algorithm for identifying the principles of the test system on a trained neural network, which differs from the existing grouping of input parameters and the activity of neurons in the network.

4. An information technology has been developed for determining patterns in the accumulated data based on the nonlinear normalization method, teaching methods and identifying the principles of the system's operation.

The practical value of the work lies in the development of a formal methodology that allows it to be used in many organizations. The dissertation contains specific practical recommendations on the use of the proposed methodology and techniques in

various applied fields. The implementation of the analytical system software, created within the framework of the dissertation, can be used to automate analytical work.

Application of work results. The results of scientific research were used at the Department of Information Systems and Technologies of the Educational and Scientific Institute of Information Technologies of the State University of Telecommunications when performing research work on the topic "Development of a system for active management of packet queue in TCP / IP networks using REM regulators" (No. 0119U101284, FLS., Kyiv), contractual works "Analysis of the market and research of prospects for the development of computer information equipment" (FLS., Kyiv) and "Study of processing trajectory information in measuring and computing systems" (FLS., Kyiv) , introduced into the production process at the enterprises of Huawei Ukraine LLC, PIPL UA LLC, ITG LLC and in the educational process of the State University of Telecommunications (Kyiv).

Basic provisions for protection:

1. For the first time, a method of nonlinear data normalization based on the sequential execution of transformations of a variable type of nonlinearity has been developed.

2. The algorithm for learning the neural network has been improved, it differs from the existing ones by the application of the method for setting adaptive parameters.

3. Entered the further development of the algorithm for identifying the principles of the test system on a trained neural network, which differs from the existing grouping of input parameters and the activity of neurons in the network.

4. An information technology has been developed for determining patterns in the accumulated data based on the nonlinear normalization method, teaching methods and identifying the principles of the system's operation.

The topic of the dissertation work and the results obtained directly correspond to the priority of the development of information and communication technologies in Ukraine until 2021. According to the Law of Ukraine "On priority directions of

development of science and technology", dated July 11, 2001 No. 2623-III, as amended in accordance with the Law of Ukraine "On scientific and scientific and technical activities" dated November 26, 2015 No. 848-VIII. The dissertation was completed in accordance with the plans of scientific and scientific and technical activities of the State University of Telecommunications and is part of the research within the framework of the research work "Development of an active packet queue management system in TCP / IP networks using REM regulators" (No. 0119U101284, FLS., Kyiv), as well as contractual works "Analysis of the market and research of prospects for the development of computer-information equipment" (FLS., Kyiv) and "Study of processing trajectory information in measuring and computing systems" (FLS., Kyiv).

Keywords: information technology, artificial intelligence, neural networks, clustering, data mining, knowledge discovery, big data.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. V. Savchenko, V. Akhramovych, A. Tushych, I. Sribna, and I. Vlasov, “Analysis of Social Network Parameters and the Likelihood of its Construction”, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no 2, pp. 271-276, 2020.

2. K. Storchak, N. Yakovenko, A. Tushych, I. Sribnaya, and O. Polonevich, “Improving the material quality of network equipment due to a mechanism of surface hardening”, *Scientific discussion*, vol. 1, no. 49, pp. 34-39, 2020.

3. А.М. Тушич, К.П. Сторчак, А.П. Бондарчук, та А.О. Макаренко, “Вимоги до інтелектуальних систем аналізу даних та їх класифікацій”, *Науково-технічний журнал “Телекомунікаційні та інформаційні технології”,* №1, с. 31-36, 2019.

4. К.П. Сторчак, А.М. Тушич, та А.П. Бондарчук, “Кластерний аналіз даних з використанням штучних нейронних мереж”, *Науковий журнал “Зв’язок”,* №6, с. 36-38, 2018.

5. К.П. Сторчак, А.М. Тушич, К.С. Козелкова, та М.М. Степанов, “Інтелектуальний аналіз даних з використанням нейронних мереж”, *Науковий журнал “Зв’язок”,* №4, с. 17-19, 2018.

6. К.П. Сторчак, А.М. Тушич, О.М. Ткаленко, В.М. Чорна, та Т.М. Жила, “Аналіз вимог до проектування хмарної платформи для інтернету речей”, *Науковий журнал “Зв’язок”,* №6, с. 26-34, 2019.

7. О.А. Золотухіна, О.М. Ткаленко, А.М. Тушич, В.М. Чорна, та О.Р. Нікітенко, “Концепція розвитку підсистеми передавання мультимедійних повідомлень IMS”, *Науково-технічний журнал “Телекомунікаційні та інформаційні технології”*, №4, с. 81-89, 2019.

8. І.С. Сиротенко, І.С. Щербина, К.П. Сторчак, та А.М. Тушич, “Аналіз ефективності викристання нейронних мереж на прикладі багатошарового персептрону та мережі Кохонена”, *Науковий журнал “Зв’язок”,* №5, с. 17-19, 2020.

9. Б.В. Шефкін, І.В. Красюк, В.О. Хоменчук, К.П. Сторчак, та А.М. Тушич, “Дослідження та впровадження нейронної мережі на основі TENSORFLOW”, *Науковий журнал “Зв’язок”,* №6, с. 18-20, 2020.

10. К.П. Сторчак, Д.В. Кравець, А.М. Тушич, та Д.В. Сорокін, “Аналіз методів організації прав користувачів у GNU\Linux системах”, *Науковий журнал “Зв’язок”,* №4, с. 38-40, 2020.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

11. А.М. Тушич, “Використання штучних нейронних мереж для створення IoT рішення фільтрації VPN трафіку”, на *XI Міжнар. наук.-техн. конф.*

студентів та аспірантів "Перспективи розвитку інформаційно-телекомунікаційних технологій та систем", Київ, 2019, с. 361.

12. А.М. Тушич, "Машинне навчання з використанням TENSORFLOW", на *XII Міжнар. наук.-техн. конф. студентів та аспірантів "Перспективи розвитку інформаційно-телекомунікаційних технологій та систем",* Київ, 2020, с. 379.

13. М. Пелепей, та А. Тушич, "Штучний інтелект – друг, ворог чи помічник людини", на *IX Міжнар. наук.-техн. конф. студентства та молоді "Світ інформації та телекомунікацій"*, Київ, 2019, с. 303-304.

14. А.М. Тушич, "Аналіз доцільності використання автоматизованої системи інтелектуального аналізу даних на основі штучних нейронних мереж", на *VII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених з автоматичного управління присвячена Дню космонавтики*, Херсон, 2019, с. 76-77.

15. Д.Я. Алтинніков, та А.М. Тушич, "Як штучний інтелект та IOT доповнюють один одного", на *Всеукр. наук.-техн. конф. "Сучасний стан та перспективи розвитку IOT"*, Київ, 2020, с. 303-304.

16. А.М. Тушич, та П.В. Лебединець, "Дослідження системи моніторингу Zabbix для IT-інфраструктури підприємства", на *VII Наук.-техн. конф. "Сучасні інфокомунікаційні технології"*, Київ, 2018, с. 20-21.

17. А.М. Тушич, та О.М. Скрипаль. "Безпека функціонування телекомунікаційніх систем та мереж", на *XII Міжнар. наук.-техн. конф. "Проблеми інформатизації"*, Київ, 2018р. – С.75-76.

18. А.М. Тушич, та П.В. Лебединець, "Дослідження відкритого програмного забезпечення поштового сервера та клієнта", на *XII Міжнар. наук.-техн. конф "Проблеми інформатизації"*, Київ, 2018, с. 105.

19. Б. Свердлюк, Ю.Каграманова, та А. Тушич, “Інтернет речей”, на *IX Міжнар. наук.-техн. конф. студентства та молоді “Світ інформації та телекомунікацій*”, Київ, 2019, с. 107-108.

ЗМІСТ

ВСТУП

Актуальність теми.

В даний час накопичені надзвичайно об'ємні бази даних, в яких зберігається інформація з різних областей. Для роботи з ними розроблені спеціальні технології, які вирішують задачі накопичення, зберігання, статистичної обробки та управління. Однак розробці методів і програм для аналізу даних, які можуть ідентифікувати потенційно корисну, але неявну інформацію, приділяється не багато уваги. Отримання цієї інформації може дати життєво важливий імпульс науковим дослідженням в інших областях. Цей нетривіальний витяг неявної, раніше невідомої і потенційно корисної інформації з великих баз даних відомий як інтелектуальний аналіз даних або виявлення знань. Під час виявлення знань використовуються концепції, розроблені в таких областях, як машинне навчання, технологія баз даних, статистика та інші.

Сьогодні для інтелектуального аналізу даних широко використовуються системи аналізу та пакети статистики.

1. Аналітичні системи повинні вирішувати конкретні задачі з чітко визначеною областю застосування. Ці системи сильно розрізняються за використовуваними методами, наприклад, системи аналізу фінансового ринку побудовані на основі методів технічного аналізу. Технічний аналіз - це поєднання декількох методів прогнозування динаміки цін і вибору оптимальної структури інвестиційного портфеля на основі різних емпіричних моделей поведінки ринку. Оскільки вся логіка зазвичай міститься в системі і не виводиться з історії ринку, вимоги до статистичної значущості отриманих моделей і можливості їх інтерпретації не мають сенсу [1, 2].

2. Статистичні пакети орієнтовані на класичні методи математичної статистики - кореляційний, регресійний, факторний аналіз та інші [3-5]. Основним недоліком цього класу систем є те, що їх неможливо ефективно

використовувати для аналізу даних без глибоких знань статистики. Крім того, системи, засновані на обробці статистичної інформації, вимагають від аналітиків робити апріорні припущення про моделі. Як правило, потрібна спеціальна обробка вихідних даних (наприклад, випадкові вибірки), деякий вибір моделей з набору затверджених моделей (для перевірки адекватності опису даних) і, нарешті, професійна інтерпретація результатів. Традиційні математичні статистичні методи, на яких засновані статистичні пакети, в основному корисні для перевірки заздалегідь сформульованих гіпотез і попереднього дослідного аналізу, що лежить в основі оперативної аналітичної обробки (OLAP) [3, 6-8].

Доповненням і в багатьох випадках альтернативою згаданим вище методам є стратегія пошуку знань в базах даних.

Одна з вимог до інтелектуальних систем вилучення знань - ефективність і масштабованість. Робота з дуже великими базами даних вимагає ефективних алгоритмів, а неточність і часто неповнота даних створює додаткові проблеми при отриманні прихованих шаблонів.

Тут нейронні мережі мають перевагу, оскільки вони є ефективним засобом обробки зашумлених даних. Однак основною претензією до нейронних мереж завжди була відсутність явного вираження виявлених правил [1, 2, 5, 9, 10].

Навчені нейронні мережі здатні генерувати приховані знання з даних: створюється можливість прогнозування, класифікації та розпізнавання образів, але їх логічна структура з традиційним підходом залишається прихованою від користувача.

В цьому відношенні нейронні мережі в даний час розглядаються тільки як інструмент прогнозування, а не як засіб розуміння. Класичний підхід для нейронних мереж - метод «чорного ящика» - передбачає створення імітаційної моделі без явної формулювання правил прийняття рішень для нейронних мереж. Ці правила містяться у вагах навченої нейронної мережі, але зрозуміти їх, перевівши на мову правил «якщо … то …», не представлялося можливим. У

зв'язку з цим розробка методик, що дозволяють створювати аналогічні правила, що пояснюють вирішення задач з нейронними мережами, є актуальним завданням. Таким чином, нейронні мережі можна використовувати не тільки для прогнозів, але і для отримання знань з баз даних.

Створення правил виводу і баз даних знань традиційно вважалося прерогативою експертних систем. Експертні системи спеціально орієнтовані на обробку даних з використанням деяких правил виведення, які повинні бути отримані від експертів в певній галузі знань. Експертні системи призначені для реалізації ланцюжків міркувань, що імітують аналіз ситуації експертом в певній області застосування.

Питання застосування інформаційних технологій в системах аналізу даних, в тому числі, в задачах інтелектуального аналізу, розглядають такі світові та українські вчені як Кохонен Т., Кохаві Р., Гроссберг С., Бертсекас Д., Шеннон К., Цицикліс Д., Екштейн Дж., Кригель Х., Джон Д., Ленглі П., Іба У., Уідроу Б., Беркман Л., Самойленко А., Стєклов В. та інші.

Зв'язок роботи з науковими програмами, планами, темами.

Тематика дисертаційної роботи і отримані результати безпосередньо відповідають пріоритетності розвитку інформаційних та комунікаційних технологій в Україні до 2020 р. згідно із Законом України «Про пріоритетні напрями розвитку науки і техніки», від 11.07.2001 № 2623-III, зі змінами внесеними згідно із Законом України «Про наукову та науково-технічну діяльність» від 26.11.2015 № 848-VIII. Дисертаційна робота виконана відповідно до планів наукової і науково-технічної діяльності Державного університету телекомунікацій і є частиною досліджень в рамках науково-дослідної роботи «Розробка системи активного управління чергою пакетів в мережах TCP/IP з використанням REM-регуляторів» (№ 0119U101284, ДУТ, м. Київ), а також госпдоговірних робіт «Аналіз ринку та дослідження перспектив розвитку комп'ютерно-інформаційного обладнання» (ДУТ, м. Київ) та «Дослідження

обробки траєкторної інформації в вимірювально-обчислювальних системах» (ДУТ, м. Київ).

Мета і завдання дослідження.

Метою даної дисертації є оптимізація процесу виявлення прихованих закономірностей, що містяться в базах даних за допомогою вдосконаленої методики обробки даних.

Для досягнення цієї мети необхідно вирішити такі *задачі*:

1. Визначити вимоги до сучасних аналітичних систем, виходячи з представлених вимог, для обґрунтування вибору математичного апарату ядра аналітичної системи.
2. Розробити метод попередньої обробки даних, що аналізуються.
3. Провести дослідження на основі побудови алгоритму аналізу даних, використовуючи обрану нейромережеву технологію.
4. Обрати оптимальну топологію і параметри обраної топології.
5. Розробити алгоритм виявлення знань в аналізованих даних.
6. Розробити програмну реалізацію інформаційної технології для імітаційного моделювання процесів системи.

Об'єктом дослідження є процес функціонування системи аналізу даних.

Предметом дослідження є методики та інформаційна технологія аналізу даних.

Методи дослідження. Результати проведених і представлених в дисертації досліджень отримані з використанням методів системного аналізу, теорії інформації, теорії ймовірностей, методів статистичного аналізу, комбінаторики, нейромережевого моделювання, кластерного аналізу, оптимізації, формалізації та методів динамічного програмування.

Наукова новизна одержаних результатів полягає у наступному:

1. Вперше розроблено метод нелінійної нормалізації даних, який ґрунтується на послідовному виконанні перетворень змінного типу нелінійності

2. Удосконалено алгоритм навчання нейронної мережі, який відрізняється від існуючих застосуванням методу задання адаптивних параметрів

3. Набув подальшого розвитку алгоритм виявлення принципів роботи тестової системи на навченій нейронній мережі, який відрізняється від існуючих групуванням вхідних параметрів і активності нейронів мережі

4. Розроблено інформаційну технологію визначення закономірностей в накопичених даних на основі методу нелінійної нормалізації, методик навчання та виявлення принципів роботи системи

Практичне значення отриманих результатів.

Практична цінність роботи полягає в розробці формальної методології, яка дозволяє використовувати її в багатьох організаціях. Дисертація містить конкретні практичні рекомендації по використанню запропонованої в ній методології і прийомів в різних прикладних областях. Впровадження програмного забезпечення аналітичної системи, створеного в рамках дисертації, може бути використано для автоматизації аналітичної роботи.

Особистий внесок здобувача. Всі положення, які виносяться на захист, належать особисто автору. В роботах, які опубліковано в співавторстві, особисто здобувачу належать: проаналізовано параметри мереж [11], досліджено процес підвищення якості матеріалів мережевого обладнання із застосуванням нейромережевих технологій [12], систематизовано вимоги до інтелектуалних систем аналізу даних відповідно до їх класифікації [13], досліджено процес кластерного аналізу даних з використанням нейронних мереж та на його основі розроблено алгорим виявлення закономірностей в даних [14], побудовано алгоритм системи аналізу даних з використанням нейронних мереж [15], проаналізовані вимоги до проектування хмарної платформи на основі нейронних мереж [16], досліджено концепцію розвитку підсистеми передавання мультимедійних повідомлень з використанням нейронних мереж [17], досліджено ефективність застосування багатошарового персептрону та мережі

Коханена [18], здійснено моделювання структури нейронної мережі [19], досліджено методи організації прав користувачів у системах [20].

Апробація результатів дисертації. Основні результати дисертаційних досліджень доповідалися й обговорювалися на таких конференціях і семінарах [21-29]:

- XI Міжнародна науково-технічна конференція студентів та аспірантів «Перспективи розвитку інформаційно-телекомунікаційних технологій та систем» (15-19 квітня 2019р., м. Київ);
- XII Міжнародна науково-технічна конференція студентів та аспірантів «Перспективи розвитку інформаційно-телекомунікаційних технологій та систем» (13-17 квітня 2020р., м. Київ);
- IX Міжнародна науково-технічна конференція студентства та молоді «Світ інформації та телекомунікацій» (10 жовтня 2019р., м. Київ);
- VII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених з автоматичного управління присвячена Дню космонавтики (10-12 квітня 2019р., м. Київ);
- Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT» (3 квітня 2020р., м. Київ);
- VII Наук.-техн. конф. “Сучасні інфокомунікаційні технології” (5 грудня 2018 р., м. Київ);
- XII Міжнар. наук.-техн. конф. “Проблеми інформатизації” (12-13 грудня 2018 р., м. Київ).

Структура та обсяг дисертаційної роботи. Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 133 найменувань на 14 сторінках. Загальний обсяг роботи становить 160 сторінок серед яких 117 сторінок основного тексту, 23 рисунки, 22 таблиці.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ СИСТЕМ АНАЛІЗУ ВЕЛИКИХ ДАНИХ

1.1. Вимоги до інтелектуальних систем аналізу і класифікації систем аналізу великих даних

Як згадувалося у вступі, багато бізнес-процесів пов'язані з накопиченням інформації, в результаті чого накопичилися бази даних значних обсягів, через що виникає складність їх прямого емпіричного аналізу. Для управління великими об’ємами інформації необхідні інструменти обробки даних, часто пов'язані з областю штучного інтелекту, розроблені на основі популярного сьогодні підходу до вилучення знань. Такі програмні продукти дозволяють розуміти і оцінювати дані як в кількісному, так і в якісному відношенні. Крім того, в них підкреслюється все нове, цінне, потенційно корисне.

У зв'язку з удосконаленням технологій запису і зберігання даних стало можливим накопичити значний обсяг інформації в різних областях людської діяльності. Діяльність будь-якого підприємства (торгового, виробничого, медичного, наукового і т. д.) тепер супроводжується реєстрацією і фіксацією всіх деталей його діяльності. Очевидно, що без використання продуктивної обробки, дані є сховищем непотрібної інформації [13, 15-17, 30, 31].

Визначимо сучасні вимоги до інтелектуальних систем аналізу даних. Ці системи повинні виконувати ряд функцій, а саме:

- підтримка роботи з великими об’ємами даних,
- супровід робіт даними різного якісного складу,
- виявлення та робота з зашумленими даними,
- використання єдиного математичного апарату для вирішення завдань в різних областях застосування,

- звільнення від особливих вимог до дослідника з точки зору знання математичного апарату, що лежить в основі аналізу.

Крім того, результати аналізу повинні бути легко зрозумілі фахівцям в даній області.

Аналітична обробка даних - це мультидисциплінарна область, що створена і розвивається на основі прикладних статистичних методів, методів розпізнавання образів, методів штучного інтелекту, теорії баз даних і т. д (рис. 1.1). Тому значна кількість методів і алгоритмів реалізована в різних існуючих системах видобування даних. Багато з цих систем об'єднують кілька підходів одночасно. Однак у кожної системи зазвичай є основою алгоритм, від якого очікується кращий результат. Нижче наведено класифікацію інтелектуальних систем за використовуваними в них ключовими компонентами на основі робіт [13, 15, 31- 33]. Представлено короткий опис обраних класів.

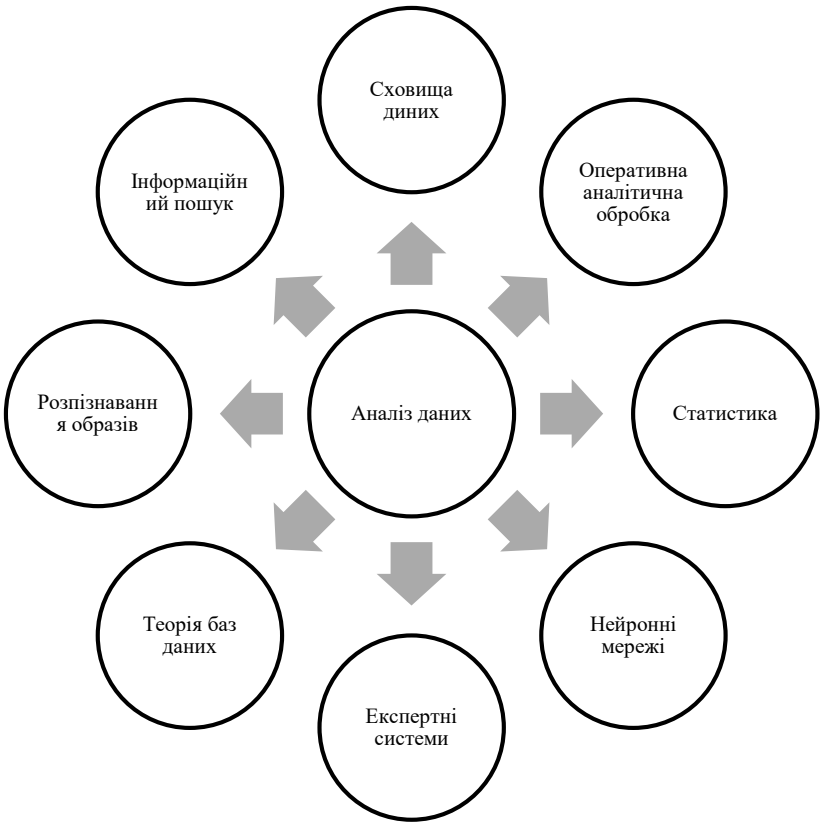


Рис. 1.1. Методи видобування даних

Аналітичні предметно-орієнтовані системи

Як було відзначено у вступі, такі системи дуже різноманітні, вибір використовуваних в цих системах методів повинен здійснюватися в залежності від класів розв'язуваних завдань. Наприклад, методи аналізу фінансових ринків засновані на технічному аналізі. Зверніть увагу, що безліч систем більш орієнтовані на західний ринок, часто не враховують особливості використання на українському ринку. Вони більшою мірою задовольняють потребу в більш простому розумінні, ніж інші обговорювані системні класи, що є зрозумілими фахівцям в цих областях, зазвичай мають певні інтерфейси для завантаження даних і пропонують інші переваги спеціалізованих систем. Є багато програм цього класу. Серед тих, що можна купити в Україні, - MetaStock, Candlestick Forecaster і Wall Street Raider. Однак як системи інтелектуального аналізу даних, вони не задовольняють одну з вимог сучасних аналітичних систем - можливості використовувати систему для обробки даних з широкого діапазону областей.

Статистичні пакети

Хоча останні версії майже всіх відомих статистичних пакетів містять елементи розпізнавання знань поряд з традиційними статистичними методами, вони як і раніше орієнтовані на класичні методи - кореляцію, регресію, факторний аналіз та інші. Головний недолік систем цього класу полягає в тому, що їх не можна ефективно використовувати для аналізу даних без глибоких знань в області статистики. Зазвичай при інтелектуальному аналізі даних з використанням пакетів статистичних програм необхідно багаторазово застосовувати серію одних і тих же елементарних операцій. Однак в цих системах інструменти для автоматизації дослідного процесу або відсутні, або мають бути запрограмовані в деяких внутрішніх системах на мові, яка рідко відома для користувача, якщо він не є ні статистом, ні програмістом. Всі ці фактори роблять потужні сучасні статистичні пакети занадто громіздкими для масового

використання. Крім того, ці системи часто досить дорогі. Приклади, доступні в Україні: SAS, SPSS (PASW Statistics) та Statgraphics (Statgraphics Centurion).

Нейронні мережі

Це великий клас різних систем, архітектура яких в деякій мірі імітує побудову нервової тканини з нейронів. В одній з найбільш поширених архітектур - багатошаровому персептроні зі зворотним поширенням помилки - функція нейронів емулюється як частина ієрархічної мережі, при цьому кожен нейрон нижнього рівня пов'язаний своїми входами з виходами нейронів попереднього рівня. На нейрони вхідного шару подаються значення вхідних параметрів, на основі яких повинна формуватися здатність прогнозувати, класифікувати і т. д. Ці параметри є сигналами, які послідовно поширюються від вхідного рівня через прихований до вихідного шару і послаблюються або посилюються відповідно до адаптивних ваг. В результаті на виході нейронів вихідного шару генерується значення, що представляє відповідь. Перш ніж нейронну мережу можна буде використовувати, її необхідно спочатку навчити на даних, для яких відомі значення вхідних і вихідних параметрів. Це навчання складається з вибору значень ваг міжнейронних зв'язків, які забезпечують найбільшу близькість відповідей мережі до відомих надійних результатів. Недоліком парадигми нейронної мережі є необхідність навчальної вибірки, що адекватно характеризує досліджувану систему. Інший недолік, виявлений різними авторами [12, 33-36], полягає в тому, що навіть навчена нейронна мережа являє собою «чорний ящик». Знання, зафіксовані як вага міжнейронних зв'язків, кидають виклик тривіального аналізу та інтерпретації аналітиком.

У Україні можна купити нейромережеві системи, такі як BrainMaker, NeuroShell, Deductor Neural Analyzer. Вартість їх досить висока.

Системи міркувань, засновані на подібних випадках

Ідея систем міркувань, заснованих на подібних випадках (CBR, MBR), полягає в тому, що для прогнозування майбутнього або прийняття правильного

рішення ці системи знаходять близькі аналоги існуючої ситуації в минулому і дають ту ж відповідь, яка була правильною для неї. Цей системний клас показує хороші результати в самих різних завдань. Їх недолік в тому, що вони не створюють моделей або правил, а узагальнюють попередній досвід. Вибір рішення грунтується на всьому спектрі наявних даних. Отже, не можуть бути вибрані певні параметри, на основі яких системи створюють своє рішення для обробки даних [33, 34]. Прикладами систем, що використовують CBR, є KATE-Tools, Cubist і Pattern Recognition Workbench.

Древо прийняття рішень

Цей метод підходить для вирішення завдань класифікації і тому використовується дуже обмежено – в областях, де необхідно вирішити задачі оптимізації або прогнозування. В результаті використання методу дерева прийняття рішень деревоподібна структура створюється за допомогою правил типу «якщо … то …». Щоб віднести об'єкт до певного класу, ви повинні відповісти на питання в вузлах дерева, починаючи з його кореня. Питання мають форму «Чи є значення параметра А більшим х?» Якщо відповідь позитивна, ми переходимо до правого вузла на наступному рівні, якщо негативна - до лівого вузла; потім процес повторюється знову, і так далі, поки ми не дійдемо до одного з останніх вузлів - листів, що вказують, до якого класу повинен бути присвоєний об'єкт.

Перевагою методу є наочний вид одержуваних правил. Однак значимість стоїть гостро для методу дерева прийняття рішень. Вона полягає в тому, що все менше і менше записів відповідає окремим вузлам на кожному новоствореному рівні в дереві - дерево ділить дані на велику кількість особливих випадків. Чим більше таких випадків, чим менше навчальних прикладів потрапляє в такий окремий випадок, тим менш надійною буде їх класифікація. Якщо побудоване дерево занадто «кущасте» - складається з недоречно великої кількості дрібних гілок - воно не дасть статистично достовірних відповідей. Як показує практика,

ця задача не знаходить задовільного рішення в більшості систем, що використовують дерева прийняття рішень [34]. Цей метод використовується в багатьох системах. Найбільш відомі C5.0, Clementine, SIPINA. В Україні є, зокрема, IDIS, Deductor Tree Analyzer Lite і PolyAnalyst.

Генетичні алгоритми

Видобуток даних - не основне застосування генетичних алгоритмів, яке слід розглядати в більших масштабах як спосіб вирішення комбінаторних та оптимізаційних задач. Наразі генетичні алгоритми були включені до стандартного набору інструментів для інтелектуальної обробки даних. Цей метод імітує природний процес відбору в природі. Необхідно знайти оптимальне рішення задачі з точки зору конкретного критерію. Нехай кожне рішення повністю описується низкою чисел або наборів нечислового характеру, тобто якщо є необхідність вибрати серію з фіксованою кількістю ринкових параметрів, які найбільше впливають на його динаміку, це буде серія назв цих параметрів. Цю здатність можна назвати набором хромосом, що визначають підготовленість особистості вирішувати задачі. Значення параметрів, що визначають рішення, називаються генами. Пошук оптимального рішення подібний до створення популяції особин, представлених їх хромосомними наборами. У цьому розвитку діють три механізми: відбір найсильнішого набору хромосом, яким відповідають найбільш ефективні рішення; схрещування - генерування нових особин шляхом змішування наборів хромосом від вибраних особин; мутація - випадкові зміни генів у деяких особин в популяції. В результаті зміни поколінь виникає рішення задачі, яке неможливо вдосконалити.

У роботі [35] представлений метод застосування генетичного алгоритму з метою отримання чітких знань про функціонування системи (правила «якщо … то …») інвертованого (перевернутого) маятника на прикладі вирішення задачі стабілізації.

З точки зору представленого аналізу, генетичні алгоритми мають дві слабкі сторони: одне лише формулювання задачі з точки зору термінології не дозволяє проводити статистичний аналіз значущості отриманого рішення; тільки фахівець може ефективно сформулювати завдання та визначити критерій відбору хромосом. Зокрема, процес створення першого набору хромосом, критерії вибору хромосом та використовувані процедури є евристичними та не гарантують пошуку «найкращого» рішення. Як і в реальному житті, еволюція може застрягти в непродуктивній гілці. І навпаки, можна навести приклади, в яких дві неадаптивні особини, яких генетичним алгоритмом було виключено з еволюції, можуть створити високопродуктивне потомство після декількох повторень. Це особливо актуально при вирішенні масштабних задач зі складними внутрішніми зв’язками. Через ці фактори генетичні алгоритми тепер слід розглядати як інструмент дослідження, а не як інструмент аналізу даних для практичного застосування в різних сферах людської діяльності. Продукт такого типу, що доступний в Україні - система GeneHunter групи Ward Systems.

Методи нелінійної регресії

Пошук залежності цільової змінної від решти здійснюється як функція певного типу. Наприклад, один із найефективніших алгоритмів цього типу - метод групового підходу до аргументів (MGHA) [36] - шукає взаємозв'язки у вигляді поліномів. Отримана картина залежності - поліном, є придатною для аналізу та інтерпретації (хоча на практиці вона все ще занадто складна). Однак чіткість отриманих взаємозв’язків набагато гірша, ніж логічне тлумачення залежностей, виявлених у формі «якщо … то …», що логічно зрозуміло досліднику. В даний час MGHA впроваджується інтелектуальними системами, пропонованими в системі NeuroShell групи Ward Systems.

Еволюційне програмування

На сьогоднішній день це наймолодша та найперспективніша галузь інтелектуального аналізу даних, реалізована, зокрема, в системі PolyAnalyst [37,

38]. Суть методу полягає в тому, що гіпотези про тип залежності цільових змінних від інших змінних формулюються системою у вигляді програм на внутрішній мові програмування. Якщо це універсальна мова, теоретично в ній може бути виражена будь-яка залежність. Коли система знаходить програму, яка точно виражає залежності, які вона шукає, вносить невеликі зміни та вибирає серед створених таким чином процедур, процедури, які підвищують точність. Таким чином, система будує кілька ліній генетичних програм, які змагаються за точність вираження бажаних взаємозв'язків. Спеціальний модуль системного перекладу перекладає знайдені залежності з внутрішньої мови системи на зрозумілу для користувача мову (математичні формули, таблиці тощо), щоб їх було легше отримати. Слід зазначити, що недоліки, властиві генетичним алгоритмам, такі як складність відбору особин для подальшого розвитку популяції та можлива зупинка подальшого еволюційного розвитку (часто трапляються в природі як види, що перебувають під загрозою зникнення), залишаються у розглянутого методу.

Обмежені алгоритми пошуку

Обмежені алгоритми пошуку були розроблені М.М. Бонгардом, щоб знайти логічні закономірності в даних. Вони ефективні при вирішенні задач у різних сферах [39-41].

Ці алгоритми обчислюють частоту комбінацій простих логічних подій у підмножинах даних. Приклади простих логічних подій: $x = a$; $x < a$; $x > a$; $a < x < b$ тощо, де x - будь-який параметр, a і b - константи.

Обмеженням є тривалість поєднання простих логічних подій. На основі аналізу розрахованих частот робиться висновок про придатність даної комбінації для створення асоціацій у даних, класифікації, прогнозуванні тощо.

Найяскравішим представником цього підходу є система WizWhy від WizSoft. Хоча автор системи не розкриває особливостей алгоритму роботи WizWhy, на основі результатів тестування системи були зроблені висновки про

наявність обмеженого пошуку. Автор WizWhy стверджує, що його система розпізнає всі логічні правила «якщо … то …» у даних. Насправді це твердження було перевірено в роботі [20, 42] і було доведено, що воно суперечливе. По-перше, максимальна довжина комбінації «якщо … то …» в системі WizWhy 6, як правило, а по-друге, з початку алгоритму виконується евристичний пошук простих логічних подій, на яких все додатково аналізується. Як тільки автори [43] зрозуміли ці особливості WizWhy, вони могли легко запропонувати найпростішу тестову задачу, яку система взагалі не могла б вирішити. Інше питання полягає в тому, що система генерує рішення для відносно невеликих даних протягом розумного періоду часу. Проте, все ж, ця система демонструє високу ефективність у вирішенні практичних задач.

Системи візуалізації багатовимірних даних

Певною мірою засоби графічного відображення даних підтримуються усіма системами аналізу даних. Водночас системи, що спеціалізуються лише на цій функції, займають значну частку ринку. Прикладом цього є DataMiner 3D від словацької компанії Dimension5.

У таких системах основний акцент робиться на простоті використання користувацького інтерфейсу, що дозволяє призначати різні параметри діаграми розсіювання об'єктів бази даних (записів) до аналізованих показників. Ці параметри включають колір, форму, орієнтацію щодо своєї осі, розміри та інші властивості графічних елементів зображення [44]. Крім того, системи візуалізації даних оснащені практичними інструментами для масштабування та обертання зображень. Переглядаючи дані з різних позицій, можна визначити деякі досить очевидні закономірності, але не можна побачити нетривіальні, складні взаємозв'язки з цим інструментом. Системи обробки зображень можуть коштувати кілька сотень доларів.

1.2. Обґрунтування вибору підходу до розробки інтелектуальної системи аналізу даних

Як інтелектуальна система аналізу даних, яка найкращим чином відповідає сучасним вимогам до аналітичних систем, у цій роботі пропонується вибрати клас систем, що використовують технологію аналізу нейронної мережі.

Таблиця 1.1

Відповідність аналітичних систем до вимог, встановлених до них, «+» - відповідають вимогам, «-» - не відповідають вимогам

Аналітична система	дані великих об'ємів	зашумлені дані	єдиний математичний апарат	знання математичного апарату	ясність результату
предметно-орієнтовані системи	+	-	-	+	+
статистичні пакети	+	+	+	-	-
нейромережеві пакети	+	+	+	+	-
системи на основі методу найближчого сусіда	-	-	+	+	-
системи на основі методу дерев прийняття рішень	+	-	-	+	+
системи на основі методів еволюційного програмування	+	+	+	-	-
системи обмеженого перебору	-	-	-	+	+

Причини такого вибору можна знайти в табл. 1.1 аналізу відповідності аналітичних систем їхнім вимогам та наявність переваг нейромережевого

підходу перед звичайними математичними методами: математичної статистики, кластерів, регресії, факторного аналізу тощо. Порівняно з цими методами, підхід, що заснований на використанні штучних нейронних мереж, має такі наступні переваги:

- створення математичної парадигми для всіх завдань: можна одночасно вирішити кілька класифікаційних або прогностичних задач з однієї мережі. Завдяки нейронним мережам з відносно невеликою кількістю нейронів можна вирішувати досить складні задачі з різних областей;
- нейронні мережі вивчають моделі, за допомогою яких їх легко перекваліфікувати при надходженні нових даних або перенавчити для обробки даних з іншої області застосування: якщо замінити етап програмування (налаштування) навчанням, система не повинна пред'являти жодних вимог користувачеві до знань математичного пристрою, щоб зробити роботу з аналітичною системою більш зрозумілою та доступною;
- у нейронних мережах може використовуватися будь-яка кількість незалежних і залежних функцій: кількість прикладів для різних класів (при вирішенні задачі класифікації) може змінюватися. Нейронна мережа має метод розрахунку важливості незалежних характеристик і можливості мінімізації їх кількості.

Ці переваги приносяться із запозичення ідей про те, як працює мозок. Штучні нейронні мережі, і в цілому вся нейроінформатика, з’явилися, коли були зроблені спроби змоделювати мозок на основі кібернетичних, а не нейробіологічних ідей. Тому весь підхід до нейронної мережі базується на ідеї побудови комп’ютерного пристрою з великою кількістю простих обробних елементів (нейронів), які обробляються паралельно. Ці нейрони функціонують незалежно один від одного і пов’язані між собою односторонніми каналами передачі інформації - синапсами.

Штучна нейронна мережа, побудована відповідно до цього підходу, має наступні переваги, аналогічні своїм біологічним прототипам [13, 14, 45-47]:

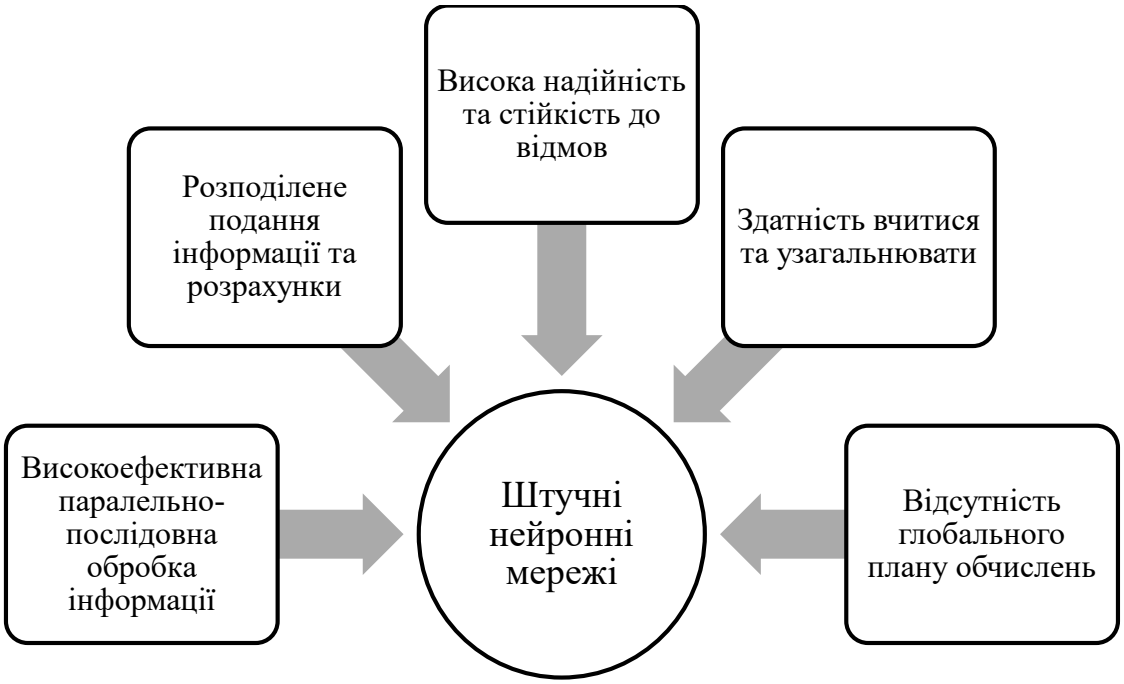


Рис. 1.2. Переваги штучних нейронних мереж

- Високоефективна паралельно-послідовна обробка інформації, максимальний потенціал паралелізму та найбільш ефективне використання архітектури паралельних обчислень у порівнянні з іншими обчислювальними технологіями. Величезний паралелізм нейрокомп'ютерів, необхідний для ефективної обробки зображень, забезпечує локальну обробку інформації в нейронних мережах. Кожен нейрон реагує лише на локальну інформацію, отриману в будь-який момент від тих самих нейронів, які до нього підключені, не посилаючись на загальну арифметичну схему, загальну для електронних обчислювальних машин. Таким чином, алгоритми нейронної мережі є локальними, і нейрони можуть працювати паралельно.
- Розподілене подання інформації та розрахунки.

- Висока надійність та стійкість до відмов окремих елементів, що складають нейронну мережу.
- Здатність вчитися та узагальнювати.
- Відсутність глобального плану обчислень у нейронних мережах також передбачає особливий спосіб їх програмування. Він також має локальний характер: кожен нейрон змінює свої адаптаційні параметри - синаптичні ваги - відповідно до місцевої інформації про ефективність роботи всієї мережі в цілому. Спосіб поширення такої інформації в мережі та відповідна адаптація нейронів є навчанням у природі. Цей метод програмування дозволяє ефективно враховувати особливості методу обробки даних, необхідного мережі. Алгоритм не визначений заздалегідь, а генерується із самих даних - прикладів, на яких тренується мережа. Таким чином, біологічні нейронні мережі розробили ефективні алгоритми обробки сенсорної інформації в процесі самонавчання [11, 48-50]. Процес програмування замінюється процесом навчання нейронної мережі. Привабливою особливістю нейро-обчислень є уніфікований принцип навчання нейронних мереж - мінімізація емпіричних помилок. Функція помилок для оцінки конкретної конфігурації мережі визначається зовні залежно від цілей навчання. Але потім мережа починає поступово змінювати свою конфігурацію - стан усіх своїх синаптичних ваг - щоб мінімізувати цю помилку. Завдяки цьому в процесі навчання мережа краще справляється з дорученим їй завданням. Образно кажучи, цей процес можна назвати пошуком мінімуму функції помилки $H(\alpha)$, що залежить від набору всіх синаптичних ваг мережі α.

Висновки до розділу 1

У цьому розділі представлено вимоги до сучасних систем інтелектуального аналізу даних, засновані на методах та алгоритмах, що використовуються в аналітичних системах, класифікація цих систем, показано переваги та недоліки

вибраних класів систем інтелектуального аналізу даних. Було зроблено цілеспрямований вибір системи аналізу даних, яка найкраще відповідає вимогам аналітичної системи - це клас систем інтелектуального аналізу даних, заснованих на парадигмі нейронної мережі.

Цей клас аналітичних систем може працювати з великими за обсягом зашумленими даними, може використовуватися для вирішення різноманітних задач, не вимагає спеціальних знань від користувача (процес конфігурації замінюється етапом навчання). Проте має недолік – неможливість тривіального аналізу та простої інтерпретації результатів, які дає нейронна мережа, що були зафіксовані у вагах міжнейронних зв’язків. У цьому контексті необхідно вдосконалити метод аналізу даних нейронної мережі таким чином, щоб забезпечити вилучення однозначної інформації з даних у вигляді логічних правил функціонування перевіреної системи.

РОЗДІЛ 2

КЛАСИФІКАЦІЯ МЕТОДИК АНАЛІЗУ ВЕЛИКИХ ДАНИХ

2.1. Загальна методика аналізу даних

Перш ніж створювати модель нейронної мережі для аналізу даних, опишемо один із можливих підходів до вирішення задачі інтелектуального аналізу даних.

Дослідник часто зустрічається з тим, що теоретичні можливості аналітичних методів на практиці не застосовуються до певної системи або вимагають неадекватно тривалих фінансових та інтелектуальних витрат для реалізації.

У більшості випадків аналітик стикається з ситуацією, коли важко сформулювати чіткі припущення щодо досліджуваної системи. Модель невідома, а єдиним джерелом інформації для її побудови є таблиця експериментальних даних «вхід-вихід», кожен рядок якої містить значення вхідних властивостей об'єкта та відповідні значення вихідних властивостей.

В результаті дослідник вимушений використовувати всі види евристичних або експертних припущень щодо вибору інформаційних ознак, класу моделі та параметрів обраної моделі. Ці припущення аналітика ґрунтуються на його досвіді, інтуїції та проникненні у сенс аналізованого процесу. Висновки, зроблені з цього підходу, базуються на простій, але фундаментальній гіпотезі про монотонність простору рішень, яка може бути виражена наступним чином: «Подібні ситуації на вході призводять до подібних реакцій системи на вихід». Ідея інтуїтивно зрозуміла, і її зазвичай достатньо для отримання практично прийнятних рішень у кожному окремому випадку [51, 52].

За допомогою цього методу висока точність рішення присвячується реальній ситуації. Процес вилучення знань з даних відбувається за схемою,

подібною до встановлення законів фізики: збір експериментальних даних, їх упорядкування в таблиці та пошук моделі міркувань, яка:

- показує отримані результати,
- дає можливість прогнозувати нові факти.

Це враховує, що наявні знання про аналізовану систему, як і про будь-яке фізичне явище, є певною мірою приблизними. Загалом, будь-яка реальна система міркувань передбачає різні типи наближення. Робота [53] визначає термін видобування даних як спробу узаконити фізичний підхід до вирішення задач аналізу даних, на відміну від математичного підходу.

Фізичний підхід - це такий підхід, при якому аналітик готується до того, що аналізована система є занадто заплутаною та непридатною для точного аналізу із застосуванням суворих аналітичних методів. Однак все-таки можна скласти хорошу картину її поведінки за різних обставин, підходячи до задачі з різних точок зору, базуючись на знаннях предметної області, досвіді, інтуїції та використанні різних евристичних підходів. У цьому випадку відбувається перехід від наближеної моделі до все більш точних уявлень про аналізовану систему. Загальна схема роботи наведена на рис. 2.1.

Таким чином, аналіз даних передбачає наступне:

- аналіз повинен ґрунтуватися на досвіді експерта;
- потрібно розглянути задачу з різних сторін;
- не слід одразу прагнути високої точності: необхідно знайти рішення від більш простих і грубих моделей до все більш складних і точних;
- слід зупинитися, як тільки отримуємо прийнятний результат, а не намагатися отримати ідеальну модель;
- з часом і накопиченням нової інформації цикл повинен повторюватися.

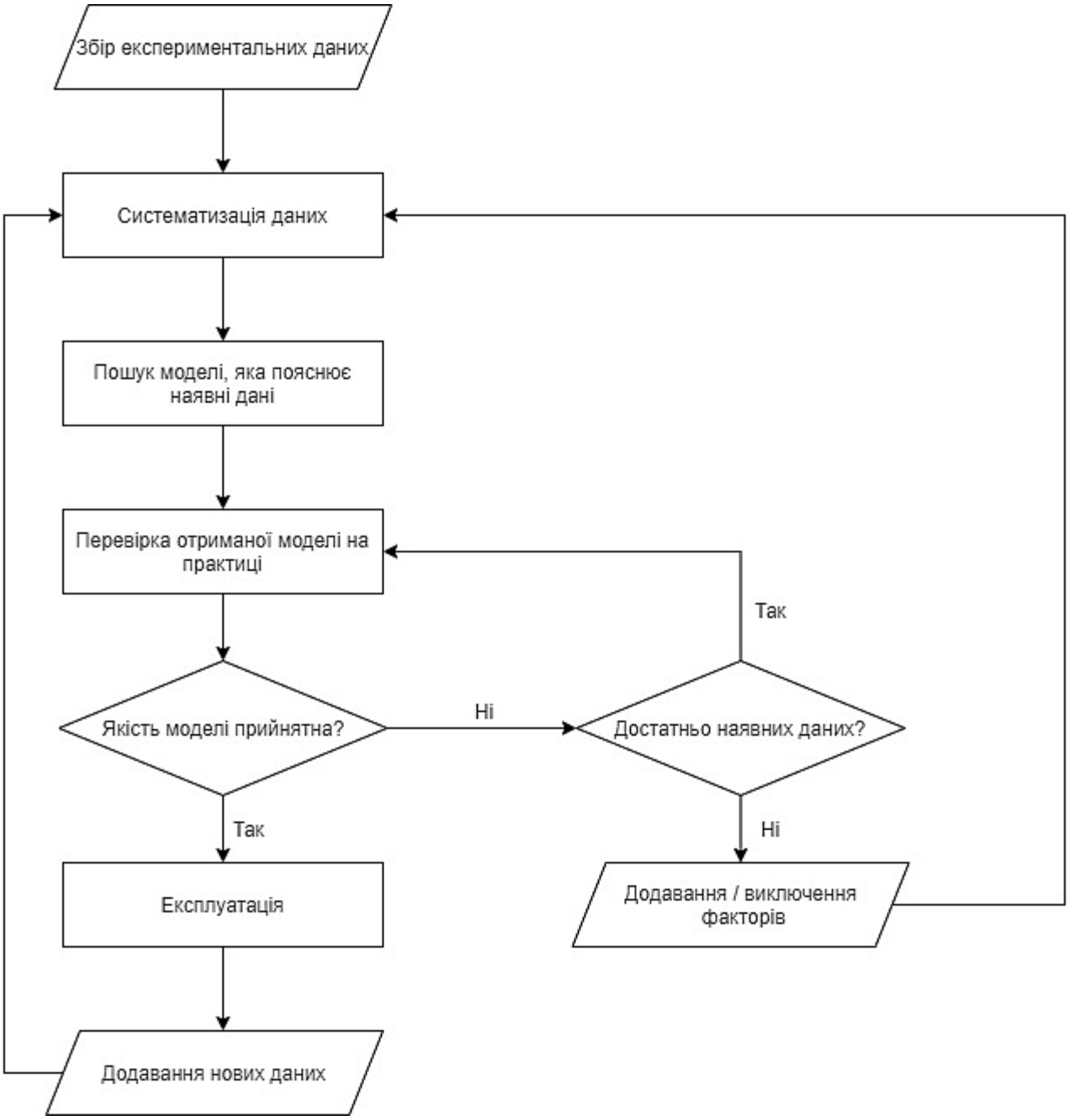


Рис. 2.1. Загальна схема аналізу даних

2.2. Основні особливості та класифікація штучних нейронних мереж

Давайте опишемо, що таке штучна нейронна мережа, за якими законами вона працює і як вона вирішує задачі.

Нейронні мережі - це мережі взаємопов’язаних простих елементів, які називаються нейронами.

Особливістю нейронних мереж є глобальний характер зв’язків. Основні елементи штучних нейронних мереж - формальні нейрони - спочатку спрямовані на роботу з широкосмуговою інформацією. Кожен нейрон нейронної мережі, як правило, підключений до всіх нейронів попереднього шару (рис. 2.2). Це основна відмінність між формальними нейронами та основними елементами послідовних ЕОМ - логічні ворота лише з двома входами. Тому процесори загального призначення мають складну архітектуру, засновану на ієрархії модулів, кожен із яких має певну функцію. Навпаки, архітектура нейронних мереж проста і вичерпна. Зв'язки спеціалізуються на етапі навчання під впливом конкретних даних [54-67].

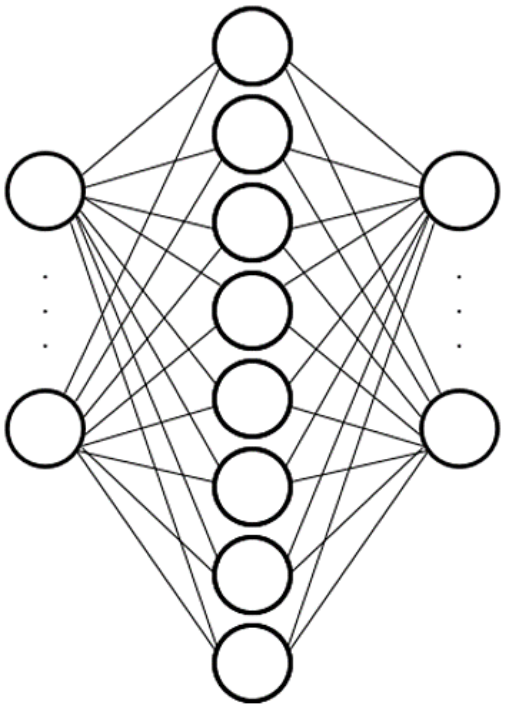


Рис. 2.2. Глобальність зв’язків у штучних нейронних мережах

В основі використовуваних концепцій лежить ідея про те, що нейрони можна моделювати за допомогою відносно простих автоматів, а загальна складність мозку, гнучкість його функціонування та інші важливі властивості визначаються зв'язками між нейронами. Кожне з'єднання представляється як простий елемент, що використовується для передачі сигналу. Кінцевим виразом цієї точки зору може бути гасло: «Структура зв’язків - це все, властивості елементів - ніщо» [66].

У нейронній мережі виділяються вхідний і вихідний рівні. Нейронна мережа отримує інформацію через вхідні рецептори, а потім генерує вихідні сигнали, передаючи цю інформацію через себе і перетворюючи її за допомогою структурних елементів.

Вибір топології нейронної мережі - це окремий крок у роботі з нейронною мережею, який має критичний вплив на всі подальші дії. Давайте класифікуємо парадигми нейронної мережі за структурою.

Серед топологій штучних нейронних мереж існує дві основні архітектури: [68-72] – багатошарові мережі (мережі прямого поширення) та повністю зв’язні (рекурентні, із зворотним зв’язком) мережі.

Загалом, можливі варіанти класифікації нейронних мереж представлені в табл. 2.1.

Таблиця 2.1

Класифікація нейронних мереж

За наявністю зворотніх зв'язків	За типом сигналів	За типом навчання	За однорідністю
Мережі прямого поширення сигналу	Двійкові	З учителем	Гомогенні
Рекурентні мережі	Неперервні	Без учителя	Гетерогенні

Візуальне зображення основних топологій нейронної мережі показано на рис. 2.3.

Багатошарові мережі: нейрони розташовані в кілька шарів (рис. 2.3). Нейрони першого шару вловлюють вхідні сигнали, трансформують їх і направляють через розгалуження до нейронів другого шару. Потім на k-й рівень, який забезпечує вихідні сигнали, запускається друга зміна тощо. Якщо не вказано інше, кожен вихід з i-го шару надходить на вхід усіх нейронів $(i+1)$-го шару. Кількість нейронів у кожному шарі може бути довільною і жодним чином не пов'язана заздалегідь з кількістю нейронів в інших шарах.

Стандартний спосіб застосування вхідних сигналів полягає в тому, що всі нейрони першого шару приймають кожен вхідний сигнал. Типовими прикладами топологій багатошарової мережі є одношаровий персептрон, багатошаровий персептрон і мережа радіальних базисних функцій. Мережі прямого поширення є статичними в тому сенсі, що вони генерують набір вихідних значень для даного входу, які не залежать від попереднього стану мережі.

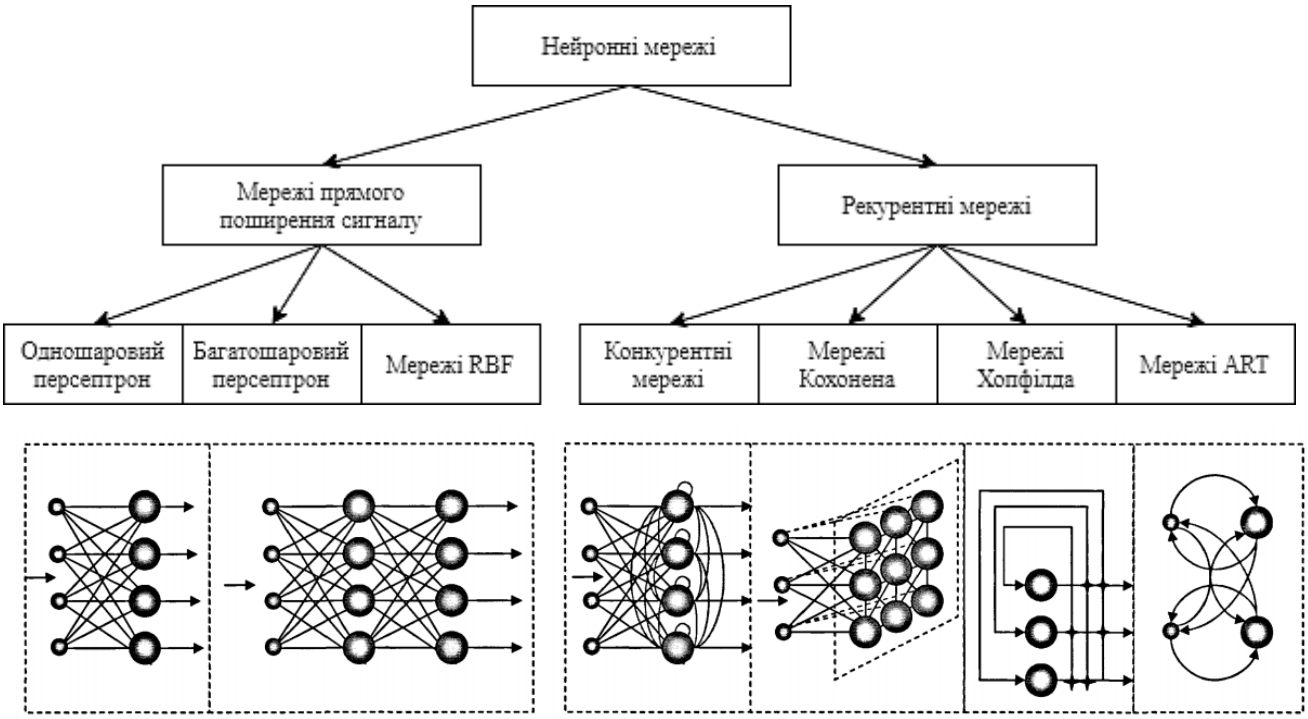


Рис. 2.3. Топології нейронних мереж

Повнозв’язні мережі: кожен нейрон надсилає свій вихідний сигнал іншим нейронам, включаючи власний. Вихідними сигналами можуть бути всі або деякі вихідні сигнали нейронів через кілька циклів функціонування мережі. Всі вхідні сигнали надходять на всі нейрони. Прикладами таких архітектур є конкурентні мережі, самоорганізовані карти (мапи) Кохонена та мережі Хопфілда. Рекурентні мережі є динамічними, оскільки зворотний зв'язок змінює вхідні нейрони всередині них, що призводить до зміни стану мережі.

Таблиця 2.2

Застосування нейронних мереж до розв'язання задач, де «+» – можна застосувати, «-» – не можна застосувати

Мережа / задача	Асоціативна пам'ять	Стиснення інформації	Прогнозування	Оптимізація	Класифікація	Кластеризація
Мережа Хопфілда	+	-	-	+	-	-
Мережа зустрічного поширення	+	+	-	-	-	-
Мережа радіального базису	-	-	+	-	+	-
Карта (мапа) Кохонена	-	-	-	+	+	+
Багатошаровий персептрон	+	+	+	-	+	-
Двонаправна асоціативна пам'ять	+	-	-	-	-	-
Мережа Хеммінга	+	-	-	-	+	-
Ймовірнісна мережа	-	-	-	-	+	-
Мережа адаптивного резонансу	-	-	-	-	+	+

У табл. 2.2 показана можливість використання нейронних мереж для вирішення різних задач.

Елементи багаторівневих і повнозв'язних мереж можна вибрати декількома способами. Однак є стандартний вибір - нейрон з неоднорідним адаптивним лінійним суматором на вході.

2.3. Схема аналізу даних за допомогою штучних нейронних мереж

Опишемо схему аналізу даних, представлену на рис. 1.1, з точки зору використання механізму нейронної мережі як структури системи, що пояснює дані, які досліджуються. Принципова схема запропонованого підходу показана на рис. 2.4. Етапи показані у порядку їх виконання. Однак, якщо на поточному етапі отримується незадовільний результат, іноді доводиться повертатися до одного із попередньо завершених етапів. Варіанти переходу кроку обговорюються більш докладно в окремому описі кроку у відповідному підрозділі. Розглянемо короткий опис представленої схеми.

На першому етапі методології на основі експертного висновку визначається надлишковий набір факторів, які повністю описують досліджуваний об'єкт, збирається інформація в розділах вибраних факторів та проводиться попередня обробка систематизованої інформації. Цей крок є в основному формулюванням задачі, яку потрібно вирішити. Це єдиний етап представленої схеми аналізу даних, реалізація якої залежить від області застосування.

На другому етапі визначається топологія, вибирається тип зв’язків, відбирається кількість шарів і кількість нейронів у кожному шарі, визначається функція активації нейронів та принцип навчання нейронної мережі. У цій роботі пропонуються методи встановлення цих параметрів за замовчуванням та автоматичної їх зміни, прозорі для дослідника, без необхідності глибоких знань технології нейронних мереж.

Етап 1. Попередній аналіз даних досліджуваної задачі

1. Визначення набору факторів, які значимо описують досліджувану систему	2. Накопичення і систематизація експериментальних даних	3. Стандартизація даних: відмова від використання абсолютних значень, нормування, нормалізація

Етап 2. Визначення параметрів нейронної мережі

1. Вибір топології: вибір типу мережі, кількості шарів і нейронів в кожному шарі	2. Вибір правила навчання	3. Вибір функції активації нейронів

Етап 3. Навчання нейронної мережі

1. Формування підмножин прикладів: навчальна та тестова вибірки	2. Безпосередньо процес навчання	3. Перевірка ефективності навчання

Етап 4. Спрощення нейронної мережі

1. Зниження розмірності входів	2. Оптимізація кількості шарів	3. Оптимізація кількості нейронів у шарах	4. Оптимізація загальної кількості синапсів

Етап 5. Витяг правил з нейронної мережі

1. Обмеження значень ваг нейронів	2. Структура таблиці ваг	3. Отримання правил в явному вигляді	4. Оптимізація отриманих правил

Рис. 2.4. Схема аналізу даних з використанням нейромережевого підходу

На третьому етапі створюються навчальні приклади, проводиться навчання мережі та оцінюється якість синтезованої моделі.

Четвертий крок - довести нейронну мережу до так званої логічно зрозумілої форми. Як випливає з назви, такі мережі більш готові представити виявлений причинно-наслідковий зв’язок в аналізованих даних у зрозумілій для дослідника формі.

Завершальним етапом роботи інтелектуальної системи аналізу даних, заснованої на підході нейронної мережі, є представлення аналітику сукупних залежностей ваг нейромережевих з'єднань у наочній, простій та зрозумілій формі. На цьому етапі розглядаються різні підходи до вирішення задачі «чорного ящика» [72-78].

Представлена схема відповідає всім вимогам до аналітичних систем:

- її можна використовувати для вирішення різних задач у багатьох областях застосування;
- для розв’язування задач використовується один математичний апарат;
- для того, щоб мати можливість працювати з системою, досліднику не потрібні ґрунтовні знання використовуваного математичного апарату;
- для аналізу можна використовувати зашумлені дані;
- відсутність принципових обмежень щодо типу введеної інформації;
- система відображає знайдені відношення у формі, зрозумілій досліднику.

Важливою перевагою запропонованої схеми аналізу є здатність вирішувати задачі, в яких пошук аналітичних зв'язків між вхідними та вихідними даними є складним або неможливим.

Обмеження у використанні цього підходу в основному зумовлені специфічністю завдань, що підлягають вирішенню відповідним математичним апаратом. Перелік існуючих обмежень:

- Потрібні навички збору інформації. База аналізованих експериментальних даних є очевидною. Для ефективного синтезу взаємозв’язків

у даних, тобто для побудови інформаційної моделі аналізовані дані повинні узгоджуватися.

- У гіпотетичній ситуації традиційні схеми показують кращі результати порівняно із запропонованою схемою аналізу, якщо відома функція системи або відомий алгоритм обчислення будь-яких значень аргументів.
- Отриманий набір логічно зрозумілих правил є повним лише в межах досліджуваного набору даних. Це не вичерпний опис того, як працює система.

Висновки до розділу 2

У розглянутому розділі була описана загальна схема аналізу даних, на основі якої була розроблена система аналізу даних із використанням нейромережевих технологій. В рамках запропонованої схеми було описано хід аналізу даних у системах інтелектуального аналізу даних на базі нейронних мереж.

Розглянуто поняття штучної нейронної мережі, за якими законами вона працює, як вирішує задачі та які розрізняють топології нейронних мереж.

Проведено аналіз можливості використання нейронних мереж для вирішення різних задач та описано клас задач, для вирішення яких оптимальним є застосування запропонованої технології.

Було побудовано схему аналізу даних з використанням нейромережевого підходу та детально розглянуто всі етапи роботи системи.

Виділено існуючі обмеження системи, що полягають у наявності навичок збору інформації, погіршенні результату порівняно з іншою системою, якщо відомий алгоритм обчислення будь-яких значень та повнота набору логічно зрозумілих правил, отриманих після аналізу, лише в межах досліджуваного набору даних.

РОЗДІЛ 3

РОЗРОБКА МЕТОДИКИ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ЕМПІРИЧНИХ ДАНИХ

3.1. Формулювання задачі аналізу даних

Першим кроком в аналізі даних є формулювання задачі. Як зазначалося раніше, це єдиний етап запропонованої схеми аналізу даних, який, очевидно, залежить від області застосування при її реалізації.

Нейронні мережі здавна використовуються в різних сферах людської діяльності. За допомогою штучних нейронних мереж вирішуються наступні задачі [79]:

Класифікація. Задача полягає в тому, щоб вказати належність вхідного образу (наприклад, мовного сигналу чи рукописного символу), представленого вектором ознак, до одного або декількох заздалегідь визначених класів. Відомі програми включають розпізнавання літер, розпізнавання мови, класифікацію сигналів на електрокардіограмі та класифікацію клітин крові.

Кластеризація. При вирішенні задачі кластеризації, також відомої як «некерована» класифікація зображень, не існує такого поняття, як шаблон навчання з назвами класів. Алгоритм кластеризації базується на подібності зображень і розміщує близькі образи в кластері. Відомі випадки, коли кластеризація використовується для отримання знань, стиснення даних та вивчення властивостей даних.

Апроксимація функцій (функціональне наближення). Припустимо, у нас є навчальний набір $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$ (пари даних вхід-вихід), сформований невідомою функцією $y = f(x)$, яка спотворена шумом. Задача апроксимації полягає в знаходженні оцінки невідомої функції $f(x)$.

Апроксимація функції необхідна для вирішення багатьох технічних та наукових задач, пов'язаних із моделюванням.

Передбачення / прогнозування. Нехай n дискретних зразків $\{y(t_1), y(t_2), \dots, y(t_n)\}$ подано в послідовні моменти часу $t_1, t_2, \dots, t_n$. Задача полягає у прогнозуванні значення $y(t_{n+1})$ у деякий момент часу t_{n+1}. Передбачення / прогнозування має значний вплив на прийняття рішень у бізнесі, науці та техніці. Прогнозування цін на акції та прогноз погоди є загальним застосуванням методів передбачення / прогнозування.

Оптимізація. Численні задачі математики, статистики, техніки, науки, медицини та економіки можна розглядати як задачі оптимізації. Задачею алгоритму оптимізації є пошук рішення, яке задовольняє системі обмежень і максимізує або мінімізує цільову функцію. Задача комівояжера, що віднесена до класу NP-повних, є класичним прикладом задачі оптимізації.

Управління. Розглянемо динамічну систему, визначену набором $\{u(t), y(t)\}$, де $u(t)$ - це дія, що контролює вхід, а $y(t)$ - вихід системи в момент часу t. У системах управління з еталонною моделлю метою управління є обчислення такої вхідної активності $u(t)$, в якій система слідує бажаній траєкторії, визначеній еталонною моделлю. Одним з прикладів є оптимальне управління двигуном.

Слід зазначити, що використання нейромережевого підходу виправдано при вирішенні задачі аналізу лише однієї складної системи. Коли відома функція системи, найкращим інструментом моделювання є традиційні засоби математичного аналізу, і немає необхідності використовувати парадигму нейронної мережі.

Дамо формальний опис складної системи на основі [80-82]. Ця система складається з декількох компонентів. Нашою кінцевою метою буде побудова моделі системи, яка описує її поведінку та має прогностичні властивості, а на

основі аналізу моделі визначає принципи роботи системи. Модель у багатьох додатках може замінити випробувану систему, а отримані в результаті принципи поведінки моделі адекватно описують функціонування самої системи.

Кожен елемент системи має свої властивості та поведінку залежно від власного стану та зовнішніх умов. Якщо всі можливі прояви системи зводяться до суми проявів її компонентів, така система проста, хоча кількість її компонентів може бути великою. Традиційно для опису простих систем використовуються аналітичні методи, які полягають у поступовому поділі системи на компоненти та створенні компонентних моделей, які стають дедалі простішими. В основному це метод математичного моделювання [83], в якому моделі описуються як рівняння, а їх рішення базується на прогнозуванні поведінки системи. Для простих систем використання нейромережевого підходу може бути виправданим лише тоді, коли кількість простих елементів досить велика і обчислення кожного елемента вимагає значної кількості часу. У такій ситуації може бути корисно використовувати нейронні мережі для прискорення прийому результату роботи системи. Однак сформульовані правила поведінки системи за повнотою та якістю опису поступаються формулам та алгоритмам, які можна отримати за допомогою математичного моделювання.

Сучасні технічні системи (наприклад, порт, залізниця, інженерні споруди, приладобудування, транспортні засоби, багато медичних задач тощо) наближаються до рівня складності, коли їх спостережувана поведінка та спостережувані властивості не зводяться до простої суми властивостей окремих компонентів, що підлягають зменшено. Коли компоненти об’єднуються в систему, створюються якісно нові властивості, які неможливо визначити шляхом аналізу властивостей компонентів.

У випадку залізниць незначні відхилення в роботі окремих станцій, незначні зміни або збої в розкладі руху поїздів, відхилення на етапі організації процесу завантаження та розвантаження на станції можуть призвести до якісно

нової поведінки всієї залізничної системи, тобто заторів. Виникнення заторів має протилежний вплив на режими роботи компонентів, що може призвести до серйозних аварій тощо. Стан заторів неможливо повністю визначити з окремого аналізу, наприклад, характеристик ліній однієї станції. Однак нормальна робота цього шляху в системі може призвести до заторів.

Системи, де основні властивості можуть бути втрачені після вилучення компонентів, а якісно нові властивості з'являються після додавання компонентів, вважаються складними. Модель складної системи, заснована на принципах традиційного аналізу, буде безповоротно неадекватною для аналізованої системи, оскільки її якісні особливості будуть втрачені при поділі системи на окремі компоненти [84].

Одним із можливих способів виходу із ситуації є створення моделі на основі синтезу компонентів. Синтетичні моделі є практично єдиною альтернативою в соціології, довгостроковому прогнозуванні погоди, макроекономіці та медицині. Останнім часом для вивчення технічних та інженерних систем часто використовують синтезовані інформаційні моделі. У багатьох додатках інформаційно-математичні компоненти можуть утворювати єдину модель (наприклад, зовнішні умови описуються вирішенням рівнянь математичної фізики, а реакція системи - інформаційною моделлю).

Основним принципом інформаційного моделювання є принцип «чорного ящика». На відміну від традиційного аналітичного підходу, що моделює внутрішню структуру системи, метод синтетичного «чорного ящика» імітує зовнішню роботу системи. З точки зору користувача моделі, структура системи прихована в «чорному ящику», що імітує поведінку системи.

Кібернетичний принцип «чорного ящика» був розроблений в рамках теорії ідентифікації системи [85], яка пропонує широкий параметричний клас основних функцій або рівнянь для створення системної моделі, а сама модель синтезується шляхом взяття параметрів найкращого значення із умов для даної функції

відповідного рівняння поведінки системи. У цьому випадку структура системи жодним чином не відображається на структурі модельних рівнянь.

Функціонування системи в контексті синтезованої моделі описано лише в інформаційних цілях, на основі експериментальних даних або спостережень реальної системи. Перш ніж пропонувати підхід до відкритого «чорного ящика» до вилучення принципів системи, інформаційні моделі, як правило, програвали формальним математичним моделям та експертним системам з точки зору ступеня «пояснюваності» результатів, але не мали обмежень щодо результатів змодельованої системи та їх важливе практичне значення. З появою можливості виведення правил, які логічно зрозумілі аналітику з навченої нейронної мережі (синтезованої моделі), з’являються принципово нові можливості для розуміння того, як працює складна система.

Специфіка та складність запропонованих завдань визначає відмінності в поведінці системи та її моделі, які синтезуються в синаптичних вагах нейронних мереж [86-90]:

- Набір правил роботи системи, отриманих із навченої нейронної мережі, є неповним. Загалом, вхідні та вихідні простори імен не можуть містити всіх параметрів, необхідних для опису поведінки системи. Це пов'язано як з технічними обмеженнями, так і з обмеженнями нашого розуміння змодельованої системи. Чим більшою є кількість змінних, тим жорсткіші вимоги до обсягу експериментальних даних, необхідних для побудови моделі. Вплив опущених (прихованих) вхідних параметрів може порушити унікальність змодельованої функції системи.

- Експериментальна база даних, на якій базується модель, вважається зовнішньою. У той же час дані завжди містять похибки різного типу, шум та розбіжності між окремими вимірами. За винятком простих випадків, не можна повністю виключити спотворення даних.

- Експериментальні дані зазвичай мають будь-який розподіл у просторі змінних задачі. Як результат, отримані моделі мають неоднорідну надійність і точність у різних областях зміни параметрів.
- Експериментальні дані можуть містити пропущені значення (наприклад, через втрату інформації, збій вимірювальних датчиків, неможливість виконати весь набір аналізів тощо). Будь-яке тлумачення цих значень, у свою чергу, погіршує властивості моделі.

Ці особливості в даних та при формулюванні задач вимагають особливого підходу до норм і правил, прийнятих для функціонування системи.

3.2. Підготовка даних до інтелектуального аналізу

Збір і систематизація експериментальних даних.

Збір експериментальних даних - важлива частина першого етапу аналізу даних. Як згадувалося раніше, основна задачає, що постає при зборі даних - це можливість отримати інформацію про те, як працює система. Технічна сторона цього процесу тісно пов'язана з теорією баз даних і не викликає особливих питань. Теорія реляційних баз даних вирішила багато задач, пов'язаних з організацією надійного зберігання інформації, і забезпечила швидкий доступ до неї та можливість одночасного доступу. При зборі і систематизації обмеженого обсягу інформації можна використовувати формати файлових баз даних, в той час як при побудові більшого сховища даних для компаній переважно використовують реляційні або багатовимірні бази даних, підтримка яких забезпечується аналізом операційних даних (OLAP) від основних виробників, таких як IBM (DB2), Microsoft (Microsoft SQL Server), Oracle, вартість яких варіюється від одиниць до сотень тисяч доларів, в залежності від обсягу поставки. При виборі тієї чи іншої системи зберігання даних [91-99] можна застосувати принципи Кодда [100]. У світлі можливостей побудови інтелектуальної системи

аналізу даних на основі описаних технологій зберігання інформації можна уявити загальну структуру проектованої системи (рис. 3.1).

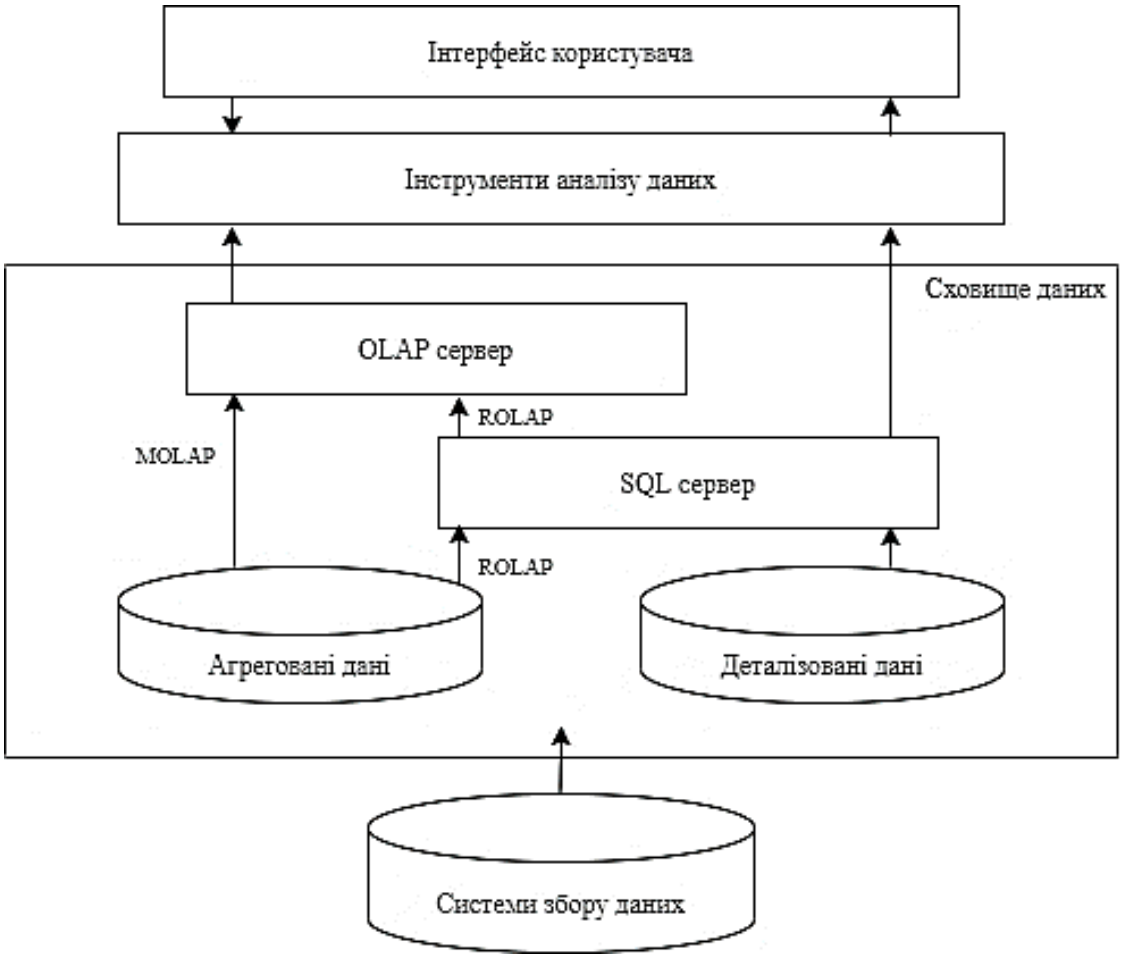


Рис. 3.1. Структура інтелектуальної системи аналізу даних

Попередня обробка даних

Останнім кроком першого етапу аналізу даних (рис. 2.5) є стандартизація даних: відмова від абсолютних величин, масштабування, нормалізація. Правильна попередня підготовка даних є ключовим фактором для успішного вирішення задач, запропонуємо новий спосіб цієї підготовки. Основною метою цього етапу є збільшення інформаційного наповнення вхідних даних. Для цього через попередню обробку даних необхідно отримати розподіл кожного компонента вхідних даних, який є майже рівномірним, що призводить до

зменшення ентропії, а згідно з теоремою Шеннона кількість інформації в даних обернено пропорційна їх ентропії. Інформативний зміст прикладів можна збільшити, рівномірно розподіливши вхідні дані. Для цього в методиці, розробленій в дисертації, було запропоновано ряд процедур: відмова від використання абсолютних значень вхідних параметрів та перехід до відносних, нормування даних (лінійна або заснована на статистичних характеристиках ряду), а потім нелінійне нормування із змінним типом нелінійності.

Як правило, на практиці необхідно представляти набір навчальних даних у вигляді, доступному нейронній мережі. Попередня обробка даних може стати найбільш трудомістким компонентом аналізу нейронних мереж. Це пов’язано з тим, що алгоритми нейронних мереж вже вбудовані в аналітичні програми, і процес вирішення прикладних задач, включаючи підготовку даних, є виключною відповідальністю дослідника.

Опишемо весь технологічний ланцюжок, тобто необхідні етапи підготовки даних:

- Вибір найважливіших записів.
- Кодування вводу-виводу: нейронні мережі можуть працювати лише з числами.
- Нормування даних: результати нейроаналізу не повинні залежати від вибору одиниць вимірювання.
- Підготовка даних: видалення очевидних закономірностей з даних полегшує нейронним мережам виявляти нетривіальні закономірності та усувати неоднорідності та невідповідності інформації.

Розглянемо ці кроки детальніше для попередньої обробки даних. Хоча попередня обробка безпосередньо не пов'язана з нейронними мережами, вона є однією з важливих складових запропонованого підходу до аналізу даних. Успішне навчання нейронної мережі може по суті залежати від форми, в якій подається інформація, необхідна для її навчання.

Основним принципом такої попередньої обробки є збільшення інформативного змісту прикладів з метою підвищення ефективності навчання.

Важливою задачею на етапі попередньої обробки є вибір вхідних змінних, які є найбільш важливими для прогнозів. Для отримання такого значущого набору вхідних даних існують методи оцінки значущості вхідних даних. Ці методи визначають лінійне або нелінійне значення вхідних даних.

Значимість входів у лінійній моделі передбачає лінійну залежність результатів від входів:

$$\left(y_j^\alpha - \bar{y}_j\right) = \sum\nolimits_k w_{jk}\left(x_k^\alpha - \bar{x}_k\right)$$

Якщо ми опускаємо k-й компонент вхідних даних при визначенні результатів і замінюємо його середнім значенням цих змінних, модель стає більш грубо визначеною, тобто похибка збільшується на величину [101-104]:

$$\delta E = \frac{1}{P}\sum\nolimits_{j\alpha}\left(y_j^\alpha - \bar{y}_j^\alpha\right)^2 = \frac{1}{P}\sum\nolimits_{j\alpha}\left(w_{jk}(x_k^\alpha - \bar{x}_k)\right)^2 = \overline{(x_k^\alpha - \bar{x}_k)^2}\sum\nolimits_j\left(w_{jk}\right)^2 = \sum\nolimits_j\left(w_{jk}\right)^2$$

(припускаючи, що дані нормуються до їх дисперсії). Таким чином, значимість k-го входу визначається сумою квадратів відповідних ваг.

Лінійна модель може бути створена без участі нейронних мереж. Сила нейроаналізу полягає в його здатності знаходити більш складні нелінійні взаємозв'язки. Іноді доцільно спочатку усунути відомі лінійні залежності, щоб полегшити правильний нелінійний аналіз. Вищезазначений підхід не застосовується для визначення нелінійної значущості вхідних даних. Для вирішення цієї задачі можна скористатися алгоритмом підрахунку ящиків (box-counting).

Алгоритми підрахунку ящиків засновані на підрахунку кількості заповнення прикладами P_i-комірок, на які спеціально розділений простір змінних

$X \otimes Y$. Ці числа використовуються для оцінки щільності ймовірності розподілів прикладів, що розподіляються в комірках. Щоб визначити значимість вхідних даних, ви повинні оцінити передбачуваність результатів, наданих певним набором полів введення. Чим вища передбачуваність, тим кращий відповідний набір входів. Метод підрахунку ящиків не робить припущень щодо характеру взаємозв'язку між входами та результатами. Отже, ця методика пропонує найбільш загальний рецепт для визначення важливості вхідних даних, одночасно дозволяючи оцінити ступінь передбачуваності результатів. Важливість техніки підрахунку ящиків (box-counting) полягає в тому, що можна швидко передбачити якість рішення, не знаходячи рішення самостійно [105].

Зазвичай набір полів виводу є фіксованим, і необхідно вибрати найкращу комбінацію з обмеженої кількості полів вводу. Вищезазначені методи оцінки значущості вхідних даних дозволяють систематично здійснювати початковий відбір вхідних змінних перед тренуванням нейронної мережі. Існує дві стратегії автоматичного створення простору об’єктів: послідовне додавання найважливіших входів та метод ортогоналізації.

Послідовне додавання найважливіших входів - це можливість формувати простір об’єктів з урахуванням фактичного значення входів - поетапно вибираючи найважливіші входи в якості подальших функцій. Перша ознака - це запис із найвищою індивідуальною значимістю, друга ознака - це запис, який пропонує найбільшу передбачуваність у поєднанні з уже вибраною тощо. Цей метод не гарантує пошуку найкращої комбінації вхідних даних, оскільки насправді враховується лише частка від загальної кількості комбінацій вхідних даних, і значимість кожної нової функції залежить від попередньо зробленого вибору. Однак вичерпний пошук практично неможливий. Іншим недоліком вищеописаного підходу є необхідність обчислювати перехресну ентропію в просторі дедалі більших розмірів із збільшенням кількості обраних ознак.

Метод ортогоналізації позбавлений цього недоліку, оскільки заснований на використанні техніки підрахунку ящиків лише в невеликих множинах.

Наступна систематична процедура дозволяє ітеративно вибирати найважливіші ознаки, які є лінійними комбінаціями вхідних змінних:

$$\tilde{X} = W \cdot X$$

(підмножина входів - це окремий випадок лінійної комбінації, тобто формально можна знайти краще рішення, ніж доступне, вибравши найважливіші комбінації входів).

Щоб визначити значення кожного вхідного компонента, використаємо індивідуальне значення цього входу кожного разу: $I(Y, X_k)$.

Після обчислення індивідуального значення входів, у вихідному вхідному просторі ми знаходимо напрямок, що відповідає найбільшій (нелінійній) чутливості виходів до змін входів. Цей напрямок градієнта визначає перший вектор ваги, який вказує на перший компонент простору ознак:

$$w_{1k} = I(Y, X_k)$$

Наступний компонент слід шукати так само, як і перший, але в просторі, перпендикулярному вибраному напрямку, для якого ми проектуємо всі вхідні вектори на цей простір:

$$X^{(1)} = X - (w_1 X) \cdot w_1$$

У цьому просторі градієнт передбачуваності можна знову порахувати, визначивши індивідуальну значимість передбачуваних входів тощо. На кожному наступному етапі обчислюється індивідуальна значимість $I(Y, X_k^{(n)})$ для вхідної проекції

$$X^{(n)} = X - (w_1 X) \cdot w_1 - \cdots - (w_n X) \cdot w_n,$$

що не вимагає збільшення розміру аналізу підрахунку ящиків. Таким чином, вищезазначений метод дозволяє створити простір об’єктів будь-якого виміру без втрати точності.

Як і більшість математичних методів, алгоритми нейронної мережі працюють лише над числами. Робота нейронної мережі базується на арифметичних операціях множення та додавання, які не визначені для рядкових значень.

Не всі вхідні та вихідні змінні у вихідному просторі є числовими. Тому такі змінні слід закодувати та перетворити в числову форму перед початком роботи з нейронною мережею. Основною метою всіх фаз попередньої обробки даних є мінімізація ентропії.

Перетворюючи всі дані в числову форму, а потім нормуючи їх, усі поля введення та виведення відображаються в одиничному ящику. Задача моделювання нейронних мереж полягає у пошуку статистично надійних зв'язків між вхідними та вихідними змінними. Єдиним джерелом інформації для статистичного моделювання є приклади з навчальної вибірки. Чим більше інформації містить кожен приклад, тим краще будуть використовуватися доступні нам дані.

Розглянемо будь-яку складову нормованих (попередньо оброблених) даних: $\widetilde{x_i}$. Середній обсяг інформації, внесений кожним прикладом $\widetilde{x_i}^{\alpha}$, обернено пропорційний ентропії розподілу значення цього компонента $H(\widetilde{x_i})$. Якщо ці значення зосереджені на відносно невеликій площі одиничного інтервалу, інформаційний вміст такого компонента є невеликим. В межах ентропії, коли всі значення змінної збігаються, ця змінна не містить жодної інформації, тоді якщо значення змінної $\widetilde{x_i}^{\alpha}$ рівномірно розподіляються в одиничному інтервалі, інформація такої змінної максимальна [106].

Таким чином, загальним принципом попередньої обробки даних для навчання є мінімізація ентропії вхідних та вихідних даних. Цього правила слід дотримуватися при кодуванні нечислових змінних.

Існує два основних типи нечислових змінних: впорядковані та категоріальні. В обох випадках змінна належить до одного з дискретних класів

$\{c_1, c_2, \dots, c_n\}$, але в першому випадку ці класи впорядковані - їх можна розділити на: $c_1 > c_2 > \cdots > c_n$, тоді як у другому випадку такого порядку немає.

Для кодування впорядкованих змінних досить проставити відповідно номери категорій числові значення, які зберігали б поточний порядок. У той же час залишається свобода вибору - кожна монотонна функція породжує власний метод кодування. Як загальне правило вище, ми повинні прагнути мінімізувати ентропію закодованих даних. Для сигмоїдних функцій активації всі вихідні значення знаходяться в кінцевому діапазоні - зазвичай $[0,1]$ або $[-1,1]$. З усіх функцій статистичного розподілу, які визначені на скінченному інтервалі, рівномірний розподіл має мінімальну ентропію. Тобто кодування змінних з числовими значеннями повинне, якщо це можливо, призвести до рівномірного заповнення одиничного інтервалу закодованими прикладами. За допомогою цього методу кодування всі приклади несуть приблизно однакове інформаційне навантаження.

Категоріальні змінні також можна кодувати, як описано вище, випадковим їх нумеруванням. Однак це призводить до запровадження неіснуючого порядку, що ускладнює вирішення задачі. Для подолання цього недоліку використовується двійкове кодування типу $n \to n$, коли імена n категорій кодуються значеннями n двійкових нейронів, а перша категорія як $(1,0,0,\dots,0)$, друга, відповідно, - $(0,1,0,\dots,0)$ тощо, до n-ї: $(0,0,0,\dots 1)$. Детальнішу інформацію про можливі варіанти кодування категоріальних змінних можна знайти в [107-110].

Нормування даних

Нормуючи дані, можна ввести неоднорідні дані у порівнянні області та наблизити їх розподіл до однорідного.

На етапі збору даних про функціонування системи сукупні параметри можуть мати абсолютно різні розміри. Звичайно, результати моделювання

нейронних мереж не повинні залежати від одиниць виміру цих величин. Щоб мережа інтерпретувала свої значення рівномірно, всі вхідні та вихідні значення повинні бути приведені до єдиної шкали. Крім того, для збільшення швидкості і якості навчання потрібна додаткова попередня обробка даних, щоб вирівняти розподіл значень перед етапом навчання.

Приведення даних до єдиної шкали забезпечується шляхом нормування кожної змінної до діапазону її значень. У найпростішому вигляді це лінійне перетворення [111]:

$$\tilde{x}_i = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$$

в одиничний відрізок: $\tilde{x_i} \in [0, 1]$. Узагальнення відображення даних на інтервалі $[-1,1]$, рекомендованому для вхідних даних, тривіальне.

Лінійне нормування оптимальне, коли значення змінної x_i щільно заповнюють деякий інтервал. Але цей лінійний підхід можна застосувати не завжди. Таким чином, якщо в даних є відносно рідкісні викиди, які набагато перевищують типовий розподіл, то ці викиди будуть визначатися відповідно до попередньої формули, шкалою нормування. Це призведе до того, що велика частина значення нормованої змінної $\tilde{x_i}$ буде зосереджена близько нуля: $|\tilde{x_i}| \ll 1$.

Тому при нормуванні безпечніше керуватися не крайніми значеннями, а типовими, тобто статистичними характеристиками даних, такими як середнє значення і дисперсія:

$$\tilde{x}_i = \frac{x_i - \bar{x}_i}{\sigma_i}, \bar{x}_i \equiv \frac{1}{P}\sum_{\alpha=1}^{P} x_i^{\alpha}, \sigma_i^2 \equiv \frac{1}{P-1}\sum_{\alpha=1}^{P}(x_i^{\alpha} - \bar{x}_i)^2$$

У цьому випадку велика частина даних буде в єдиній шкалі, тобто типові значення всіх змінних будуть порівняні.

Однак тепер нормовані значення не потрапляють в гарантований діапазон значень, більше того, максимальний розподіл значень $\tilde{x_i}$ заздалегідь невідомий. Для вхідних даних це може не мати значення, але вихідні змінні будуть

використовуватися як посилання для вихідних нейронів. Якщо нейрони сигмоїдальні, вони можуть набувати значень тільки з одного діапазону, і в цьому випадку для встановлення сумісності між навчальною вибіркою і нейронною мережею необхідно обмежити діапазон зміни змінної.

Лінійне перетворення, як ми бачили, не може нормувати більшу частину даних і в той же час обмежувати діапазон можливих значень цих даних.

В результаті лінійне нормування по заданих формулах зводить всі значення до інтервалу $[-1;\ 1]$, нормування на основі статистичних характеристик ряду - до нуля математичних сподівань і одиничної дисперсії. Однак найкращий ефект дає комбінування цього методу нормування з нелінійним нормуванням з використанням функції активації по заданих формулах (таке нормування обов'язкове для вхідного компонента, що задає необхідне мережеве значення при навчанні - для приведення даних до області функції активації):

$$\tilde{x}_i = f\left(\frac{x_i - \bar{x}_i}{\sigma_i}\right), f(a) = \frac{1}{1 + e^{-ga}}$$

Нововведенням в цьому підході є змінна форма функції активації, яка дозволяє підвищити ефективність нормування за рахунок кращого наближення розподілу даних до рівномірного. Щоб можна було змінити тип функції активації, в її склад введений спеціальний коефіцієнт g.

Наведені гістограми (табл. 3.1) ілюструють розподіл нормованих даних після кожного кроку.

З табл. 3.1 видно, що в порівнянні з вихідними даними і лінійним нормуванням, нормування, засноване на статистичних характеристиках, робить розподіл набагато ближчим до однорідного, але діапазон даних вже $[-0{,}56;\ 0{,}71]$, і застосування лінійного нормування до цього розподілу приводить дані не тільки до інтервалу $[-1;\ 1]$, але також змінює щільність заповнення цього інтервалу.

Тобто послідовне застосування нормування на основі статистичних характеристик даних (рис. 3.2) і лінійного нормування (рис. 3.3), звичайно, ідентичне використанню тільки лінійного нормування. Таким чином, те, що було досягнуто в результаті нормування, заснованого на статистичних характеристиках нульових математичних сподівань і одиничної дисперсії, втрачається.

Однак, якщо послідовно застосовувати нормування на основі статистичних характеристик ряду і розроблене нелінійне нормування (рис. 3.4), то стає можливим застосувати лінійне нормування, яке в цьому випадку переносить дані в інтервал $[-1;\ 1]$, але вже не змінює щільність заповнення цього інтервалу (рис. 3.5).

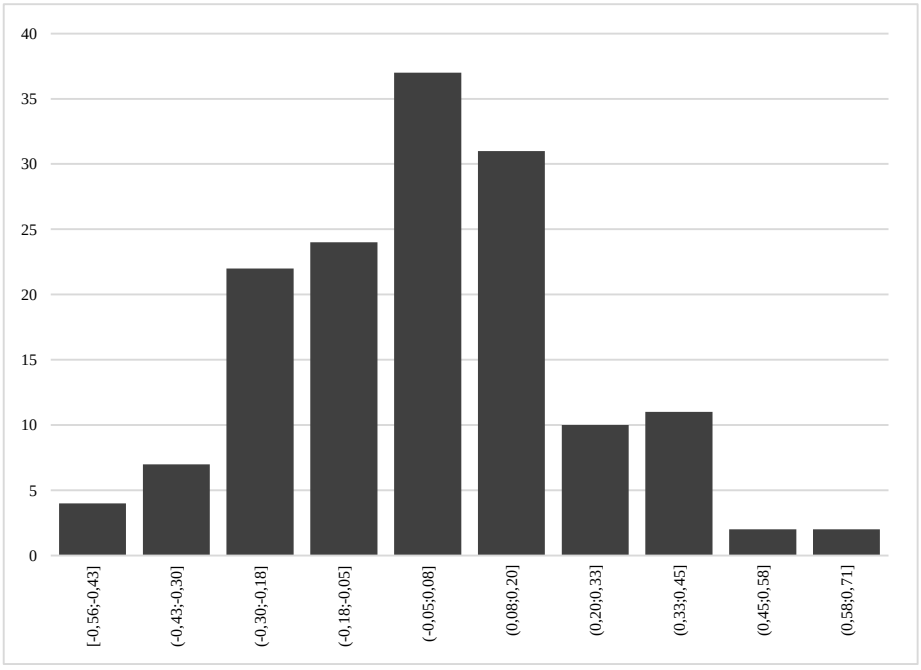
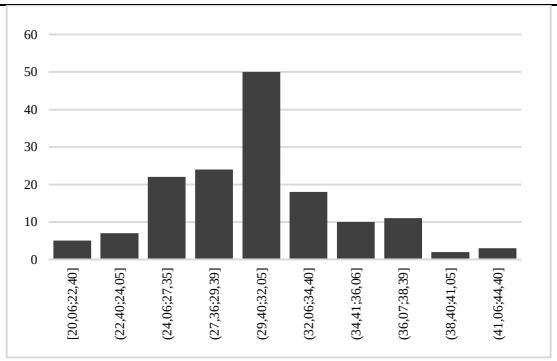
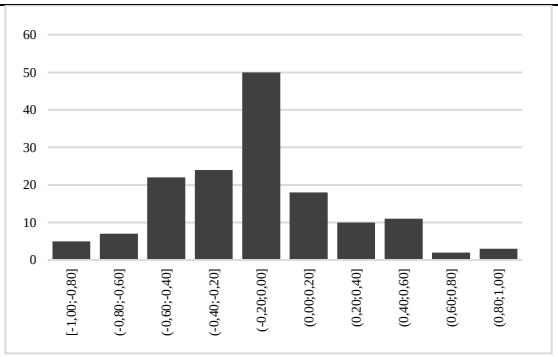
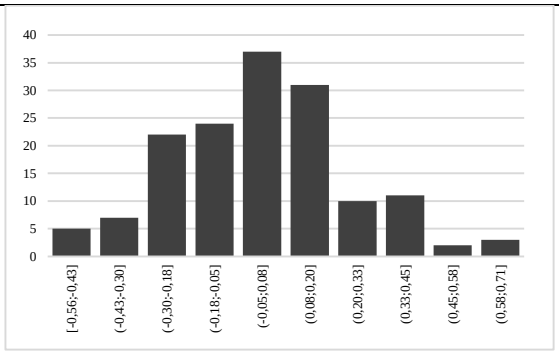
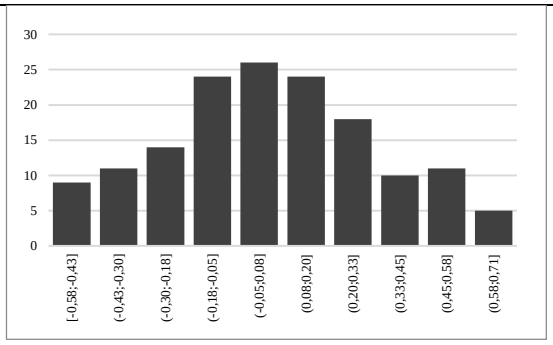
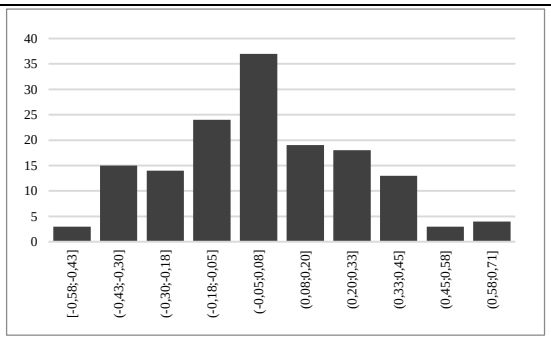


Рис. 3.2. Нормування на основі статистичних характеристик ряду

Таблиця 3.1

Методи нормування

<table>
<tr><td>Вихідні дані</td><td> 
</td></tr>
<tr><td>Лінійне нормування
$$S_i = \frac{x_i - midZnach}{\frac{znach}{2}},$$
$$znach = \max_i x_i - \min_i x_i,$$
$$midZnach = \frac{\max_i x_i + \min_i x_i}{2}$$</td><td> 
</td></tr>
<tr><td>На основі статистичних характеристик даних
$$S_i = \frac{x_i - \bar{x}_x}{\bar{\sigma}_x}, \bar{x}_x = \frac{1}{N}\sum_{i=1}^{N} x_i,$$
$$\bar{\sigma}_x = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x}_x)^2$$</td><td> 
</td></tr>
<tr><td colspan="2">Розроблений спосіб нормування даних з видом нелінійності, що змінюється
$$S_i = f\left(\frac{x_i - \bar{x}_x}{\bar{\sigma}_x}\right), f(a) = \frac{1}{1 + e^{-ga}}$$</td></tr>
<tr><td> 

де $g = 4$</td><td> 

де $g = 2$</td></tr>
</table>

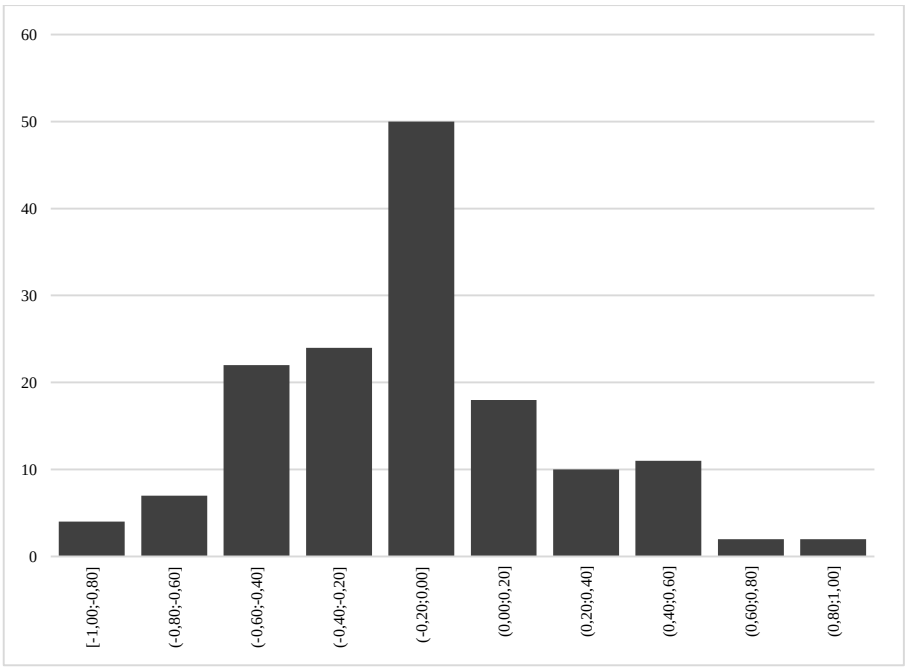


Рис. 3.3. Нормування на основі статистичних характеристик ряду і лінійне нормування

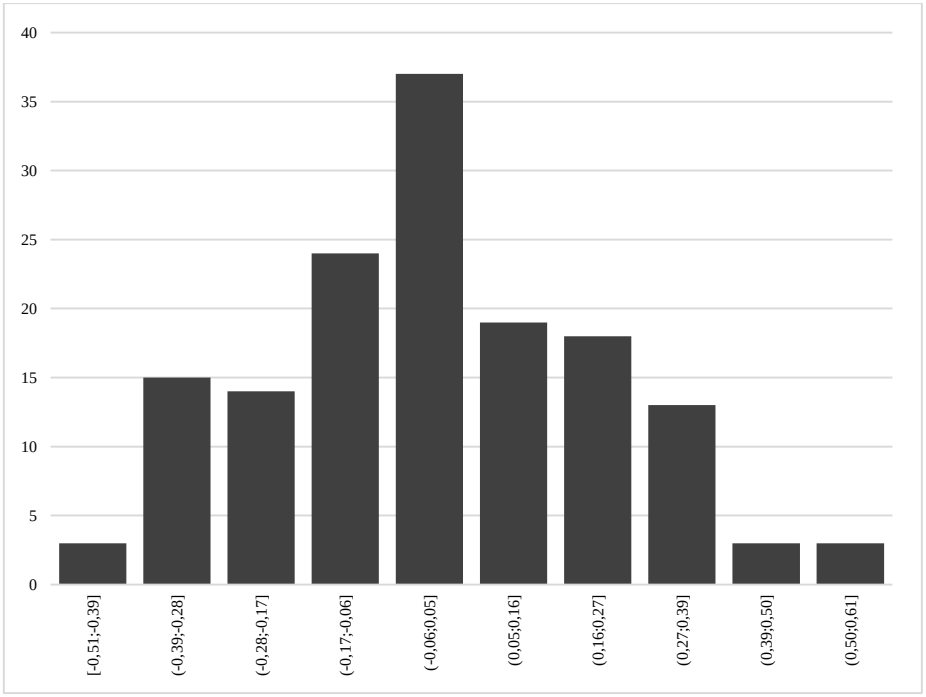


Рис. 3.4. Розроблений спосіб нелінійного нормування ($g = 2$)

Як показано на рис. 3.5, розподіл значень після цього нелінійного перетворення є набагато ближчим до рівномірного.

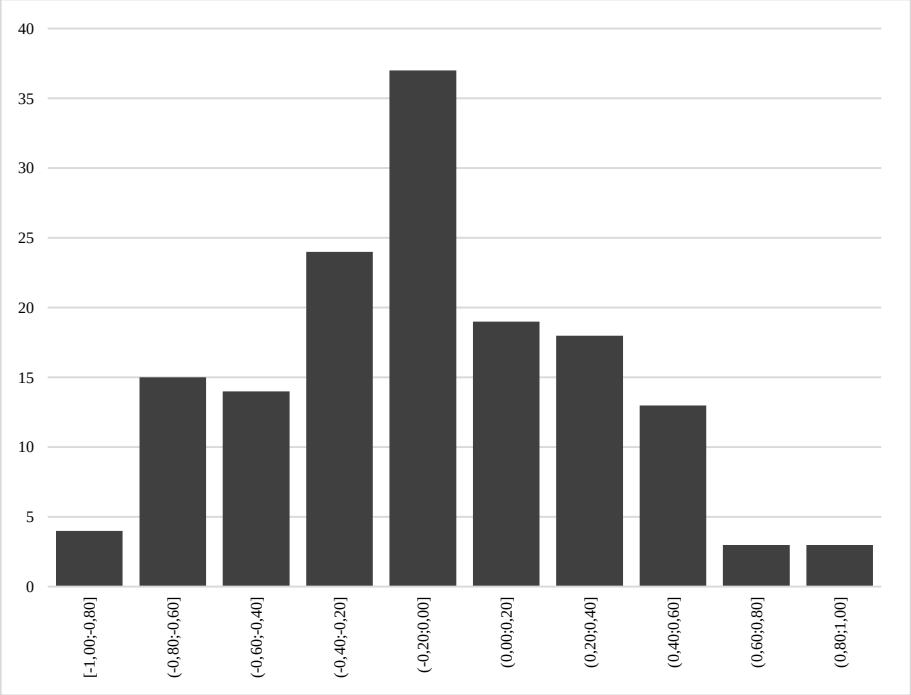


Рис. 3.5. Розроблений спосіб нелінійного нормування ($g = 2$) і лінійне нормування

До цього часу методи мінімізації ентропії кожного входу (виходу) розглядались окремо. Існують технології, що дозволяють загально мінімізувати ентропію вхідних даних [112]. Загальне нормування: «вибілювання» входів.

Якщо два входи не є статистично незалежними, їх загальна ентропія менша за суму окремих ентропій: $H(\tilde{x}_i\tilde{x}_j) \leq H(\tilde{x}_i) + H(\tilde{x}_j)$. Як тільки досягається статистична незалежність вхідних даних, отже можна збільшити інформаційну насиченість вхідної інформації. Однак для цього потрібен більш складний метод загального нормування вхідних даних.

Замість використання окремих дисперсій для нормування розглянемо вхідні дані загалом. Необхідно знайти таке лінійне перетворення, яке дозволить

мінімізувати його загальну ентропію. Для спрощення задачі, замість більш складної умови статистичної незалежності, вводиться вимога декорелювати нові вхідні дані (для будь-якого x_i виконується $\overline{(x_\iota - \bar{x}_\iota)(x_\jmath - \bar{x}_\jmath)} = 0, \forall i \neq j$) після цього перетворення. Для цього ми обчислюємо середній вектор та матрицю коваріації даних за такими формулами:

$$\bar{x} \equiv \frac{1}{P}\sum_{\alpha=1}^{P} x^{\alpha}, \sum\nolimits_{ij}^{X} \equiv \frac{1}{P-1}\sum_{\alpha=1}^{P}(x_i^{\alpha} - \bar{x}_i)(x_j^{\alpha} - \bar{x}_j)$$

Тоді ми знаходимо лінійне перетворення, яке діагоналізує матрицю коваріації. Відповідна матриця складається з власних векторів стовпців матриці коваріації:

$$\sum_j \sum\nolimits_{ij}^{X} U_{jk} = \lambda_k U_{ik}$$

Лінійне перетворення, яке називається «відбілюванням» $\tilde{x}_i = (x_k - \bar{x}_k)U_{ki}/\sqrt{\lambda_i}$, що перетворює всі вхідні дані у некорельовані величини із середнім значенням, рівним нулю та одиничною дисперсією [113]. Це перетворення зменшує загальну ентропію вхідних даних, оскільки воно вирівнює розподіл даних у навчальній вибірці.

3.3. Створення структури нейронної мережі

Визначення параметрів нейронної мережі

Визначення параметрів нейронної мережі є важливим та трудомістким етапом аналізу даних. Через вимогу до сучасних аналітичних систем - простоту використання та відсутність у дослідника спеціальних знань щодо математичного ядра системи - необхідно запропонувати універсальну структуру нейронної мережі, яка може вирішити більшість задач і автоматично пристосовується до змін у системі.

Вирішення цієї задачі полягає у виборі топології нейронної мережі, яка є найбільш повною з усіх архітектур нейронних мереж, розроблених на сьогодні з точки зору класів задач, що підлягають вирішенню.

Вибір топології

На основі вищезазначеної класифікації топології нейронних мереж (рис. 2.3), розглянемо архітектури нейронних мереж і виберемо серед тих, які найкраще відповідають вимогам. Ця фаза тісно пов'язана з іншим етапом аналізу: спрощенням нейронної мережі, обговореним нижче. Будемо зрозуміти під логічно прозорою архітектурою, топологію, що характеризується, серед інших, більшою вербалізацією.

Постановка задачі вибору оптимальної структури для нейронної мережі передбачає поетапну роботу з нейронною мережею. Одна з можливих схем роботи з нейронною мережею наведена на рис. 3.6. Пунктирна лінія позначає стадію, яка часто відсутня в програмах: нейроімітатор.

На відміну від звичних схем функціонування нейронної мережі, етапи відбору архітектури та відбору структурних елементів нейронної мережі тут навмисно розділені (структурними елементами є шари, нейрони та глобальні внутрішні зв’язки). Виділяючи етапи, можна стверджувати, що архітектура нейронної мережі обирається дослідником на першому етапі і не може зазнавати змін у майбутньому, навіть при використанні методу спрощення. Дійсно, видалення синапсів може призвести до видалення нейрона, але ні перший, ні другий не ведуть до циклів у функціонуванні мережі, якщо не ввести їх у фазу архітектурного синтезу.

Тому може бути поставлене питання про вибір оптимальної архітектури нейронної мережі для вирішення конкретної задачі.

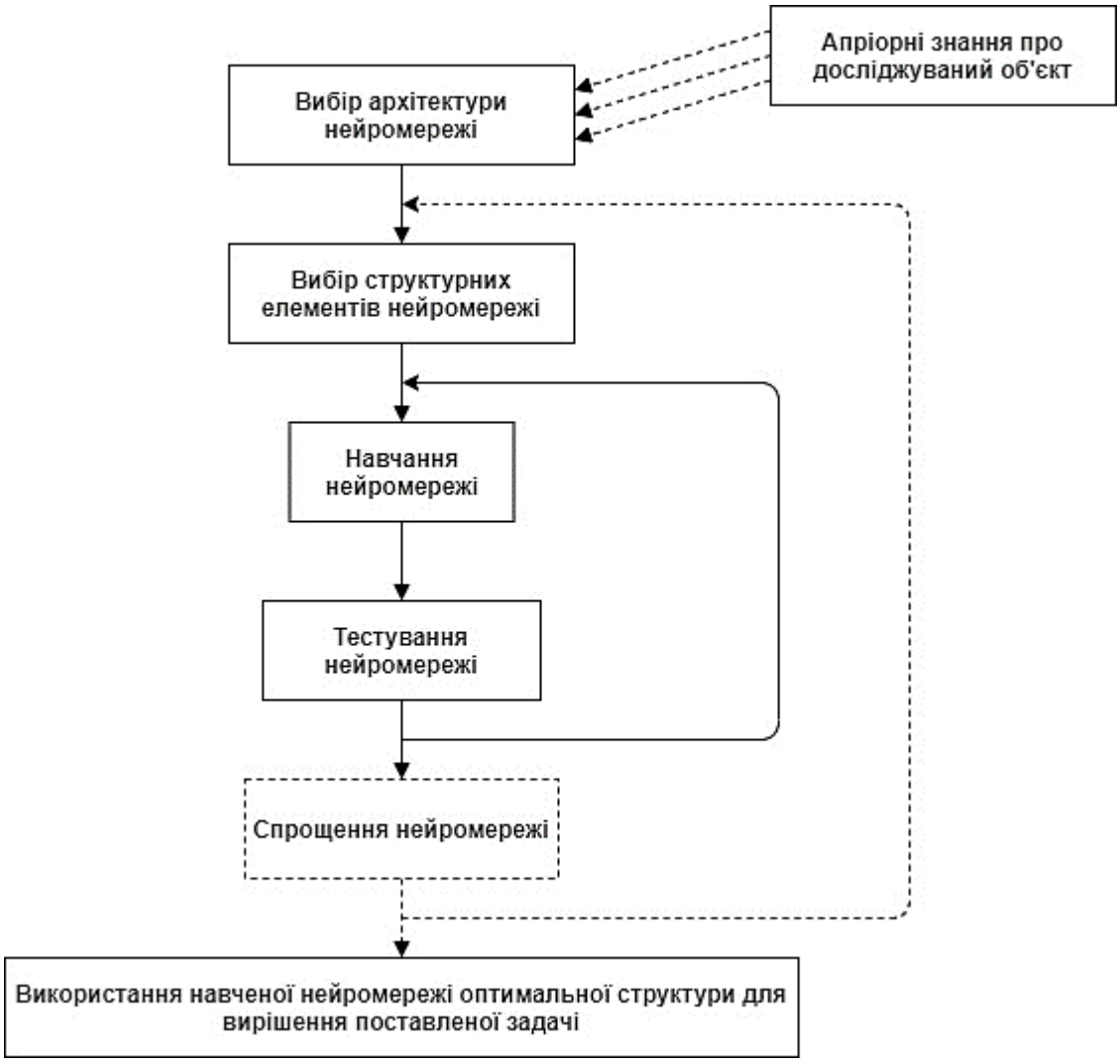


Рис. 3.6. Етапи роботи з нейронною мережею

Як показано на рис. 3.6, архітектура вибирається при створенні нейронної мережі, і, як видно, робота з нею в переважній більшості випадків не призводить до змін в архітектурі. Таким чином, можна створити клас спочатку універсальних і логічно прозорих архітектур нейронних мереж і працювати з ними в майбутньому, починаючи зі створення мережі і закінчуючи введенням набору правил роботи досліджуваної системи.

Звернемо увагу на основи функціонування нейронної мережі [18, 19, 114], про які слід пам’ятати при синтезі її архітектури:

- можливості мережі зростають із збільшенням кількості нейронів, щільності зв'язків між ними та кількості призначених шарів;
- складність алгоритмів функціонування мережі (наприклад, впровадження декількох типів синапсів або запровадження циклів зворотного зв'язку) збільшує можливості нейронної мережі.

Слід зазначити, що будь-який метод вдосконалення здатності нейронної мережі вирішувати задачу неминуче призводить до ускладнень у розумінні способу вирішення задачі. Тому на вибір архітектури нейронної мережі суттєво впливає тип задачі, для якої розроблена мережа. Для вирішення деяких видів задач в даний час доступні оптимальні конфігурації, які описані в [115-118].

Для вирішення задачі можна спроектувати кілька нейронних мереж з різною топологією, які вирішать задачу із задовільною точністю. У цьому випадку доцільніше вибрати мережу з найпростішою архітектурою, тобто з мінімально можливої кількості нейронів без зворотного зв’язку з обмеженою кількістю глобальних зв’язків.

Чим складнішою є задача для нейронної мережі, тим більше нейронів вона повинна містити, а отже, може бути розроблена більш складна структура впорядкованих нейронів у шарах, з'єднаних глобальними зв'язками.

Теоретично кількість шарів і кількість нейронів у кожному шарі нейронної мережі можуть бути довільними, в даний час вона практично необмежена використанням комп'ютерних ресурсів або мікросхем, на яких вона зазвичай реалізована. Чим складніша мережа, тим складніші задачі можна вирішити.

Критерії логічної ясності та універсальності архітектур нейронних мереж не обов'язково відповідають критеріям найпростішої технічної або програмної реалізації мережі. Тому представляється можливим використовувати їх як універсальні та логічно прозорі мережі з багаторівневою архітектурою. Багатошарові мережі, швидше за все, керуються ідеєю природного способу вирішення задачі. У багатошарових мережах існує певний розподіл набору

вхідних характеристик між нейронами від першого до останнього шару. Наявність циклів у архітектурі мережі ускладнює процес відстеження руху елементів вздовж обчислювального графа [119].

Нейронні мережі будь-якої архітектури можна представити як багатошарові мережі. Єдине обмеження полягає в тому, що мережа не повинна мати нескінченних циклів. Іншими словами, у мережах із циклами (повністю повнозв'язними та спорідненими) процес роботи мережі повинен завершуватися через певну задану кількість циклів. Існують мережі з повністю повнозв'язними архітектурами, для яких номер циклу не відомий заздалегідь. Робота цих мереж переривається, коли досягається такий факт, як певний стан рівноваги на виході, якщо новий цикл суттєво не змінює значення мережевих виходів. Крім того, кількість робочих циклів різна для кожної копії, що надходить у мережу. Такі мережі навряд чи можна звести до логічно прозорої форми [119].

Тому ми вибираємо багатошарову архітектуру з прямим поширенням сигналу як логічно прозору архітектуру нейронної мережі. Однак цей клас топології, як показано на рис. 2.3, включає кілька типів нейронних мереж (наприклад, багатошаровий персептрон, мережі радіально базисних функцій). На основі класифікації топології нейронних мереж, якщо це можливо, з точки зору їх застосування при вирішенні різних типів задач (табл. 2.2), можна свідомо вибрати найбільш універсальну архітектуру нейронної мережі - багатошаровий персептрон.

Визначення необхідної кількості нейронів

Існує два можливих підходи до вирішення задачі вибору оптимальної кількості нейронів. Перший заснований на твердженні, що чим більше нейронів ви використовуєте, тим надійнішою буде ваша мережа. На користь цього підходу можна навести приклад людського мозку. Чим більше нейронів, тим більше зв'язків між ними і тим складніші задачі може вирішити нейронна мережа. Більше того, відомо, що використання свідомо більшої кількості нейронів для

вирішення задачі, ніж потрібно, повинно призвести до навчання мережі [120]. Інструкція перетворення не відповідає дійсності, тобто якщо ви почнете тренувати мережу з невеликою кількістю нейронів, мережа може не навчитися вирішувати задачу через свою просту структуру, у цьому випадку процес навчання доведеться повторити, але з більшою кількістю нейронів. Ця перспектива (чим більше, тим краще) користується популярністю серед розробників програмного забезпечення для нейронних мереж. Багато з них припускають можливість використання необмеженої кількості нейронів як перевагу їх розвитку [120].

Друга точка зору заснована на принципі: чим більше регульовані параметри, тим гірше наближення функції в тих областях, де її значення раніше були невідомі. З математичної точки зору, навчальні задачі нейронних мереж зводяться до продовження функції, заданої в кінцевій кількості точок у всій області визначення. При цьому підході вхідні дані приймаються як аргумент функції, а відповідь мережі - значення функції. У роботі [121] наведено приклад апроксимації масивної функції поліномами 3-го та 8-го ступенів і показано, що наближення, отримане поліномом 3-го ступеня, більш відповідає ідеї правильного наближення. Незважаючи на простоту наведеного прикладу, суть задачі наочно показана [121].

Другий підхід визначає необхідну кількість нейронів якомога меншою, але достатньою, щоб підтримувати здатність мережі вирішувати задачу. Недоліком цього підходу є той факт, що мінімально необхідна кількість нейронів не відома заздалегідь, і її визначення шляхом поступового збільшення кількості нейронів вимагає значного часу і безпосередньо залежить від масштабу задачі та кількості прикладів у навчальній вибірці. Кількох десятків нейронів виявляється достатньо для вирішення багатьох задач.

На основі аналізу розглянутих положень ми можемо зробити висновок, що мережа з мінімальною кількістю нейронів повинна краще наближати функцію,

але визначення цієї достатньої кількості нейронів вимагає великих обчислювальних втрат під час експериментів на навчальних мережах. Якщо кількість нейронів надмірна, можна отримати мережу, яка зможе навчитися з першого разу, але існує небезпека побудови неіснуючого наближення, оскільки при достатній кількості нейронів мережа також буде вчитися на випадково сформованих даних. Тому нам слід уникати екстремальних підходів до вирішення задачі відбору нейронів і намагатися відібрати більше ніж необхідну кількість нейронів, але не багато. Цього можна досягти, наприклад, збільшивши кількість нейронів у мережі кілька разів після кожної невдалої спроби навчання [121]. Однак існує більш надійний спосіб оцінити мінімальну кількість необхідних нейронів - за допомогою процедури контрастування [120, 121]. Детальний опис існуючих можливостей видалення надлишкових нейронів буде описано при розгляді четвертого етапу аналізу - спрощення нейронної мережі.

3.4. Навчання нейронної мережі

Наступним кроком в аналізі даних за допомогою нейронних мереж є навчання нейронних мереж. Вміння вчитися - це основна особливість нейромережевої технології. У контексті штучної нейронної мережі процес навчання можна розглядати як встановлення ваги зв'язку для ефективного виконання спеціальної задачі. Нейронній мережі зазвичай доводиться регулювати ваги зв'язків для наявного навчального набору. Продуктивність мережі покращується в міру циклічного регулювання вагових коефіцієнтів. Здатність мереж навчатися на прикладах робить їх більш привабливими порівняно з підходами, які дотримуються певної системи робочих правил, сформульованих експертами.

Для проектування навчального процесу спочатку необхідно мати модель зовнішнього середовища, в якому працює нейронна мережа - знати інформацію,

доступну мережі. Ця модель визначає парадигму навчання [88]. Крім того, необхідно зрозуміти, як відрегулювати вагу мережі - які правила навчання регулюють процес налагодження. Алгоритм навчання - це процедура, яка використовує правила навчання для встановлення ваг.

Алгоритм навчання нейронних мереж

Існує три парадигми навчання: «з учителем» (контрольоване), «без вчителя» (самонавчання) та комбіноване навчання [50, 88, 100]. У першому випадку нейронна мережа має правильні відповіді для кожного прикладу введення. Ваги налаштовані таким чином, щоб мережа давала відповіді якомога ближчі до відомих правильних відповідей. Вдосконалена версія контрольованого навчання передбачає, що відома лише критична оцінка правильного виходу нейронної мережі, але не саме значення вихідного сигналу. Смонавчання не вимагає знання правильних відповідей для кожного навчального прикладу. У цьому випадку виявляється внутрішня структура даних або кореляція між вибірками в системі даних, що дозволяє класифікувати зразки. У змішаному навчанні деякі ваги визначаються контрольованим навчанням, решта отримується самостійним навчанням.

Теорія навчання враховує три основні властивості, пов’язані з навчанням за зразками [50, 52, 88, 100, 105]: потужність, складність вибірки та обчислювальна складність. Потужність означає, скільки зразків мережа може запам'ятати і які функції та межі прийняття рішень можуть бути створені на ній. Складність зразків визначає кількість навчальних прикладів, необхідних для досягнення узагальненості мережі. Занадто мало прикладів, разом із надмірною кількістю нейронів, можуть спричинити перепідготовку мережі, тобто мережа буде гарною у класифікації, прогнозуванні тощо. У навчальних прикладах, але погано в тестових прикладах, той самий розподіл фактично є предметом. У роботі [122] виділено чотири основних типи правил навчання: виправлення по помилці, машина Больцмана, правило Хебба та конкурентне навчання. Коротко опишемо

кожне правило навчання, щоб знайти серед них відповідну універсальну логічно прозору топологію нейронної мережі, обрану для використання в навчанні - багатошаровий персептрон.

Правило корекції за похибкою. При контрольованому навчанні необхідний вихід d заданий для кожного прикладу введення. Фактичний вихід мережі y може не відповідати необхідному. Принцип виправлення помилок під час навчання полягає у використанні сигналу $(d - y)$ для регулювання ваг так, щоб похибка поступово зменшувалася. Навчання відбувається лише тоді, коли персептрон помиляється. Відомі різні модифікації цього алгоритму навчання.

Навчання Больцмана. Це стохастичне правило навчання, яке випливає з інформаційних теоретичних та термодинамічних принципів [47]. Метою навчання Больцмана є коригування ваг таким чином, щоб стани видимих нейронів задовольняли бажаний розподіл ймовірностей. Навчання Больцмана можна розглядати як окреме виправлення помилок, в якому похибка розуміється як невідповідність між кореляціями станів у двох режимах.

Правило Хебба. Найдавнішим правилом навчання є навчальний постулат Хебба. Хебб покладався на такі нейрофізіологічні спостереження: коли нейрони по обидва боки синапсу активізуються одночасно і регулярно, сила синаптичного зв'язку зростає. Важливою особливістю цього правила є те, що зміна синаптичної ваги залежить лише від активності нейронів, пов'язаних із даним синапсом. Це значно спрощує ланцюжок навчання.

Конкурентне навчання. На відміну від навчання Гебба, при якому кілька вихідних нейронів можуть вистрілювати одночасно, у конкурентному навчанні вихідні нейрони конкурують між собою за активізацію. Подібне навчання відбувається в біологічних нейронних мережах. Конкурентне навчання дозволяє групувати вхідні дані: подібні приклади згруповані за мережею відповідно до кореляцій та представлені одним елементом.

Таким чином, для навчання багатошарового персептрона існує теоретична можливість використання наступних правил навчання: правило гавчання Хебба та правило корекції за похибкою.

У багатошарових мережах оптимальні вихідні значення нейронів усіх шарів, за винятком останнього, зазвичай невідомі, і два або більше шарів персептрона більше не можна навчити, керуючись лише значеннями помилок на виходах нейронної мережі. Одним із способів вирішення цієї задачі є розробка наборів вихідних сигналів, що відповідають вхідним сигналам для кожного шару нейронної мережі, що, звичайно, є дуже трудомісткою операцією і не завжди можливо. Другий варіант - це динамічне регулювання ваги синапсу, під час якого зазвичай вибираються найменші з'єднання і змінюються в той чи інший бік, і зберігаються лише ті зміни, які зменшили похибку на виході всієї мережі. Цей метод експериментального налаштування, незважаючи на очевидну простоту, вимагає громіздких розрахунків. Нарешті, третім, більш прийнятним варіантом є розповсюдження сигналів помилок з виходів нейронної мережі на її входи у напрямку, протилежному прямому розповсюдженню сигналів у звичайній роботі [49, 84, 100, 120]. Цей алгоритм навчання мережі називається процедурою зворотного розповсюдження. Ми виберемо його як алгоритм навчання персептрону.

Відповідно до методу найменших квадратів значення, яке мінімізується функцією цільової похибки нейронної мережі, є таким:

$$E(w) = \frac{1}{2}\sum_{j,P}\left(y_{j,P}^{(N)} - d_{j,P}\right)^2 \qquad (3.1)$$

де $y_{j,P}^{(N)}$ - фактичний вихідний стан нейрона j на вихідному шарі N нейронної мережі при застосуванні P-го образу до його входів; d_j - ідеальний (бажаний) вихідний стан цього нейрона.

Сума виконується по всіх нейронах вихідного шару та над усіма даними, обробленими мережею. Мінімізація виконується методом градієнтного спуску, що означає встановлення вагових коефіцієнтів наступним чином:

$$\Delta w_{ij}^{(n)} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{3.2}$$

Тут w_{ij} – ваговий коефіцієнт синаптичного з'єднання, що з'єднує i-й нейрон $(n-1)$-го шару з j-м нейроном n-го шару, η - коефіцієнт швидкості навчання, $0 < \eta < 1$.

Запишемо другий множник із формули 3.2:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{ds_j} \cdot \frac{ds_j}{\partial w_{ij}} \tag{3.3}$$

Тут, під y_j, як і раніше, вихід нейрона j і s_j є зваженою сумою його вхідних сигналів, тобто аргументом функції активації. Оскільки множник dy_j/ds_j є похідною від цієї функції по її аргументу, звідси випливає, що похідна функції активації повинна бути визначена на всій осі абсцис. У цьому відношенні функції єдиного стрибка та інші функції активації з неоднорідностями не підходять для розглянутих нейронних мереж. Вони використовують гладкі функції, такі як гіперболічний тангенс або класичний сигмоїд з експонентою. У випадку гіперболічного тангенса:

$$\frac{dy}{ds} = 1 - s^2 \tag{3.4}$$

Третій множник $\partial s_j/\partial w_{ij}$, очевидно, такий самий, як вихід попереднього шару нейрона $y_i^{(n-1)}$.

Що стосується першого множника у формулі 3.3, його можна легко розкласти наступним чином:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \tag{3.5}$$

Тут виконується сума по k між нейронами шару $(n+1)$. Вводячи нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \tag{3.6}$$

отримуємо рекурсивну формулу для обчислення величин $\delta_j^{(n)}$ шару n із величин $\delta_k^{(n+1)}$ старшого шару $(n+1)$.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)}\right] \cdot \frac{dy_j}{ds_j} \tag{3.7}$$

Для вихідного шару

$$\delta_i^{(N)} = (y_i^{(N)} - d_i) \cdot \frac{dy_i}{ds_i} \tag{3.8}$$

Тепер можемо записати формулу 3.2 у розгорнутому вигляді:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \tag{3.9}$$

Іноді для того, щоб процес корекції ваг забезпечив деяку інерцію, яка згладжує різкі стрибки при русі поверхнею цільової функції, формула 3.9 доповнюється значенням зміни маси в попередній ітерації [84]:

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot \left(\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}\right) \tag{3.10}$$

де μ - коефіцієнт інерції, t - номер поточної ітерації.

Таким чином, повний алгоритм навчання нейронних мереж із використанням процедури зворотного розповсюдження складається таким чином [49]:

1. Застосування одного з можливих образів до мережевих входів і під час нормальної роботи нейронної мережі, коли сигнали поширюються від входів до виходів, обчислення їх значення. Нагадаємо, що

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} \cdot w_{ij}^{(n)} \tag{3.11}$$

де M - кількість нейронів у шарі $(n-1)$ з урахуванням нейрона з постійним вихідним станом $+1$, що задає зміщення; $y_i^{(n-1)} = x_{ij}^{(n)}$ - i-й вхід нейрона j шару n.

$y_j^{(n)} = f(s_j^{(n)})$, де $f()$ – сигмоїд,

$$y_q^{(0)} = I_q,$$

де I_q - q-й компонент вектора вхідного образу.

2. Обчислення $\delta^{(N)}$ для вихідного рівня, використовуючи формулу 3.8. Обчислення зміни ваги $\Delta w^{(N)}$ шару N, використовуючи формулу 3.9 або 3.10.

3. Обчислення за формулами 3.7 та 3.9 (або 3.7 і 3.10) відповідно $\delta^{(n)}$ та $\Delta w^{(n)}$ для всіх інших шарів, $n = N-1, \ldots 1$.

4. Налаштування всіх ваг в нейронній мережі

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (3.12)$$

5. Якщо похибка мережі значна, перехід до кроку 1. В іншому випадку - вихід.

Складність обчислень цього алгоритму становить $\sim PW^2$ операції.

Недоліком цього методу навчання знаходження локального мінімуму і заміна формули 3.9 на 3.10, тобто введення моменту вирішує цю задачу лише для невеликих локальних мінімумів. Альтернативним методом навчання нейронних мереж є генетичний метод, який забезпечує знаходження глобального мінімуму функцій; однак, генетичний метод навчання, очевидно, потребуватиме більше часу для навчання мережі через можливість розвитку популяції, що переходить у стадію виродження. В [100] наводиться приклад, коли для підготовки нейронної мережі з одним прихованим шаром, що складається з 10 нейронів, п’яти входів та трьох виходів за допомогою генетичного алгоритму, потрібно приблизно 10000 ітерацій генетичного алгоритму, навчання однієї і тієї ж мережі за допомогою градієнтного методу вимагає близько 1000 ітерацій на комп’ютері з однаковою потужністю. Хоча математичну складність однієї ітерації генетичного алгоритму не порівнювали з однією ітерацією методу зворотного розповсюдження, наведений випадок передбачає збільшення часу навчання нейронної мережі генетичним методом майже на порядок порівняно з алгоритмом зворотного поширення. При застосуванні генетичного алгоритму для

мережевого навчання виникає багато питань щодо його налаштування та оптимізації на автоматичному рівні (відповідно вимог до систем аналізу даних - втручання користувача в математичний апарат має бути мінімальним (повністю відсутнім)), але загалом можливість його використання дуже цікава.

У класичному алгоритмі градієнтного навчання мережі у якості функція похибки використовується найменший квадрат. Формула 3.3 показує подання похідної складної функції як добуток часткових похідних. Тоді частинна похідна по виходу $\frac{\partial E}{\partial y_j}$ буде дорівнювати:

$$\frac{\partial E}{\partial y_j} = \left(\frac{1}{2}\left(y_j - d_j\right)^2\right)' = \left(y_j - d_j\right) \tag{3.13}$$

Можна запропонувати використовувати як функцію похибки таку спеціально розроблену функцію:

$$\gamma = \left(\frac{1}{1+\exp\left(c\left(|y_j - t_j| + d\right)\right)} - h\right) \cdot k, \tag{3.14}$$

тут y_j - фактичний вихід нейронної мережі, t_j - бажаний вихід нейронної мережі, h - зсув сигмоїдальної ділянки щодо осі абсцис, c - нахил сигмоїдальної лінії, d - прирівнює значенню функції до нуля без різниці між фактичним та бажаним виходом мережі, k – прирівнює значення функції до 1 при між фактичною та бажаною вихідною потужністю мережі, що становить 1.

Знайдемо d:

$$0 = \left(\frac{1}{1+\exp\left(c(d)\right)} - h\right) \cdot k;$$

$$0 = \left(\frac{1}{1+\exp(cd)} - h\right);$$

$$\frac{1}{1+\exp(cd)} = h;$$

$$1 + \exp(cd) = \frac{1}{h};$$

$$\exp(cd) = \frac{1}{h} - 1;$$

$$cd = \ln\left(\frac{1}{h} - 1\right);$$

$$d = \frac{1}{c}\ln\left(\frac{1}{h} - 1\right). \quad (3.15)$$

Знайдемо k:

$$1 = \left(\frac{1}{1+\exp\left(c(|y_j - t_j| + d)\right)} - h\right) \cdot k;$$

$$k = \left(\frac{1}{1+\exp\left(c(1+d)\right)} - h\right)^{-1}. \quad (3.16)$$

Часткова похідна бере участь у формулі 3.3, яка для функції активації сигмоїдальних нейронів буде дорівнює:

$$\frac{dy}{ds} = F(s)' = \left(\frac{1}{1-\exp(-\alpha \cdot s)}\right)' = \alpha \cdot F(s) \cdot (1 - F(s)) \quad (3.17)$$

Загальна схема функціонування нейронів наведена на рис. 3.7.

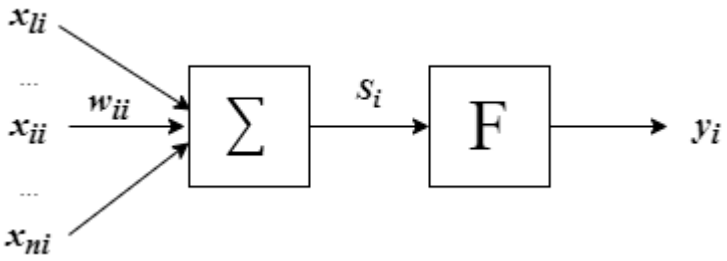


Рис. 3.7. Схема функціонування нейрона

Запишемо частинну похідну похибки введення:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial x_{ij}} \quad (3.18)$$

Отже $\frac{\partial E}{\partial y_j}$ і $\frac{dy_j}{ds_j}$ уже знайдено раніше, запишемо чому дорівнює $\frac{\partial s_j}{\partial x_{ij}}$:

$$\frac{\partial s_j}{\partial x_{ij}} = \frac{\partial\left(w_{ij} \cdot x_{ij}\right)}{\partial x_{ij}} = w_{ij}$$

Тоді формула 3.18 буде записана як:

$$\frac{\partial E}{\partial x_{ij}} = \left(y_j - d_j\right) \cdot \alpha \cdot F(s) \cdot (1 - F(s)) \cdot w_{ij}$$

для середньоквадратичної похибки (3.19)

$$\frac{\partial E}{\partial x_{ij}} = \gamma' \cdot \alpha \cdot F(s) \cdot (1 - F(s)) \cdot w_{ij}$$

для розробленої функції похибки (3.20)

Введемо аналогічні позначення до формули 3.6, отримуємо:

$$\delta_j^{(n)}{}_{(\text{мнк})} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial s_j} = \left(y_j - d_j\right) \cdot \alpha \cdot F(s) \cdot (1 - F(s)) \tag{3.21}$$

для середньоквадратичної похибки (де мнк - метод найменших квадратів).

$$\delta_j^{(n)}{}_{(\text{мс})} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial s_j} = \gamma_j' \cdot \alpha \cdot F(s) \cdot (1 - F(s)) \tag{3.22}$$

для розробленої функції похибки (мс - сигмоїдний метод).

Як буде показано пізніше у процесі програмної реалізації, розроблена функція похибки забезпечує більшу зміну ваги у разі великих відхилень фактичної вихідної потужності мережі від бажаної в порівнянні з методом найменших квадратів, що призводить до збільшення швидкості навчання мережі. У той же час, в області невеликих відхилень фактичної вихідної потужності мережі від встановленого значення, значення корекції синаптичної ваги порівняне з цим значенням (або навіть нижче) за допомогою методу найменших квадратів, що дозволяє більш точне налаштування. Більше того, зміщення точки перегину графіка до неправильного рішення дає більший запас міцності для навчання мережі.

На основі розробленої функції похибки можна запропонувати функцію загальної похибки, яка є похибкою не одного нейрона, а всієї мережі. Сумарною похибкою в загальному випадку є сума похибок нейронів вихідного шару, поділена на константу c:

$$E_i = \frac{1}{c} \cdot \sum_j \gamma_j \tag{3.23}$$

В якості критерію зупинки використовується порогове значення розробленої функції сумарної похибки: $\sum_i E_i < E_{max}$, де E_i - повна похибка i-го прикладу, E_{max} - порогове значення.

У зв’язку з тим, що вираз функції похибки, загальна похибка мережі та формули роботи алгоритму навчання взаємно залежать, використання розробленої функції похибок також передбачає використання спеціальної функції загальної похибки, а також розробка нового алгоритму навчання для нейронної мережі. Розроблений алгоритм базується на основних принципах алгоритму поширення зворотних похибок. Як теоретичне обгрунтування правила дельта, застосованого в алгоритмі (правила корекції ваг), необхідно довести наступні теореми.

Теорема 1.

Похибка i-го нейрона передостаннього шару $(n-1)$ може бути виражена похибкою j-го нейрона останнього (вихідного) шару n за формулою:

$$\gamma_i = \sum_j (y_j - t_j) \cdot c \cdot \gamma_j \cdot (1-\gamma_j) \cdot F(s_j) \cdot \left(1 - F(s_j)\right) \cdot w_{ij}, \qquad (3.24)$$

де w_{ij} - ваговий коефіцієнт синаптичного з'єднання, що з'єднує i-й нейрон шару $(n-1)$ з j -м нейроном шару n, t_j - ідеальний вихідний стан цього нейрона.

Доведення:

Через те, що похідна похибки щодо вхідного сигналу x_{ij} для останнього шару j збігається за значенням з похідною щодо відповідного виходу y_j для попереднього шару i, можемо записати:

$$\sum_j \frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_i},$$

$$\sum_j \frac{\partial E}{\partial y_i} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial x_{ij}} = \frac{\partial E}{\partial y_i},$$

$$\sum_j \gamma_j' \cdot F'(s_j) \cdot w_{ij} = \gamma_i,$$

$$\sum_j (y_j - t_j) \cdot c \cdot \gamma_j \cdot (1-\gamma_j) \cdot F(s_j) \cdot \left(1 - F(s_j)\right) \cdot w_{ij} = \gamma_i. \qquad (3.25)$$

Теорему доведено.

Теорема 2.

Похідна похибки, обумовлена ваговими коефіцієнтами j-х нейронів для вихідного шару n та i нейронів попереднього шару $(n-1)$, визначається наступним чином:

$$\frac{\partial E}{\partial w_{ij}} = \left(y_j - t_j\right) \cdot c \cdot \gamma_j \cdot \left(1 - \gamma_j\right) \cdot F\left(s_j\right) \cdot \left(1 - F\left(s_j\right)\right) \cdot y_i \qquad (3.26)$$

Доведення:

Через те, що вхідний сигнал x_{ij} для останнього шару j збігається за значенням з відповідним вихідним сигналом y_j для попереднього шару i, то:

$$\frac{\partial s_j}{\partial w_{ij}} = x_{ij} = y_i.$$

Доведення:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial w_{ki}} = \gamma_i \cdot F'(s_i) \cdot x_{ki} = \gamma_i \cdot F(s_i)(1 - F(s_i)) \cdot y_k \qquad (3.27)$$

Теорему доведено.

Теорема 3.

Похідна похибки щодо вагових коефіцієнтів нейронів для будь-яких двох шарів (за винятком вихідного) i та k (k передує i) визначається як

$$\frac{\partial E}{\partial w_{ki}} = \gamma_i \cdot F(s_i)(1 - F(s_i)) \cdot y_k \qquad (3.28)$$

Доведення:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial w_{ki}} = \gamma_i \cdot F'(s_i) \cdot x_{ki} = \gamma_i \cdot F(s_i)(1 - F(s_i)) \cdot y_k \qquad (3.29)$$

Теорему доведено.

На рис. 3.8 наведено блок-схему розробленого алгоритму навчання на основі розробленої функції похибок нейронів.

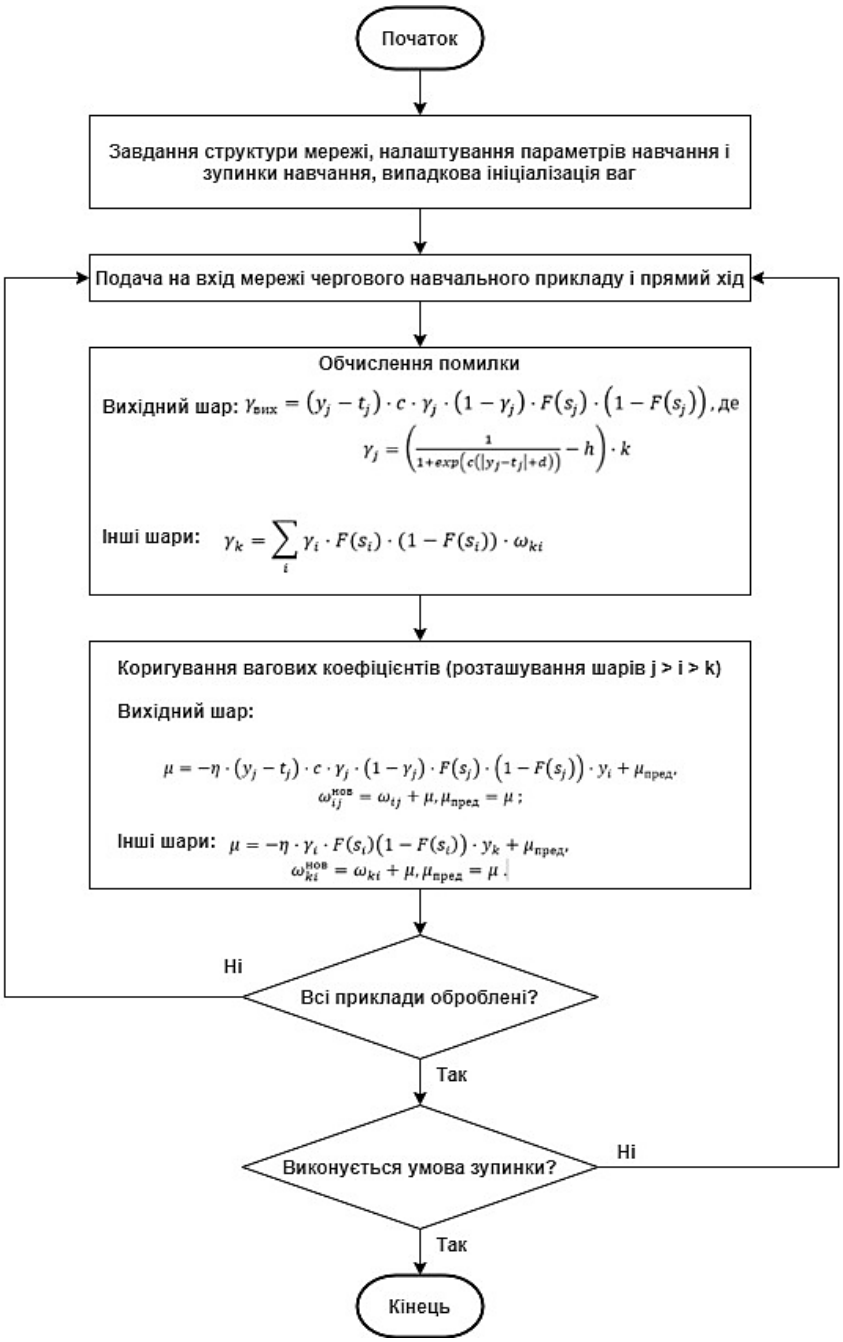


Рис. 3.8. Блок-схема розробленого алгоритму навчання нейронної мережі

3.5. Спрощення нейронної мережі

Передостанньою фазою інтелектуальної системи аналізу даних є спрощення нейронної мережі. Зменшення багатьох параметрів мережі та вхідних сигналів може служити багатьом цілям:

- привести нейронну мережу в логічну форму для дослідника;
- спрощення апаратної реалізації нейронної мережі;
- підвищення продуктивності нейронних мережевих програм;
- можливість вирішення задачі на основі меншої кількості вхідних даних і, як результат, скорочення часових та фінансових витрат на збір необхідного обсягу навчальної інформації.

Особливо важливим практичним результатом спрощення нейронної мережі, у зв'язку із задачею створення системи аналізу даних, є можливість привести нейронну мережу у форму, яка логічно зрозуміла досліднику. У цьому відношенні сьогодні представляється надзвичайно актуальною задача вдосконалення методів спрощення прихованої структури нейронної мережі.

Спрощення нейронних мереж можна зробити в наступних напрямках:

- зменшення розміру вхідних даних;
- мінімізація загальної кількості синапсів;
- вибір оптимальної архітектури.

На додаток до наведених тут напрямків, іноді також відображаються оптимізація кількості нейронів у кожному шарі та оптимізація кількості шарів. Останні два варіанти спрощення нейронної мережі, як правило, обмежуються оптимізацією загальної кількості синапсів.

Вибір оптимальної архітектури нейронної мережі пропонує безліч можливостей для спрощення мережі. Ця можливість оптимізації мережі тісно пов’язана з другим етапом аналізу даних - фазою визначення параметрів нейронної мережі - і вже обговорювалася у відповідному пункті.

Критерії логічної прозорості нейронних мереж

Після визначення вимог до архітектури ми визначаємо ряд критеріїв, відповідність яким робить мережу більш логічно прозорою. Багато критеріїв згадуються в [61, 80, 102]. На основі цих критеріїв можна оцінити, наскільки мережа наближається до ідеального логічно прозорого стану. За цими критеріями

дослідник може самостійно визначати пріоритети, залежно від власних суб’єктивних уявлень про логічну ясність, важливості одного критерію порівняно з іншим. Ось основні критерії з [61, 80, 102]:

1. Чим менше шарів нейронів у мережі, тим логічніша мережа. Кількість шарів вказується під час створення мережі і не може бути змінена, якщо мережа не спрощена. Зауважте, що випадки видалення всього шару на практиці рідкісні. Тому цей критерій слід враховувати саме на етапі синтезу архітектури мережі.

2. Зменшення кількості мережевих входів. Легше зрозуміти аргументи, що засновані на мінімальній кількості посилань. Цей критерій впливає не тільки на логічну прозорість, але також тісно пов’язаний із швидкістю навчання та якістю знайденого рішення. Методи вибору оптимального набору вхідних параметрів обговорювались вище.

3. Зменшення кількості нейронів у кожному шарі мережі. Мінімізуючи кількість сигналів, що видаються кожним шаром мережі, можна зберегти лише справді значущі проміжні нейрони і видалити непотрібні нейрони. Існують методи, що дозволяють визначити, який з нейронів у шарі має найменший вплив на загальну продуктивність і може бути видалений з достатньою точністю.

4. Зменшення кількості сигналів, що надходять до нейрона. Обмеження кількості нейронних зв’язків вводиться виходячи з того, що людина може одночасно оперувати достатньо малою кількістю просторів і намагаючись розглянути функціонування одного нейрона, буде легше працювати з нейроном з мінімальною кількістю вхідних сигналів.

5. Зменшення загальної кількості синапсів у мережі. Загальний критерій, який після виконання попередніх вимог виключає надлишкові міжнейронні зв’язки, якщо такі є.

6. Неоднорідний вхід логічно прозоріший, ніж синапси. Неоднорідний вхід нейрона постійний, тоді як синапс містить адаптивну вагу, тому на фазі спрощення слід докладати зусиль, перш за все, для видалення синапсів.

7. Необхідно перенести значення адаптивних ваг мережевих синапсів у кінцевий набір вибраних значень. Зазвичай рекомендується робити бінаризацію ваг синапсів, щоб отримати остаточну формулу залежності у типізовану форму, легшу для сприйняття.

Як буде показано нижче, є кілька способів отримати знання з навченої мережі. Вищезазначені критерії важливіші при реалізації методу вербалізації та менш важливі при отриманні правил у вигляді «якщо … то …». При другому підході критеріями 1, 4, 6, 7 можна знехтувати. Критерії 2, 3, 5 будуть значущими в будь-якому випадку, коли їх ітеративне зменшення від початку очевидно використовується як метод визначення оптимальної кількості нейронів. Навчена нейронна мережа містить усі можливі зв’язки між вхідними нейронами та нейронами прихованого шару, нейронами прихованого шару між собою, а також між прихованими та вихідними нейронами. Загальна кількість цих зв’язків, як правило, відносно велика, і аналіз їх значень для виявлення накопичених залежностей є складною і часто неможливою задачею. Щоб вирішити цю задачу, можна спробувати позбутися зайвих з’єднань, тобто посилань, які не збільшують похибку мережі після видалення - цей процес називається розрідженням. Після розрідження мережі з надмірною кількістю нейронів можна очікувати значного зменшення кількості нейронів та синапсів, що їх з’єднують. Отримана мережа стане простішою в експлуатації.

Загальна кількість з'єднань у навченій мережі становить $\sum N_i N_{i+1}$, для $i = 0, \ldots, n$, де N_i - кількість нейронів у i -му шарі, N_0 - кількість нейронів у вхідному шарі, N_n - кількість нейронів у вихідному шарі. Існують підходи до спрощення нейронної мережі шляхом усунення міжнейронних зв’язків, коли умова $\max|w_{il}| < 4\eta$ задовольняється, де w_{il} - вага з'єднання, що з'єднує i -й і l-й

нейрони, розташовані на сусідніх шарах, $\eta < 0{,}5$. Однак твердження «міжнейронний зв'язок із найменшою вагою є першим кандидатом на видалення за спрощеною процедурою» не видається таким очевидним і правдивим.

Інший спосіб відбору кандидата для видалення міжнейронних зв’язків, заснований на оцінці значущості ваг з'єднання за допомогою індексу чутливості, називається процедурою контрастування. Вперше процедура контрастних нейронних мереж на основі показників чутливості була описана в [102]. Цей підхід видається більш правильним порівняно з раніше описаним підходом. На передостанньому етапі аналізу даних - спрощенні мережі - слід застосовувати процедуру контрастування.

Процедура контрастування заснована на оцінці важливості ваг посилань у мережі. У джерелі [100] викладено основні цілі контрастування: спростити технічне впровадження мережі та зробити зрозумілішими навички мережі - продемонструвати (явно виразити) знання, набуті мережею під час навчання. Результати експериментів на контрастних нейронних мережах опубліковані в [102]. Існують також підходи, в яких не використовуються показники чутливості [120].

Ці процедури для обчислення показників значущості вхідних сигналів мережі та її налаштованих параметрів дозволяють видалити лише найменш значущі синапси. Однак може знадобитися спрощення мережі, вилучивши з неї нейрон.

Після процедури розрідження може виявитись, що нейрон не має єдиного вхідного сигналу - такий нейрон можна виключити з мережі, видаливши як сам нейрон, так і його синапси, які з’єднують його з нейронами наступного шару. І навпаки, якщо порівняти всі синапси, що з’єднують поточний нейрон з нейронами наступного шару, цей нейрон з усіма вхідними синапсами може бути видалений [120].

Таке видалення нейронів можна розглядати як ще один результат, коли працює процедура контрастного синапсу, але не як самостійну процедуру видалення нейронів.

Існує також незалежна процедура видалення нейронів, заснована на принципі роботи подвійної мережі. Градієнт функції оцінки використовується для обчислення показників значущості елементів мережі та сигналів, що дозволяє видалити ті елементи, для яких знайдений показник менше певного заздалегідь визначеного значення.

Можна запропонувати альтернативний варіант, який не включає додаткові розрахунки. В якості показника значущості нейрона ми можемо взяти суму показників значущості всіх його синапсів і всіх синапсів, через які нейрон посилає відповідь. Цей шлях оцінює сукупний ефект усіх нейронних синапсів на зміну функції оцінки. Чим менша сума показників значущості синапсу, тим менш важливий нейрон для прийняття рішень.

3.6. Витяг правил з нейронної мережі

Формулювання задачі вилучення правил із навченої нейронної мережі дозволяє з певністю класифікувати методи нейроаналізу як інструменти технології виявлення знань, що відкриває нові сфери застосування нейронних технологій. Цей етап роботи з нейронними мережами видається особливо актуальним сьогодні.

Ми можемо запропонувати дві технології отримання правил з нейронної мережі:

- вербалізація нейронної мережі,
- отримання правил, наприклад «якщо … то …».

Вербалізація нейронних мереж полягає у синтезі формул, заснованих на функціонуванні нейронної мережі. Можна використовувати цей метод, щоб

спробувати скинути лише набір правил, який нейронна мережа використовує для вирішення задачі. Цей метод можна легко реалізувати в програмному забезпеченні, але результат, отриманий за допомогою нього, легко зрозуміти лише для невеликих розмірів досліджуваної нейронної мережі. Розглядаючи досить велику нейронну мережу за допомогою методу вербалізації, неможливо встановити правила її роботи, тому вихідну нейронну мережу потрібно розділити на кілька простіших, які послідовно посилають сигнали з вихідного рівня попередньої мережі. У цьому випадку вихід досліджуваної нейронної мережі буде виходом останньої з набору мереж.

Як видно з цього прикладу (табл. 3.2), словесне викладення не підходить для простого логічного розуміння. Можна спробувати виправити цей недолік, переклавши ваги всіх міжнейронних мережевих з'єднань на одне значення із заздалегідь визначених наборів (назвемо їх - набір ваг) - для виконання бінаризації синапсів. Найкращих результатів з точки зору отримання зрозумілих правил можна досягти, встановивши мінімально можливу кількість елементів у згаданому наборі ваг, в ідеалі набір ваг може складатися лише з одного елемента. Однак отримання такого ідеального результату за допомогою методу бінаризації може призвести до втрати здатності нейронної мережі вирішувати задачу.

Наступну послідовність дій можна розробити для побудови оптимального набору ваг (алгоритм 1):

1. Додаємо перший елемент до набору ваг - 0.
2. Пам’ятаємо поточне значення ваг синапсів.
3. Замінюємо поточне значення ваги кожного синапсу на найближче значення з набору ваг.
4. Перевіряємо продуктивність нейронної мережі. Якщо мережа вирішує задачу з правильною точністю, був створений оптимальний набір ваг, якщо ні, скидаємо еталонні ваги та переходимо до кроку 5.

5. Додаємо до набору ваг - якщо це перший прохід - значення 1 і - 1, інакше - середнє арифметичне між кожними двома сусідніми елементами. Переходимо до кроку 3.

Таблиця 3.2

Вербалізація нейронної мережі, де $Поле_x$ - входи нейронної мережі, тобто $Нейрон_{x\_y}$ – y-й нейрон шару x, N - кількість входів, L - кількість шарів, M_i - кількість нейронів в i-му шарі

Формули 1-го прихованого шару:	
$Нейрон_{1\_1}$	0,02*$Поле_1$+0,083* $Поле_2$-0,27* $Поле_3$-0,15* $Поле_N$-0,137
$Нейрон_{1\_2}$	-0,004*$Поле_1$+0,337* $Поле_2$-0,71* $Поле_3$-0,68* $Поле_N$+0,5
…	…
$Нейрон_{1\_M1}$	0,32*$Поле_1$-0,1789* $Поле_2$-0,55* $Поле_3$-0,2* $Поле_N$-0,113
Формули 2-го прихованого шару:	
$Нейрон_{2\_1}$	0,07* $Нейрон_{1\_1}$+0,098* $Нейрон_{1\_2}$-0,01* $Нейрон_{1\_3}$-0,19* $Нейрон_{1\_M1}$-0,1703
$Нейрон_{2\_2}$	-0,94* $Нейрон_{1\_1}$+0,0004* $Нейрон_{1\_2}$-0,1* $Нейрон_{1\_3}$-0,098* $Нейрон_{1\_M1}$+0,2
…	…
$Нейрон_{2\_M2}$	0,26* $Нейрон_{1\_1}$-0,18* $Нейрон_{1\_2}$-0,5895* $Нейрон_{1\_3}$-0,19* $Нейрон_{1\_M1}$+0,13
…	…
Формули вихідного шару:	
$Нейрон_{L\_ML}$	0,74* $Нейрон_{L\_1}$-0,287* $Нейрон_{L\_2}$-0,3* $Нейрон_{L\_3}$+0,6* $Нейрон_{L\_ML}$-0,02

У представленому алгоритмі будуть поступово обчислюватися набори шкал для бінаризації – «0», «-1,0,1», «-1, -0,5, 0, 0,5, 1», «-1, -0,75 , - 0,5, - 0,25, 0, 0,25, 0,5, 0,75, 1» та ін., доки бінаризована нейронна мережа не збереже здатності вирішувати задачу. Найкращих показників представленого алгоритму можна досягти, якщо спробувати перекваліфікуватися на четвертому етапі у разі втрати потужності мережі, використовуючи лише ваги з набору ваг як можливу вагу синапсу.

У порівнянні з розглянутим методом вербалізації, метод отримання правил у формі «якщо … то …» дасть можливість отримати логічно більш зрозумілі правила для дослідника.

Ось офіційний опис можливого підходу до вилучення правил із навченої нейронної мережі у формі «якщо … то …»:

- Обмеження набору можливих значень виходу нейронів скінченним числом.
- Розробка таблиці правил.
- Отримання правил в явному вигляді.
- Оптимізація прийнятих правил.

Стани нейронів прихованого шару є неперервними, що ускладнює отримання правил з нейронної мережі. Першим кроком в алгоритмі вилучення правил є вирішення цієї задачі. Для цього всі значення, які отримують нейрони прихованого шару, повинні бути включені в набір (скажімо, набір дій), кількість елементів якого обмежена невеликим значенням.

Набір дій можна отримати, використовуючи алгоритм, подібний до раніше розглянутого першого алгоритму, але існує більш гнучкий механізм. Це пов’язано з тим, що можливі значення активації нейронів прихованого шару згруповані, а кластерні центри розташовані в наборі видів діяльності. Оскільки, як зазначалося раніше, кількість елементів у наборі дій обмежена, слід також вибрати невелику кількість кластерів.

Кластеризація для кожного прихованого шару виконується окремо і незалежно від інших шарів. Кращих результатів можна очікувати від кластеризації кожного нейрона окремо, але це призведе до подальших розрахунків. Опис алгоритму кластеризації значень дій нейронів у прихованих шарах:

1. Вибирається значення параметра $\varepsilon \in (0,1)$. Нехай h_1 - активність цього нейрона після представлення першого вектора в мережі. Вкажемо кількість кластерів $N_{clust} = 1$, положення кластера

$$A_{clust}(1) = h_1, num(1) = 1, sum(1) = h_1$$

2. Для всіх векторів навчального набору $k = 1, \dots, K$

- визначається активність прихованого шару нейрона h
- якщо є індекс j такий, що

$$|h - A_{clust}(j)| = \min_{j \in \{1,\dots,N_{clust}\}} |h - A_{clust}(j)|,$$

$$|h - A_{clust}(j)| \le \varepsilon, \text{ то}$$

$$num(\vec{j}) := num(\vec{j}) + 1, sum(N_{clust}) := sum(N_{clust}) + h,$$

інакше

$$N_{clust} = N_{clust} + 1, A_{clust}(N_{clust}) = h, num(N_{clust}) = 1, sum(N_{clust}) = h$$

3. Замінимо A_{clust} середнім значенням активації нейронів, об'єднаних в один і той же кластер: $A_{clust}(j) := sum(j)/num(j), j = 1, \dots, N_{clust}$

4. Перевірка точності кластеризації мережі при заміні значень активації нейрона прихованого шару на $A_{clust}(j)$. Якщо точність кластеризації нижче вказаного значення, зменшуємо значення ε і повертаємося до кроку 1.

Після використання того чи іншого методу, після заповнення набору дій, всі дії проміжних нейронів замінюються найближчим значенням із набору дій, перевіряється точність групування мережевих об’єктів. Якщо вона є прийнятною, тоді підготовка до вилучення правил закінчується.

Можливо, за допомогою технологій нейронних мереж (таких як мережі Хопфілда) можна досягти більш плавної кластеризації.

Отже, ми маємо нейронну мережу, яка із задовільною точністю вирішує задачу, кількість значень активації нейронів якої є кінцевою величиною. Тепер ми можемо скласти таблицю зв’язків між шарами нейронної мережі, склавши всі можливі комбінації між значеннями активації нейронів.

На цьому етапі неможливо оцінити кількість правил, що з'єднують вхідний рівень і 1-й прихований шар. Це пов'язано з тим, що вхідний рівень, тобто вхідні параметри, не був згрупований. Відсутність кластеризації вхідних параметрів на фазі тестування нейронної мережі дозволило зробити висновок, що мережа працює з достатньою точністю на всіх раніше розглянутих фазах. Отже, якщо після групування вхідних параметрів будуть прийняті неправильні правила - це не буде похибкою в навчанні мережі або похибкою в групуванні виходів прихованих нейронів мережевого рівня, це буде похибкою кластеризації у вхідних параметрах. Для усунення похибок у цьому випадку необхідно буде перерозподілити вхідні параметри на нові набори, кількість яких може бути збільшена порівняно з початковою кластеризацією. Для виключення цих помилок не потрібна перепідготовка та подальше навчання нейронних мереж.

Назвемо повний набір правил (між вхідним та 1-м шаром, між 1-м і 2-м шаром, ..., між $(n-1)$ і n-м шаром, між n-м і вихідним шарами, де n-кількість прихованих рівнів) - правила роботи нейронної мережі. Якщо наша мета - зрозуміти, за якими законами працює досліджувана система, а не нейронні мережі, то ситуація з кількістю правил спрощується. У цьому випадку кількість правил буде дорівнювати кількості правил для зв'язування вхідного та вихідного шарів. Давайте назвемо цей набір правил - правилами роботи системи.

Кількість правил для роботи системи, на відміну від кількості правил для роботи нейронної мережі, не залежить від кількості прихованих шарів, що дозволяє використовувати складні нейронні мережі з великою кількістю як шарів, так і нейронів у шарах. Недостатня залежність кількості правил роботи задачі від кількості прихованих шарів дозволяє відмовитися від їх кластеризації, якщо це призводить до поліпшення роботи нейронної мережі.

Кількість правил для виконання задачі можна розрахувати за формулою:

$$k = \prod_{i=1}^{n} j_i$$

де k - кількість правил, n - кількість входів, j_i - кількість кластерів на i-му вході.

Як правила роботи системи, так і правила роботи нейронної мережі можуть бути зведені до невеликого, оптимального набору правил, що точно описує вивчену задачу.

Для отримання цього оптимального набору можна використовувати традиційні методи з алгоритмів побудови дерева прийняття рішень:

- Методи, які обчислюють швидкість, з якою працює кожне правило, і видаляють ті, що нижче заданої швидкості.
- Методи, що контролюють невідповідність правил, тобто виявлення правил, реалізація яких іноді призводить до неправильного вирішення задач. Якщо нейронна мережа вирішує 100% прикладів без помилок, а вхідні параметри не групуються, ці методи не зменшать набір правил.

Ситуації можливі, якщо не для кожного правила, отриманого таким чином буде приклад із набору, тоді як поле передбачуваного класу завжди заповнюється на основі роботи нейронної мережі. Той факт, що визначаємо клас для кожного правила, незалежно від наявності прикладів навчання для цього правила, забезпечує значні переваги в отриманні оптимального набору правил (особливо отримання одного узагальнюючого правила з кількох правил) порівняно з традиційними методами аналізу даних. Загалом, запропонований спрощений набір функціональних правил можна спростити шляхом створення дерева прийняття рішень на основі знайдених правил, що призводить до значного зменшення кількості правил, але, як правило, призводить до погіршення повноти правил та їх якості.

Методи, описані в цьому пункті, дозволяють отримувати знання з даних у формі, яка логічно зрозуміла досліднику за допомогою технології роботи нейронних мереж.

Висновки до розділу 3

Детально розглянуто фазу підготовки даних для їх подальшого аналізу, запропоновано новий метод нормування нелінійних даних на основі статистичних характеристик даних. Вказано на роль нейронних мереж у запропонованому підході до аналізу даних. На основі класифікації нейронних мереж було зроблено обґрунтований вибір універсальної логічно прозорої архітектури нейронної мережі, проведено аналіз правил навчання, застосування яких можливе для цієї архітектури. Надано докладні інструкції з налаштування та навчання мережі, описано алгоритм зворотного поширення похибок, детально розглянуто існуючі методи спрощення нейронних мереж та надано рекомендації щодо їх використання. Надано визначення наборів правил - правила роботи системи, правила роботи мережі, запропоновано метод вилучення цих правил із навченої нейронної мережі. На основі існуючих алгоритмів описані можливості оптимізації отриманого набору правил.

Тому можна стверджувати, що створено весь необхідний формальний апарат, і можемо перейти до перевірки ефективності підходу на тестових та реальних задачах. Для такого контролю необхідно виконати програмну реалізацію запропонованих алгоритмів у вигляді сучасної комп’ютерної системи для інтелектуального аналізу даних.

РОЗДІЛ 4

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ЕФЕКТИВНОСТІ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЙ НЕЙРОННИХ МЕРЕЖ

4.1. Вибір мови програмування для моделювання нейронної мережі системи

Організація штучної нейронної мережі була описана вище. Проте паралельність та розподіл нейронної мережі важко уявити в архітектурах популярних мов програмування, таких як C / C ++, C #, Java і так далі.

Причиною проблеми в даному випадку є те, що важко мислити та розробляти нові ідеї мовою, яка для цього не призначена і не має відповідних елементів.

З архітектурної точки зору, мови для моделювання нейронних мереж значно відрізняються від мов програмування, що використовуються в даний час. Щоб розробити надійну мережу та зосередитись на задачах штучного інтелекту, не турбуючись про мову програмування, потрібні інструменти, які точно відображають цю сферу. Вони можуть значно спростити концептуальну сторону розвитку та вираження ідей.

Наразі для моделювання нейронної мережі значного поширення набули мови програмування Python [123, 124, 125] (за наявність фреймворків для розробки та спеціалізованих бібліотек) та Prolog [126, 127, 128, 129] (за простоту алгоритмів, що базуються на доведенні того, чи є задане цільове твердження наслідком з напередзаданих фактів та правил).

Отже, мова програмування для написання нейронної мережі повинна задовольняти наступні вимоги:

1) Біологічні нейронні мережі складаються з незалежних, паралельних, розподілених процесорних одиниць - нейронів. А отже архітектура нашої мови програмування вимагає підтримки таких елементів, незалежних процесів, які можуть запускатись паралельно і легко розподілятись на сучасному обладнанні.

2) Біонейрони спілкуються між собою за допомогою сигналу. Мова програмування повинна забезпечувати комунікацію процесів за допомогою повідомлень або сигналів.

3) Нейронна мережа потребуватиме стійкості, отже мова програмування повинна мати можливість легко відновлюватись після помилок.

4) Якщо довільний компонент мережі вимикається або «зависає», система повинна мати можливість його автоматично відновити або перезапустити. Процеси повинні мати кілька рівнів безпеки, щоб відстежувати продуктивність один одного, контролювати помилки та допомагати відновлювати дані з пошкоджених елементів.

5) Хоча нейронна мережа піклується про власне навчання, впроваджуючи нові ідеї, культивуючи та накопичуючи досвід, є також те, що біологічні системи не можуть зробити зі своїм розумом - біологічні системи не можуть змінити свої нервові структури. Однак, для моделювання нейронної мережі мова програмування повинна дозволяти «гарячу» заміну коду, щоб система могла виправляти помилки та оновлюватись в режимі реального часу.

6) Якщо система може працювати тривалий час, помилки повинні бути локальними і бути виправлені самою системою.

7) Також враховуйте, що нейромережеві системи повинні керувати багатьма складними системами, а мова повинна підтримувати просте створення драйверів для різних пристроїв.

Усі ці інструменти також пропонує функціональна мова програмування Erlang [130, 131, 132], яка, наразі, не є такою ж популярною через свою складність як Python та Prolog, хоча і має дещо більше плюсів. Відповідно до вимог, які

накладаються на вибір мови програмування для штучної нейронної мережі, Erlang володіє наступними перевагами:

1) Інкапсуляція примітивів - існує ряд механізмів для обмеження наслідків відмови. Тобто можливість ізолювати процеси, щоб вони не нашкодили один одному.

2) Паралельність - мова підтримує простий механізм створення паралельних процесів і маршрутизації повідомлень між процесами. Завдяки цьому є ефективним перемикання контексту між процесами та обмін повідомленнями. Паралельні процеси також розумно організовують час на CPU, щоб жоден із процесів не монополізував його.

3) Визначення примітивних помилок – дозволяє процесу спостерігати за іншим процесом та помічати його завершення з довільної причини.

4) Прозорість виявлення - якщо відомий ідентифікатор процесу, то є можливість надіслати йому повідомлення.

5) Динамічне оновлення коду - є можливість динамічно змінювати код у працюючій системі. Оскільки багато процесів виконують один і той же код, є механізм, за допомогою якого процеси можуть запускати «старий» та «новий» код.

6) Стабільна архітектура, яка переживе крах.

7) Оновлення коду - дозволяє оновлювати код у запущеній системі.

8) Інфраструктура - для запуску / зупинки системи, ведення журналу (логів) помилок тощо.

Erlang як мова була розроблена для створення розподілених, заснованих на процесах, парадигматично надійних та стійких до несправностей систем паралельного обміну повідомленнями.

У випадку з цією мовою існує дуже точне призначення області застосування розвитку нейрокогнітивних систем. [131, 132, 133]

4.2. Функціональне призначення програми та інструкція роботи

Розроблену систему інтелектуального аналізу даних було створено з метою виявлення прихованих закономірностей, зв’язків та функціональних законів системи, що використовується, та представлення отриманих знань у формі, зрозумілій та доступній для дослідника. Використання нейронних мереж як математичного ядра в програмі дозволяє вирішувати задачі в будь-якій області, в якій можна зібрати приклади роботи об'єкта, що досліджується.

Щоб зробити роботу з програмою більш гнучкою, ефективною та зручною для користувача, програма була реалізована у такому вигляді, що дозволяє виконувати процес ідентифікації знань поетапно.

Програма може вибирати різні варіанти нормування вхідних даних для збільшення їх інформаційного наповнення за рахунок зменшення ентропії. Можна задати кілька параметрів навчання нейронної мережі, щоб зупинити процес навчання. Процес навчання візуалізується в програмі за допомогою динамічної діаграми помилок навчання на тестових та навчальних наборах даних. Також можна налаштувати параметри мережі, такі як швидкість навчання, крутний момент, значення сигмоїдального нахилу. Якщо користувач не має достатнього досвіду використання нейромережевих технологій для вирішення досліджуваної задачі, для пошуку рішення можна використовувати значення за замовчуванням. На нейронну мережу, реалізовану в програмі, не застосовуються жодні обмеження щодо кількості шарів або кількості нейронів у кожному шарі.

Першим кроком у роботі з програмою є створення та навчання нейронної мережі. Для цього необхідно вибрати тип нейронної мережі, розробити конфігурацію, створити навчальну вибірку даних і навчити мережу. Зверніть увагу, що всі процеси автоматизовані, за винятком процесу створення шаблону навчання, який повністю покладається на плечі користувача, оскільки він присвячений певному завданню.

Після отримання результатів їх слід оцінити та зробити висновки щодо доцільності та ефективності використання мережі. Оцінка результатів навчання може базуватися на статистичних характеристиках базових значень мережі та відповідних графіках. Під час роботи з програмою користувач може змінювати параметри навчальних даних та процесу навчання. Зверніть увагу, що для створення та навчання нейронної мережі джерела даних повинні бути пов’язані.

Першим кроком у роботі з програмою є підключення до бази даних - вибір таблиці зі стандартного файлу. Також можна тут переглянути інформацію, яку містить база даних (рис. 4.1).

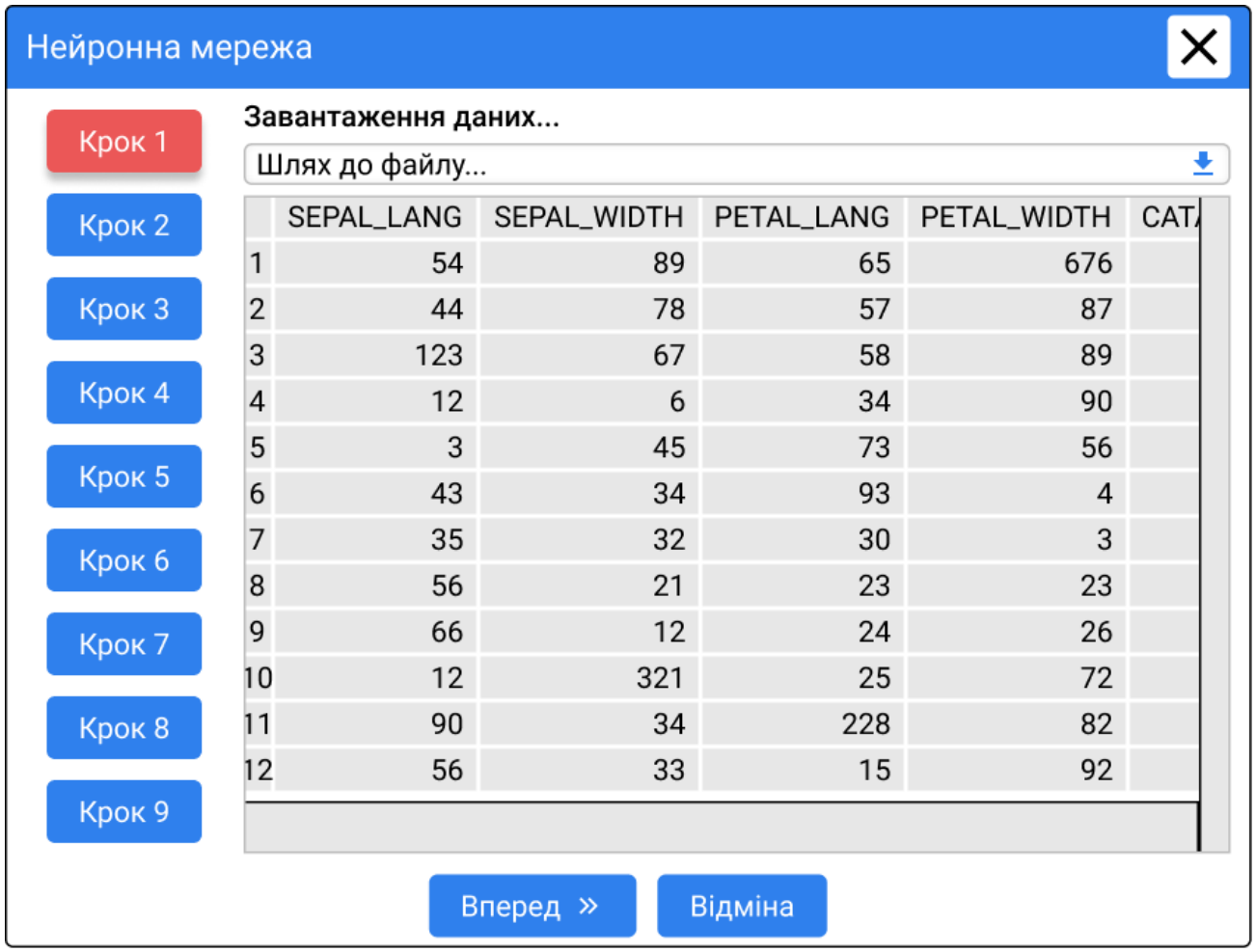


Рис. 4.1. Крок 1

На другому кроці роботи з програмою користувач встановлює параметри навчальних даних, визначивши властивості кожного з полів, які містить набір (рис. 4.2). У вікні програми на цьому кроці відображається список полів, що входять до навчального набору даних. У правій частині вікна є елементи для введення параметрів кожного поля. Щоб налаштувати параметри поля, потрібно вибрати поле зі списку, при цьому параметри будуть мати значення, яке встановили або значення за замовчуванням, якщо до цього не встановлювали.

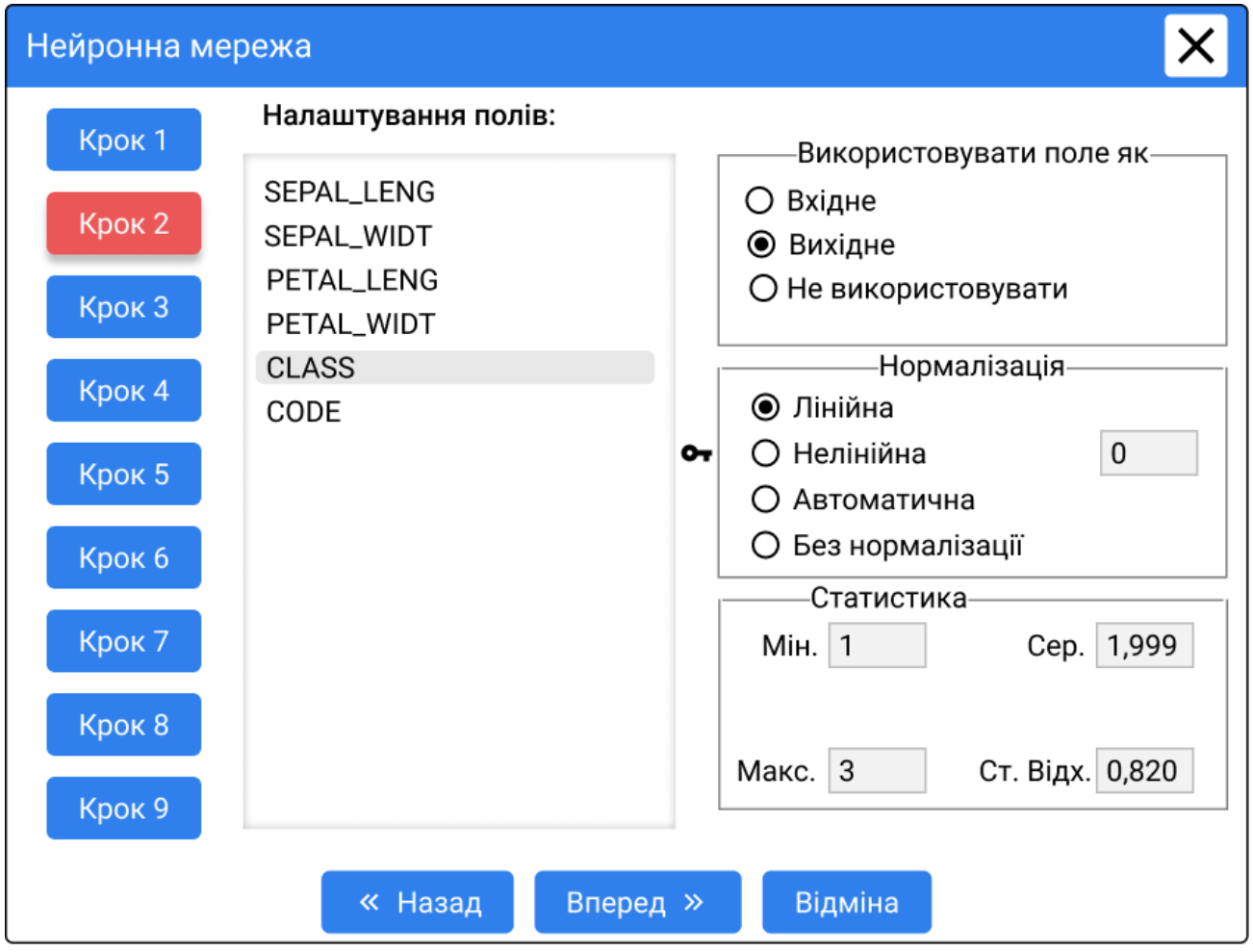


Рис. 4.2. Крок 2

Для кожного поля можна налаштувати такі параметри:

- «Використовувати поле як» - дозволяє визначити роль поля у навчанні мережі. Можливі наступні варіанти:

1) «Вхідне» - значення в цьому полі завантажуються на вхід мережі як навчальний набір.

2) «Вихідне» - значення в цьому полі є еталонними значеннями і використовуються для порівняння з вихідними даними мережі. Цільових полів може бути декілька.

3) «Не використовувати» - поле не використовуватиметься під час навчання мережі. Поле виключається із кількості полів, що використовуються за бажанням користувача.

- «Нормалізація» - це службовий параметр, який визначає шлях нормалізації даних під час обробки. Нормалізація необхідна, оскільки програма може працювати лише з нормалізованими даними.

Розділ «статистика» дозволяє отримати інформацію про основні статистичні характеристики обраного поля - максимальне, мінімальне, середнє значення та статистичне відхилення. Виконавши всі необхідні налаштування для поля навчального набору, натисніть Далі та перейдіть до наступного кроку програми.

На кроці 3 (рис. 4.3) коригуються основні параметри алгоритму: «момент», «швидкість навчання», нахил функції активації нейрона. Для алгоритму зворотного розповсюдження потрібно налаштувати три параметри.

1) «Швидкість навчання» - параметр алгоритму навчання, який контролює розмір кроку для ітеративної корекції ваги. Приймаються значення від 0 до 1. Якщо розмір кроку великий, конвергенція (процес зближення) відбувається швидше, але існує ризик того, що рішення буде пропущене. З іншого боку, якщо розмір кроку невеликий, для навчання буде потрібно занадто багато ітерацій. На практиці розмір кроку робиться пропорційним нахилу,

зберігаючи алгоритм наближеним до мінімуму. Правильний вибір швидкості навчання залежить від конкретної задачі і зазвичай робиться емпіричним шляхом.

2) «Момент» - це величина, яка впливає на корекцію синаптичних ваг під час навчання мережі. Діапазон - від 0 до 1, а рекомендоване значення - 0,9 ± 0,1.

3) «Параметр сигмоїди» - потрібно встановити нахил (крутизну) для сигмоїдальної функції активації нейронів.

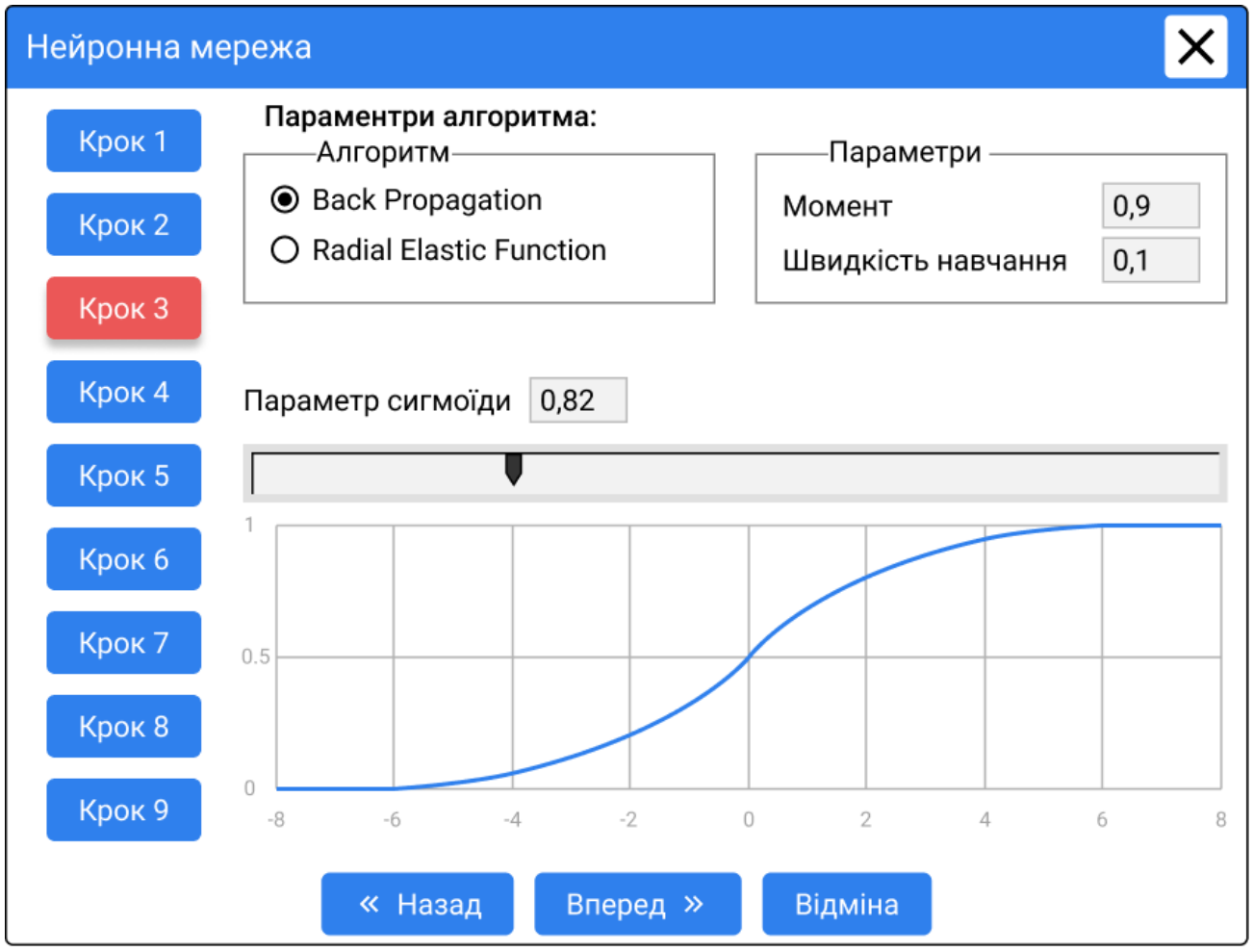


Рис. 4.3. Крок 3

На кроці 4 (рис. 4.4) встановлюються параметри нейронної мережі. У цій програмі немає обмежень щодо кількості сформованих шарів або кількості нейронів у кожному шарі.

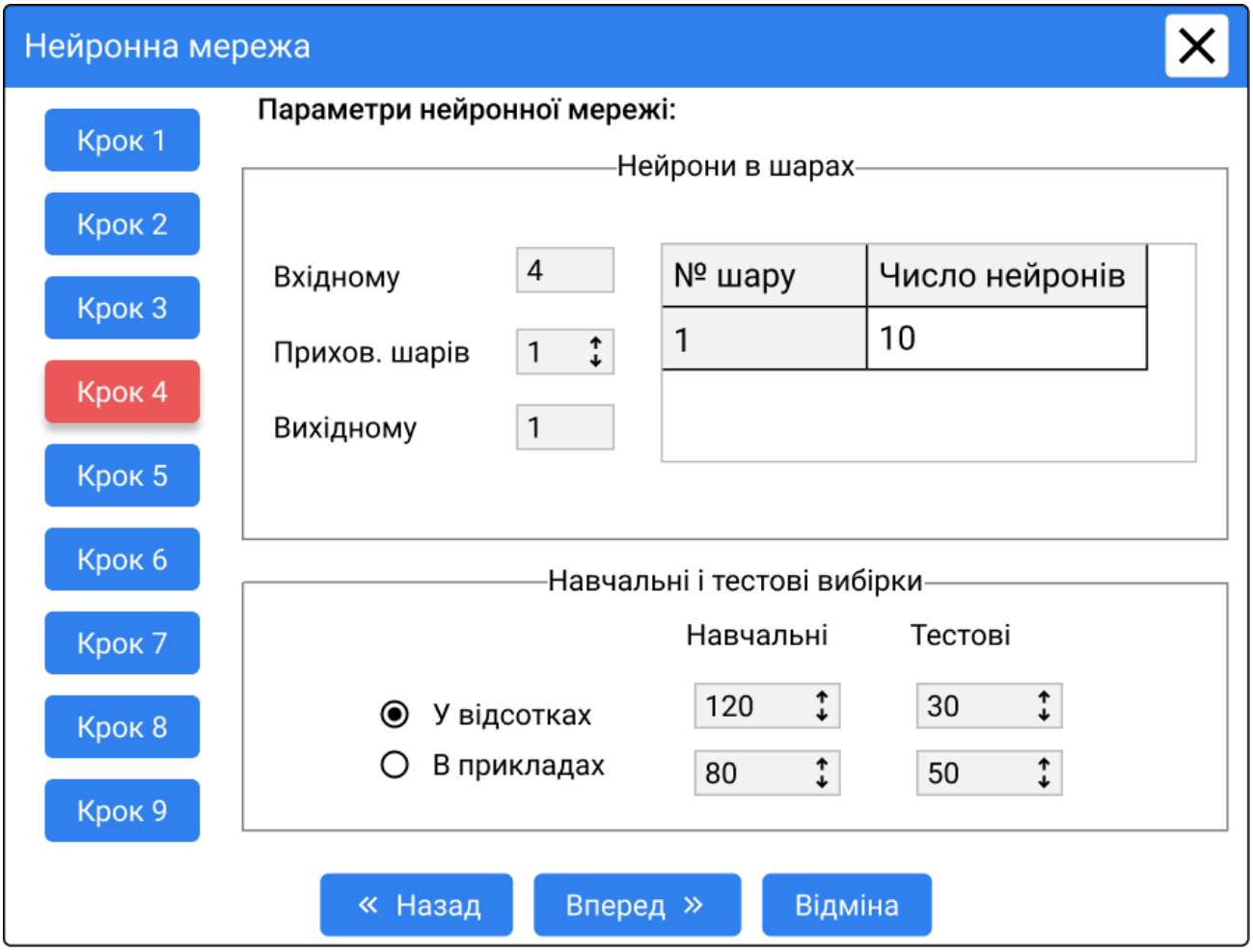


Рис. 4.4. Крок 4

«Нейрони в шарах» - цей параметр дозволяє визначити параметри нейронної мережі, такі як кількість прихованих шарів та кількість нейронів у кожному прихованому шарі. Кількість нейронів у вхідному та вихідному шарах залежить від розмірності задачі, тобто кількості полів введення та виведення, введених на попередньому кроці програми. Щоб збільшити або зменшити

кількість прихованих шарів, встановіть значення для поля «Прихованих шарів». Потім введіть кількість нейронів, яку містить кожен шар.

Для покращення контролю якості навчання нейронних мереж навчальний набір можна розділити на два набори – навчальні та тестові. Тестовий набір використовується для перевірки можливостей уже навченої мережі. Параметри, встановлені в розділі «Навчальні і тестові вибірки», використовуються для створення навчальних та тестових даних. Можна визначити два варіанти створення навчальних та тестових наборів:

1) « У відсотках» - розмір наборів визначається як відсоток від загальної кількості прикладів.

2) «В прикладах» - розмір наборів визначається кількістю прикладів, які має містити кожен набір.

Щоб відрегулювати розмір наборів, потрібно ввести відповідний відсоток або кількість прикладів у поля «Навчальні» та «Тестові».

На наступному кроці програми (рис. 4.5) потрібно вибрати умови для зупинки процесу навчання. Параметри, які відповідають умові закінчення навчання після їх досягнення, однакові для всіх алгоритмів. Умова не застосовуватиметься, доки не буде встановлено прапорець. Цей розділ містить два набори параметрів зупинки – для навчального та тестового набору.

«Після досягнення запису» - для встановлення кількості навчальних ітерацій (епох) для нейронних мереж, в яких представлені всі зразки навчального набору з подальшою їх верифікацією на тестовому наборі. Процес навчання закінчується, коли досягається вказана кількість епох.

«Максимальна похибка менша ніж» - максимальна квадратична похибка в навчальному (тестовому) наборі. Процес навчання завершується, як тільки параметр опускається нижче встановленого значення.

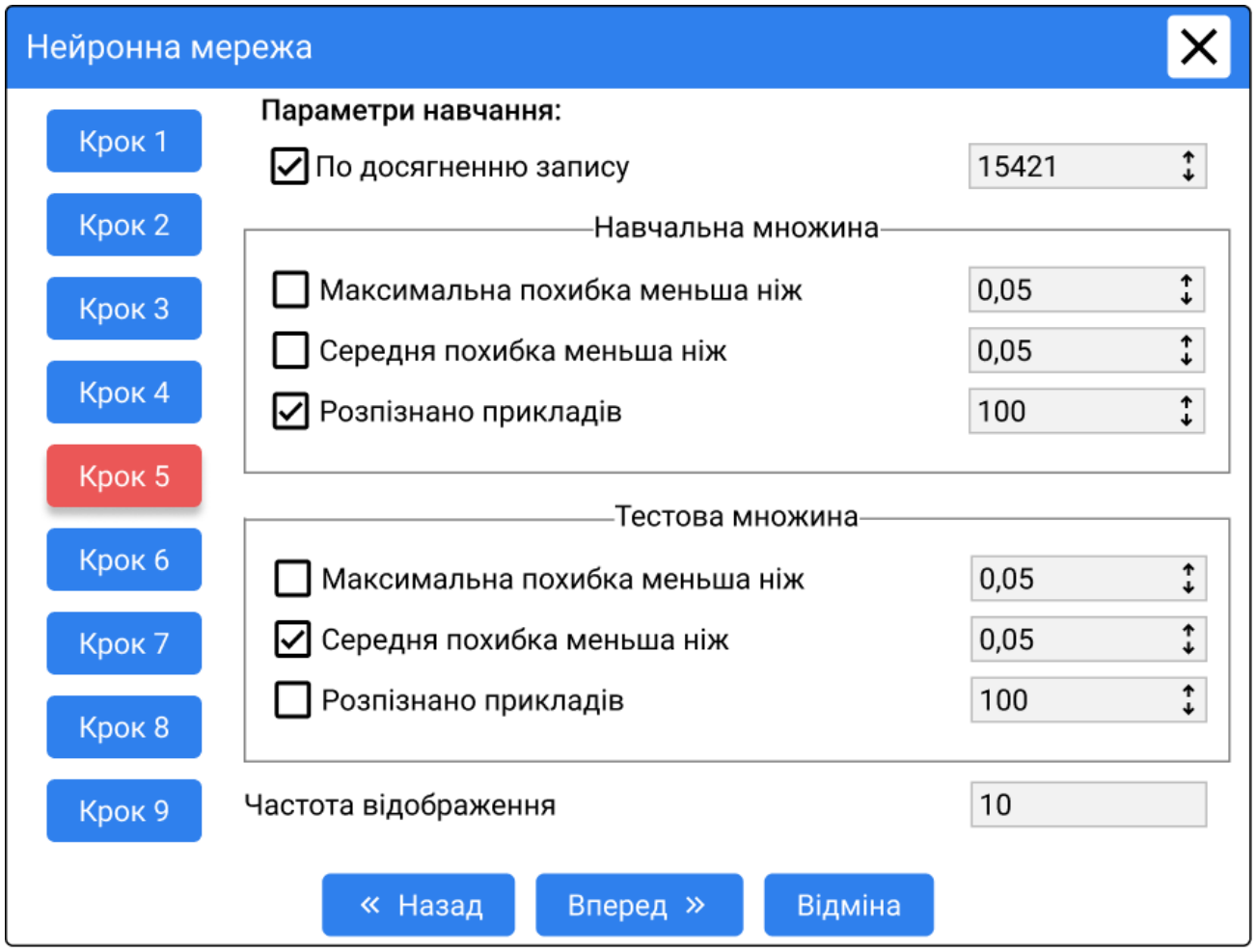


Рис. 4.5. Крок 5

«Середня похибка менша ніж» - середня квадратична похибка на навчальному (тестовому) наборі. Умова зупинки - коли значення похибки нижче встановленого значення.

«Розпізнано прикладів» - кількість розпізнаних прикладів у навчанні (тесті), після досягнення певного відсотка яких процес навчання зупиняється.

У полі «Частота відображення» вказують кількість епох, після яких інформація буде додана до графіка.

Навчання мережі є наступним кроком програми (рис. 4.6), в якому сама нейронна мережа навчається з параметрами, зазначеними на попередніх кроках. Верхню частину вікна займає група графіків, що відображають зміни

максимальних (суцільна лінія) та середніх (пунктирна лінія) квадратичних помилок для навчальних та тестових наборів.

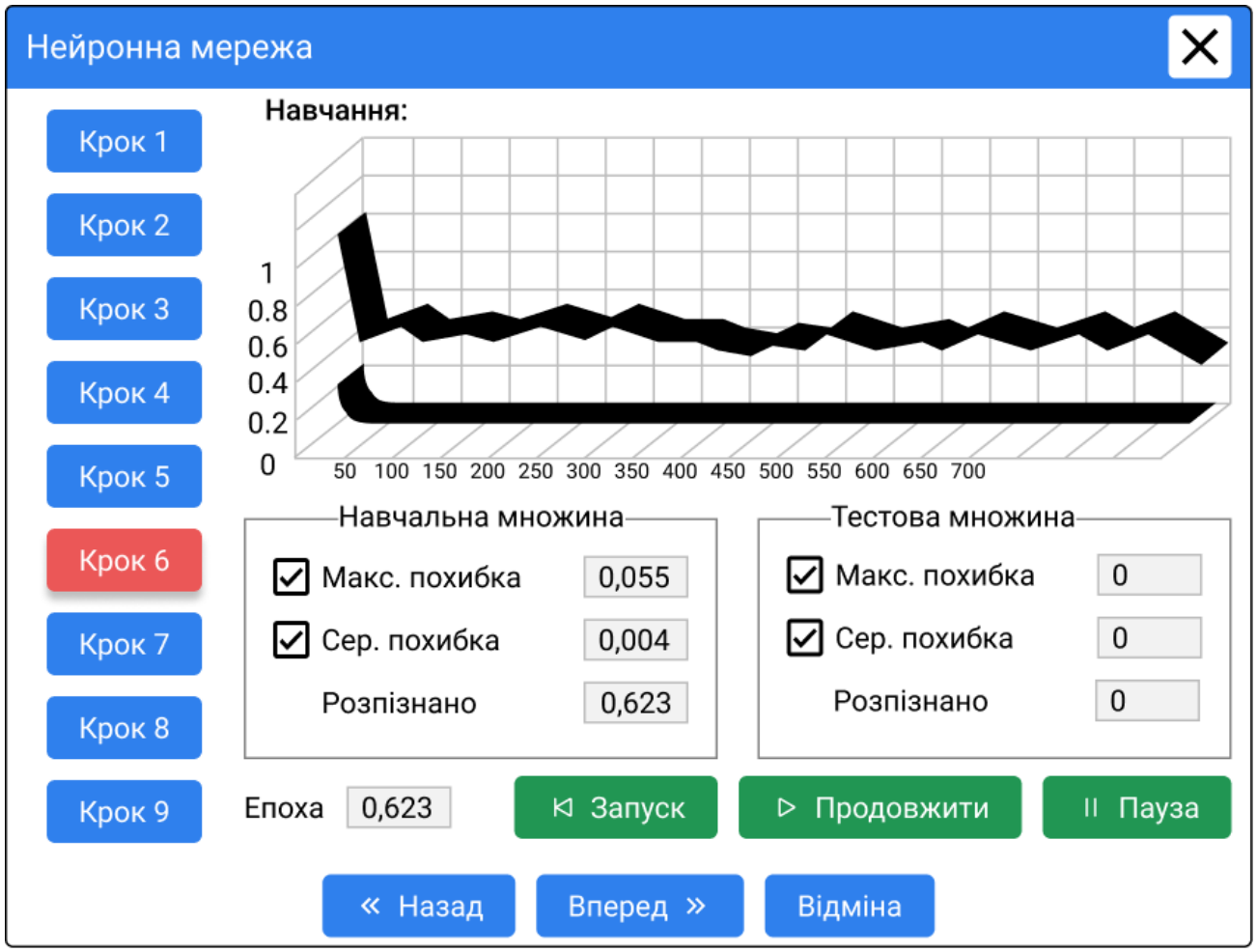


Рис. 4.6. Крок 6

Поле графіка залишається порожнім до початку навчального процесу. Під полем графіка знаходиться область, яка показує поточні максимальні та значення середньоквадратичних похибок для навчальних та тестових наборів, а також відсоток розпізнаних прикладів.

Це дозволяє користувачеві контролювати процес навчання мережі не тільки за допомогою графіків, але і точних значень помилок, які відображаються у відповідних полях. Він також показує кількість епох, що минули з початку

навчання. Щоб розпочати навчальний процес, потрібно натиснути кнопку «Запуск».

Процес навчання закінчується автоматично під час досягнення однієї з умов зупинки, описаної у вікні «Параметри навчання» або примусовим натисканням кнопки «Пауза». Користувач може примусово припинити навчання, якщо вважає подальше навчання непотрібним. При необхідності користувач може продовжити навчання з того місця, де воно було зупинене, для чого досить натиснути кнопку – «Продовжити». Однак, якщо після припинення навчання користувачем зміни в параметрах мережі відбуватимуться, навчання почнеться спочатку.

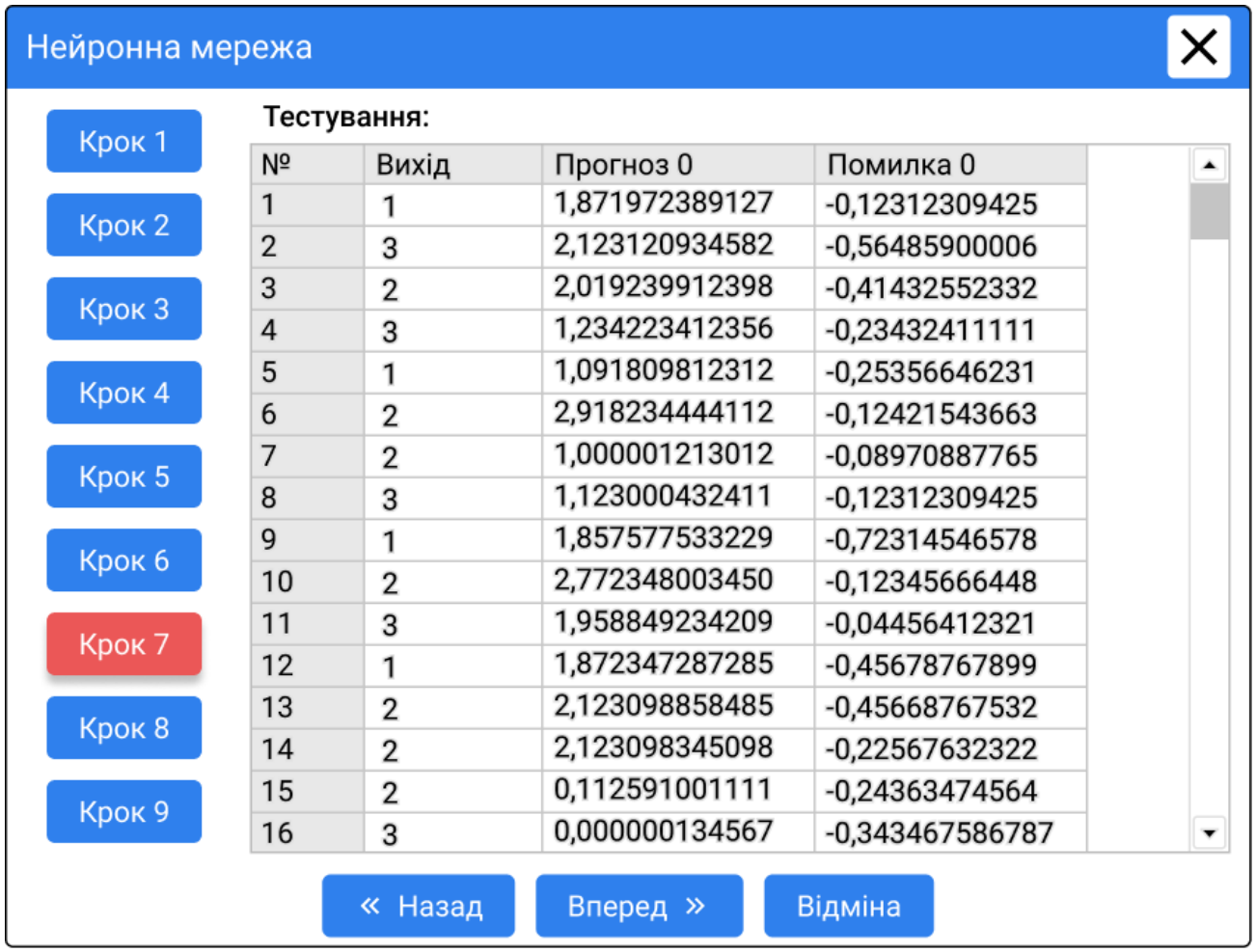


Рис. 4.7. Крок 7

На наступному етапі необхідно перевірити якість навчання нейронної мережі (рис. 4.7). У цьому вікні створюється таблиця, що містить фактичні вихідні значення та результат, отриманий з нейронної мережі. Наступний стовпець - це похибка, різниця між фактичним та очікуваним значеннями.

Якщо отримуємо незадовільний результат, можемо продовжити попередні кроки та оптимізувати свою систему.

Якщо якість результату прогнозування після відпрацювання нейронної мережі відповідає певним вимогам, можна перейти до наступного кроку - процедури спрощення нейронної мережі (рис. 4.8), яка перетворює вихідну мережу у форму, близьку до логічно прозорої. Під час натиснення кнопки «Синапс», «Нейрон», «Вхід», програма спробує видалити надмірну кількість синапсів, нейронів та входів. Зазвичай на цьому етапі якість мережі погіршується, що можна відновити шляхом подальшого навчання.

Можна переконатися в цьому, повернувшись до кроку 6 (рис. 4.6), натиснувши на відповідну кнопку. Після видалення елементів мережі, інформація про їх кількість відображатиметься під відповідною кнопкою.

Наступним етапом роботи є групування (кластеризація) значень виходів нейронів.

В програмі існує дві кнопки для кластеризвації нейронів – «Дискретизація» і «Виконати» (з піктограмою трикутника). Натискання кнопки «Дискретизація» зчитує та поточний стан нейронних виходів та проводить їх дискретизацію. Натискання кнопки «Виконати» не перечитує повторно стан нейронної активності. Попередньо завантажений набір станів дискретизується. Це може бути використано для прискорення роботи програми, коли вже зчитано кілька станів і користувач змінює лише параметр E (на 0,6). Якщо параметр E був встановлений занадто високим, а отримана кількість кластерів не дозволяє нейронній мережі набувати навичок, цей параметр слід зменшити (E може бути будь-яким значенням від 0 до 1).

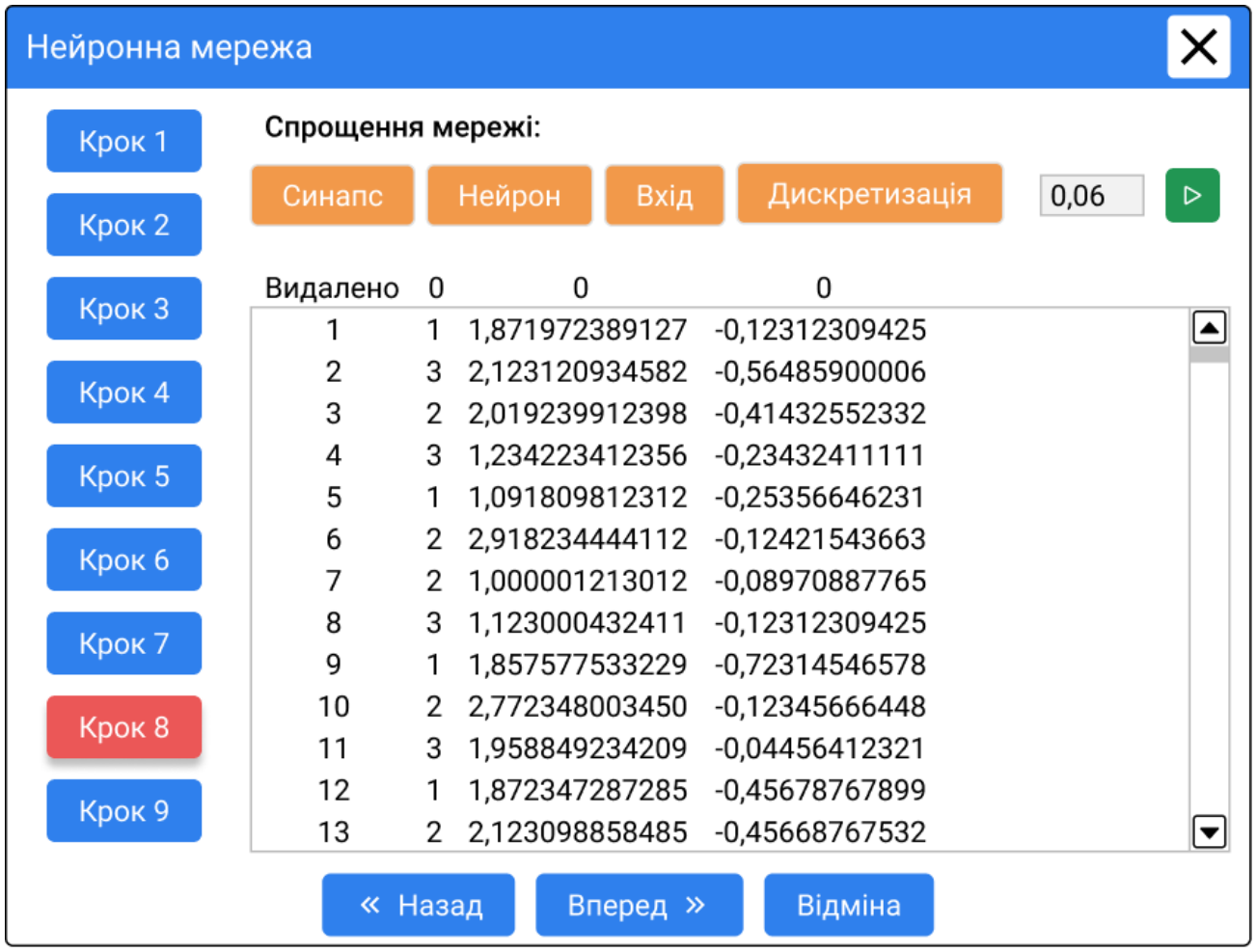


Рис. 4.8. Крок 8

Текстове поле внизу вікна відображає інформацію про хід процесу дискретизації під час перевірки стану мережі. Інформацію з цього поля можна скопіювати для подальшого аналізу після завершення програми.

Після виконання процедури кластеризації виходів нейронів та можливого повторного навчання нейронної мережі, отримуємо мережу, придатну для вилучення знань у явній формі.

Кількість правил, які опрацьовує задача, залежить від кількості кластерів, на які поділяються вхідні дані. Отже, наступним етапом є визначення оптимальної кількості кластерів, що забезпечить отримання високонадійних правил з мінімально можливою їх кількістю. Для цього кількість кластерів, на які

поділяються вхідні параметри, і кількість кластерів активних нейронів регулюється параметром E (на рис. 4.8 це 0,6). Після встановлення цього параметра, натисненням кнопки «1» заповнюється верхня таблиця центрами знайдених кластерів. Для кожного входу створюється окремий набір кластерів. Тому, кількість стовпців у таблиці відповідає кількості входів з нейронної мережі. Якщо у користувача є власні припущення щодо кластеризації вхідних параметрів на основі апріорних даних, можна змінити набори кластерів в таблиці вище і програма продовжить працювати з визначеним користувачем набором дискретних вхідних значень.

Натискання кнопки «2» заповнить ліву нижню таблицю всіма можливими комбінаціями кластерів з верхньої таблиці. Таким чином отримується набір правил, що характеризують досліджувану задачу якомога повніше. Однак ці правила є лише припущеннями, висновки ще не остаточні. Кнопка «3» використовується для заповнення наслідків (обчислення виходів нейронної мережі) для кожного раніше сформульованого правила. Отже, таблиця в нижньому лівому куті вікна програми містить кількість стовпців, що відповідає сумі кількості входів та кількості виходів мережі. Кнопка «4» призначена для додавання створених правил у файл.

В результаті кластеризації входів та побудови на їх основі правил, можуть виникнути ситуації, в яких не буде прикладів із навчального набору для сформованого правила. Можливі ситуації, коли не всі приклади правильно вирішені для певного правила. Щоб побачити, як поводяться правила для вхідного файлу, необхідно вибрати правило та натиснути кнопку «5», при цьому таблиця в нижньому правому куті заповнюється всіма прикладами в наборі, які відповідають правилу. Номер прикладу та значення його вихідних даних вставляються в таблицю праворуч (того ж ефекту можна досягти, використовуючи курсор для прокрутки записів у таблиці зліва). Натиснення

кнопки «6» заповнить відповідну таблицю кожного правила всіма прикладами, що відповідають їм. Це корисно для обчислення повної помилки для всіх правил.

Код структури мережі наведено у додатку Б.

Ця програма може бути використана як посібник для студентів усіх технічних дисциплін, які беруть участь у дослідженнях, розробці та використанні інструментів для інтелектуального аналізу даних. Програма була розроблена з використанням середовища розробки IntelliJ IDEA на мовах програмування Erlang (додаток Б), C ++. Для проектування структури програми був використаний функціональний підхід [130].

4.3. Перевірка ефективності системи за допомогою порівняння систем інтелектуального аналізу даних з точки зору пошуку логічних закономірностей

Спробуємо вирішити, здавалося б, тривіальну задачу пошуку логічних закономірностей у таблиці, що складається з 0 та 1 (табл. 4.1), використовуючи різні аналітичні методи. Таблиця розділена на дві половини на класи 1 і 0. Вона складається із 100 прикладів та двох кількісних показників X (горизонтально і праворуч від 0) та Y (вертикально і вниз від 0). Очевидно, що є 4 правила вирішення задачі:

- Якщо $X <= 5$ і $Y <= 5$, то 1.
- Якщо $X > 5$ і $Y > 5$, то 1.
- Якщо $X <= 5$ і $Y > 5$, то 0.
- Якщо $X > 5$ і $Y <= 5$, то 0.

Таблиця 4.1

Тестова задача

0	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	1	1	1	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1
7	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	1	1	1	1	1

Розглянемо можливості різних аналітичних систем та методів, які вони використовують для вирішення задачі.

WIZRULE [119]

WIZRULE не розпізнав жодного правила із налаштуваннями за замовчуванням або мінімально можливими вимогами до якості правила.

WIZRULE REPORT
GENERAL DETAILS:
File Name: D:\Samples\01problem.dbf
Total No. of Records: 100
All Rules
Minimum Probability of If-then Rules: 0,60
Minimum Accuracy Level of Formula Rules: 0,90
Minimum Number of Cases in a Rule: 3
WizRule User Licensee:
WizRule Version 4.01 Demo

See5 [118]

Система See5 також не розпізнала жодного правила ні з налаштуваннями за замовчуванням, ні з мінімально можливими вимогами до якості правила.

```
Sees [Release 1.17] Tue Jan 12 20:40:20 2021
------------
Class specified by attribute 'result'
Read 100 cases (3 attributes) from 01problem.data
Decision tree: 0 (100/50)
Evaluation on training data (100 cases):
        Decision Tree
        ------------
        Size Errors
         1   49 (49.5%) <<
        (a) (b) <-classified as
        ---- ----
        50     (a): class 0
        50     (b): class 1
Time: 0.1 secs
```

Cubist [118]

Cubist заснована на методі найближчого сусіда, не знайшла рішень. Можливо, вибравши найкраще налаштування, можна буде отримати результат. Як показано нижче (при розгляді PolyAnalyst), метод найближчого сусіда, як правило, може впоратися з цією задачею.

```
Cubist [Release 1.11] Tue Jan 12 23:58:56 2021
------------
Options:
        Allow use of instances with rules
        Use 1 nearest neighbors
        Cross-validate (10 folds)
        Use 1% of data for training
        Each rule must cover >=100% of cases
        Permit extrapolation of <=100%
   Target attribute 'result'
Read 1 cases (3 attributes) from Olproblem.data
*** folds reduced to number of cases
[ Fold 1 ]
Model:
Recommend using rules and instances
Evaluation on hold-out data (1 cases):
   Mean|error| 1.$
Summary:
   Average |error| 1.$
   Relative |error| 0.00
   Correlation coefficient -1.#J
Time: -0.0 secs
```

PolyAnalyst 4.5 [117]

PolyAnalyst 4.5 включає в свій арсенал кілька аналітичних методів: статистичні методи та створення дерев прийняття рішень. Основним елементом цієї аналітичної системи є PolyNet predictor, який використовує метод еволюційного програмування.

Слід відзначити наявну в системі PolyAnalyst 4.5 можливість побудови графічного представлення даних, як у вигляді двовимірних графіків (що в тому чи іншому вигляді присутній у багатьох системах аналізу даних), так і тривимірному (що в інших системах зустрічається значно рідше).

Модуль SS

Модуль SS зведеної статистики відображає загальну статистичну інформацію (ця функція доступна майже у всіх розглянутих системах, відображені статистичні характеристики є стандартними скрізь, тому ця функція не враховується в інших системах).

Таблиця містить атрибути та 101 запис даних.

Таблиця 4.2

Статистична інформація

Numerical and integer attributes:	Values	Mean	Std.Dev	Min	Max	Range	Median	Mode	Diff. values
"01problem_1"	100	5.5	2.887	1	10	9	6		
"01problem_2"	100	5.51	2.887	1	10	9	6		
"01problem_3"	100	0.5	0.5025	0	1	1	l		

Модуль LS

LS - лінійна регресія - цей статистичний підхід розпізнав залежність:

01problem_3 = +0.500000

І як правило, і від того, що R-squared (коефіцієнт детермінації), рівний 0, видно, що 0% початкових значень можна пояснити, а 100% решти значень залишається незрозумілою, тобто це правило не має сенсу.

Таблиця 4.3

Оцінка якості знайденої залежності

Стандартна похибка	1
R-squared	0
Стандартне відхилення	0,5025
Оброблено точок	100
Індекс значущості	-0,8808
Вільний член	0,5
Стандартне відхилення вільного члена	0,05025
F-Ratio вільного члена	99

Модуль FD

FD - знайдено залежності для 36 із 100 можливих записів.

97% точок даних знаходяться в діапазоні похибок +-0,500000. Знайдена залежність (табл. 4.5) наведена нижче.

Таблиця 4.4

Визначення якості виявленої залежності

Найвпливовішими параметрами є	"01problem_1" "01problem_2"
P-value	5,516e-006
Кількість точок, що задовольняють залежності	36
Стандартне відхилення (загальне)	0,4000000 (80,00%)
Стандартне відхилення (для популяції точок в кластерах)	0,000000 (0,00%)

Таблиця 4.5

Знайдена залежність

01problem_2/01problem_1	(-,4)	[4,8)	[8,+)
(-,4) pred.value points(cell) std.err(cell) points(subpop) std.err(subpop)	1 9 0 9 0	0,5 12 100 0 --	0 9 0 9 0
[4,8) pred.value points(cell) std.err(cell) points(subpop) std.err(subpop)	0,5 12 100 0 --	0,5 16 100 0 --	0,5 12 100 0 --
[8,+) pred.value points(cell) std.err(cell) points(subpop) std.err(subpop)	0 9 0 9 0	0,5 12 100 0 --	1 9 0 9 0

Модуль FL

FL - пошук законів, за 23 хвилини знайшов правила, якість яких наведено в таблиці (табл. 4.6).

Принцип найкращого правила за значимістю:

01problem_3 = (0.32655 * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") - 0.0775422 * "01problem_2" * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") - 0.0731539 * "01problem_2" + 0.969564) / (if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.26341 / "01problem_2"))

Принцип найкращого правила за точністю:

01problem_3 = (1.63895 * "01problem_2" * "01problem_2" * if(-0.254181 <= 1 * "01problem _1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.34572 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.34572 / "01problem_2") - 0.168777 * "01problem_2" * "01problem_2" * "01problem_2" * if(-0.254181 <= 1 * "01problem_1" and 1 *"01problem_1" < -0.254181 + 5.19708,1,5.34572 / "01problem_2") * if(-0.254181 <= 1 *"01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.34572 /"01problem_2") - 0.0170399 * "01problem_2" * "01problem_2" * "01problem_2" +0.368701 * "01problem_2" * "01problem_2" - 29.6449) / ("01probIem_2" *"01problem_2" * if (-0.254181 <= 1 *"01problem_1" and 1 *"01problem_1" < -0.254181 + 5.19708,1,5.34572 / "01problem_2") * if(-0.254181 <= 1 *"01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708, 1, 5.34572 / "01problem_2") * if(-0.254181 <= 1 * "01problem_1" and 1 * "01problem_1" < -0.254181 + 5.19708,1,5.34572 / "01problem_2") - 27.7469)

Таблиця 4.6

Оцінка якості знайденої залежності

Рівень	Стандартна похибка	Стандартне відхилення	Значущість	R-squared
Найбільше значення	0,6898	0,3449	8,942	0,5242
Найбільша точність	0,6241	0,312	0,648	0,6106

Модуль CL

На жаль, класифікація CL, заснована на лінійній регресії, розпізнала одне правило, яке виявилось неуспішним: якщо значення предиктора більше 1, тоді значення 01problem_3 має значення TRUE (клас 1), інакше - FALSE (клас 0).

Передбачений вираз:

$$1 < (+0{,}500000)$$

У наступній таблиці наведено оцінку отриманого правила (табл. 4.7).

Таблиця 4.7

Оцінка якості виявленої залежності

Помилка класифікації для класу 0:	0%
Помилка класифікації для класу 1:	100%
Загальна помилка класифікації:	50%
% невизначених прогнозів:	0%
Імовірність правильної класифікації:	-1%
Ефективність класифікації:	0%
P-value:	9,887e-030

Модуль FS

FS - правил кластеризації не знайдено.

Модуль DT

DT – дерево прийняття рішень не було створено для конкретних параметрів (табл. 4.8), як це відображено в таблицях (табл. 4.9).

Таблиця 4.8

Параметри дерева

Аргумент	Значення
Глибина дерева	1000
Мінімально дозволена «підтримка» вузла	0
Границя розділення вузла	6

Таблиця 4.9

Оцінка якості дерева

Помилка класифікації для класу Ні:	N/A
Помилка класифікації для класу Так:	N/A
Загальна помилка класифікації:	N/A
% невизначених прогнозів:	100%
Імовірність правильної класифікації:	N/A
Ефективність класифікації:	-100%

Модуль MB

MB - метод найближчого сусіда виявив найуспішнішу залежність із розлянутих методів (табл. 4.10, табл. 4.11).

Таблиця 4.10

Оцінка якості знайденої залежності

Стандартна помилка	0,368179
R-squared	0,864444
Стандартне відхилення	0,185017
Індекс значущості	0,000000
Значення наближення (neighborhood size)	3

Вибрані незалежні атрибути і відповідні їм фактори відстані (табл. 4.11).

Таблиця 4.11

Фактори відстані незалежних атрибутів

01problem_1	0,242487
01problem_2	0,242487

Deductor [116]

Система Deductor 5.3 включає різні технології аналізу даних: RawData Analyzer, Cube Analyzer, SOMap Analyzer, Neural Analyzer, Tree Analyzer.

Підсистема RawData Analyzer використовується для підготовки даних перед безпосереднім аналізом і оснащена функціональним інструментом для створення різних діаграм. Cube Analyzer - реалізує механізм технології OLAP (комплексний багатовимірний аналіз даних). Основною задачею системи Cube Analyzer є організація подання вибраних користувачем даних у формі, що відповідає сприйняттю, без обробки даних. Ці підсистеми не призначені для вирішення поставленої задачі і тому не розглядаються.

SOMap Analyzer

SOMap Analyzer - це підсистема аналізу даних, заснована на самоорганізованих картах Кохонена.

Потужний інструмент, що поєднує дві основні парадигми аналізу - кластеризацію та проектування, не зміг скласти карту, яка адекватно виконує роботу (табл. 4.12).

Neural Analyzer

Neural Analyzer - це програмний емулятор нейронної мережі з вибором різних навчальних парадигм. Такий підхід дозволив навчити мережу вирішувати поставлену задачу. Нижче наведені параметри навчання та налаштована мережа.

Таблиця 4.12

Звіт - SOMap Analyzer

Copyright (C) BaseGroup Labs 2020	
Набір даних	Навчальна вибірка
Карта по полю	Clusters
Мінімальне	0
Максимальне	1

Вхідні поля:
COL1 (F1)
норм.: лінійна
макс: 1,00E+001
мін.: 1,00E+000
серед.: 5,55E+000
дисп.: 2,87E+000
COL2 (F2)
норм.: лінійна
макс: 1,00E+001
мін.: 1,00E+000
серед.: 5,55E+000
дисп.: 2,87E+000
Цільові поля:
COL3 (F3)
норм.: без нормалізації
макс: 1,00E+000
мін.: 0,00E+000
серед.: 4,95E-001
дисп.: 5,03E-001
Завантаженість даних:
Всього в наборі даних 100 прикладів
Навчальна множина: 80 прикладів (80%)
Тестова множина: 20 прикладів (20%)
Заповнювати у випадковому порядку
Параметри навчання:
Алгоритм навчання
Back - Propagation
Функція активації:
Сигмоїда
Крутизна сигмоїди: 1,00
Зміщення: 0,00
Умови зупинки навчання:
Вважати розпізнаними приклади, якщо кв.пох. <5,00E-002
Минуло епох: 10000
Конфігурація мережі:
вхід: 0 нейронів: 2
шар: 1 нейронів: 8
вихід: 2 нейронів: 1
ваги:
шар: 1
нейрон: 0
W_0 = -5,29745118805383E+000
W_l = 4,65917918059383E+000
WT_0 = -2,36036082248946E+000
нейрон:1
W_0 = -1,16444181228178E+001
W_l = -5,33494677095488E+000

WT_1 = -3,23570563436394E+000
нейрон:2
W_0 = -3,63829156132655E+000
W_l = 3,57166607676893E+000
WT_2 = -2,82602376659606E+000
нейрон:3
W_0 = -4,90258051522314E+000
W_l = 6,00670703237645E+000
WT_3 = 2,80531950450266E+000
нейрон: 4
W_0 = -5,32136861341713E+000
W_l = -4,52543196781970E+000
WT_4 = -4,53471610122423E+000
нейрон:5
W_0 = 4,08521127491149E+000
W_l = 6,59213322788813E+000
WT_5 = -2,20939773021993E+000
нейрон:6
W_0 = -4,8687099406325 lE+000
W_l = 6,10140368808584E+000
WT_6 = 3,01580959736779E+000
нейрон:7
W_0 = -5,32739400308083E+000
W_l = -4,52901863561206E+000
WT_7 = -4,54201484866647E+000
шар: вихідний
нейрон: 0
W_0 = -1,86060482720134E+001
W_l = 1,16674402268543E+001
W_2 = -6,11904657267188E+000
W_3 = 8,03637061835150E+000
W_4 = 7,76623889330840E+000
W_5 = 1,72502793035770E+001
W_6 = 8,45995425869634E+000
W_7 = 7,78821252631481E+000
WT_0 = -l,68216337900967E+001

Парадигма нейронної мережі обробляє дані з певним ступенем точності. Однак отримання самих правил, які могли б виявити взаємозв'язок даних для аналітика, виходить за можливості цієї аналітичної системи.

Tree Analyzer

Tree Analyzer - це система аналізу даних, заснована на деревах прийняття рішень. Автори не заявляють, який метод є основою, і в алгоритмі дерева прийняття рішень немає згадки про власні уточнення. Однак з трьох реалізацій,

обговорених у цій главі, лише ця аналітична система змогла розпізнати 4 правила та охопити 54 приклади зі 100. Знайдені правила:

1) ЯКЩО COL1> 1,5 І
   COL1 <= 5,5,
   ТО COL3 = 0
   Кількість записів у правилі: 20
   Вірні приклади: 20
   Невірні приклади: 0
2) ЯКЩО COL1 = 1, 5 І
   COL2> 5,5
   ТО COL3 = 0
   Кількість записів у правилі: 5
   Вірні приклади: 5
   Невірні приклади: 0
3) ЯКЩО COL1> 1,5 І
   COL1> 5,5
   ТО COL3 = 1
   Кількість записів у правилі: 25
   Вірні приклади: 25
   Невірні приклади: 0
4) ЯКЩО COL1 <= 1,5 І
   COL2 <= 5,5
   ТО COL3 = 1
   Кількість записів у правилі: 4
   Вірні приклади: 4
   Невірні приклади: 0

Алгоритми побудови дерев прийняття рішень відповідають принципу поступового пошуку «найкращих» функцій, створюючи тим самим ілюзію індуктивного логічного висновку. У деяких випадках вони не можуть вирішити задачу і зазвичай знаходять у даних лише частини реальних логічних закономірностей. Характеристики X та Y вирішеної задачі, які розглядаються окремо, не дозволяють відокремлювати 1 від 0. Таким чином, алгоритми дерева прийняття рішень зазвичай відмовляються знаходити правило логіки на першому кроці визначення «найкращої» функції. На одному з кроків Tree Analyzer зміг частково обійти цю проблему та отримати найкращий результат від систем, що реалізують алгоритм дерева прийняття рішень.

NeuroPro 0,25 [80]

Цей програмний продукт є менеджером штучних нейронних мереж, які можна навчити. Ця програма цікава тим, що можна спростити отриману мережу, спочатку виділивши надмірну кількість нейронів. Незважаючи на те, що система вважається застарілою, лише в цій системі реалізовано спробу отримувати інформацію від навченої мережі у формі, зрозумілій досліднику. На жаль, це відбувається лише на рівні вербалізації мережі. Система не пропонує чітких правил. Математичний апарат NeuroPo вирішив цю задачу за допомогою мережі, що складається з двох прихованих шарів, кожен з п'ятьма нейронами. Нижче представлена вербалізація нейронної мережі:

Синдроми рівня 1:

Синдром1_1 = Сигмоїда1(0,7317115 * COL1 - 0,1982859 * COL2 + 0,04892339)
Синдром1_2 = Сигмоїда1(0,9706837 * COL1 - 0,0558333 * COL2 - 0,0008786368)
Синдром1_3 = Сигмоїда1(-0,09648797 * COL1 + 0,8516223 * COL2 + 0,02282562)
Синдром1_4=Сигмоїда1(-0,04130807 * COL1 + 0,899415 * COL2 -0,0283553)
Синдром1_5 = Сигмоїда1(-0,3716678 * COL1 -0,4985738 * COL2 + 0,6272115)

Синдроми рівня 2:

Синдром2_1 = Сигмоїда2(0,2403359 * Синдром1_1 + 0,2077693 * Синдром1_2 + 0,03431265 * Синдром1_3 - 0,2961471 * Синдром1_4 + 0,0688605 * Синдром1_5 + 0,38506)

Синдром2_2 = Сигмоїда2(-0,09554291 * Синдром1_1 + 0,1132906 * Синдром1_2 - 0,04713675 * Синдром1_3 - 0,1630154 * Синдром1_4 + 0,890016 * Синдром1_5 - 0,3533287)

Синдром2_3=Сигмоїда2(-0,3416102 * Синдром1_1 - 0,5418347 * Синдром1_2 - Синдром1_3 -0,1427334 * Синдром1_4 + 0,415338 * Синдром1_5 + 0,2964585)

Синдром2_4 = Сигмоїда2(0,2694791 * Синдром1_1 + 0,9937754 * Синдром1_2 + 0,3163903 * Синдром1_3 + 0,7220843 * Синдром 1_4 + 0,2753767 * Синдром1_5 + 0,4003687)

Синдром2_5 = Сигмоїда2(0,1334755 * Синдром1_1 + 0,258269 * Синдром1_2 - 0,2513632 * Синдром1_3 -0,1543842 * Синдром1_4 + 0,4768991 * Синдром1_5 + 0,4141049)

Кінцеві синдроми:

COL3 = 0,185475 * Синдром2_1 - 0,2077792 * Синдром2_2 - Синдром2_3 - Синдром2_4 + 0,5913992 * Синдром2_5 + 0,4071461

Розроблена інтелектуальна система аналізу даних на основі нейронної мережі

На основі підходів, запропонованих у попередньому розділі, була розроблена система аналізу даних. Перевіримо запропоновані підходи до вирішення задачі.

Для вирішення задачі було достатньо нейронної мережі, що складається лише з 8 нейронів в одному прихованому шарі. Вхід складався з двох нейронів, вихід - одного. Мережа навчена розпізнавати всі приклади, передбачені для навчання. Мережа була повністю навчена за 2000 кроків.

На цьому етапі процес спрощення можна пропустити, оскільки кількість нейронів була обрана малою, а структура всієї мережі наближається до логічно чіткої форми. Тепер мережу можна вербалізувати (табл. 4.13).

Таблиця 4.13

Вербалізація нейронних мереж

Формули прихованого шару 1	
Нейрон1_1	3,21114873308467 * X + 3,21387068057069 * У - 2,96353354858871
Нейрон1_2	4,13711469793585 * X + 4,14087884735904 * У -3,95376784810039
Нейрон1_3	6,29485083567893 * X + 6,29623271511132 * У -1,28826080957467
Нейрон1_4	3,84076566487926 * X + 3,84371197429532 * У - 3,63829743989262
Нейрон1_5	-5,47457339415 * X + 5,31945580498692 * У - 2,97618641493273
Нейрон1_6	0,943780697326842 * X + 0,153162410885338 * У - 1,70375735731247
Нейрон1_7	5,31878353963628 * X - 5,47774722800497 * У - 2,97951069019882
Нейрон1_8	9,75894977144238 * X + 9,75303987839576 * У + 2,02497269234449
Формули вихідного шару	
CLASS	4,66912169885151 * Нейрон1_1 + 6,54337646683553 * Нейрон1_2 + 16,7569645673660963 * Нейрон1_3 + 5,94896266815723 * Нейрон1_4 - 22,8839375572877 * Нейрон1_5 + 0,251849040357322 * Нейрон1_6 - 22,8895873035487 * Нейрон1_7 - 16,7438947685909 * Нейрон1_8 + 14,643201250775

У порівнянні з представленим методом вербалізації, метод отримання правил у вигляді «якщо … то …», дозволить отримувати більш логічно зрозумілі для аналітика закономірності.

Як показано на рис. 4.9, для даного параметра вибірки E = 0,9 набір вхідних даних для кожного входу був розділений на дві частини (табл. 4.14).

Отже, ми маємо 4 різні комбінації, які складають таблицю правил, що описують, як працює задача (табл. 4.15).

Таблиця 4.14

Кластеризація вхідних даних

Вхідні параметри	Центри кластера			
	Кластер 1	Інтервал 1	Кластер 2	Інтервал 2
X	-0,1	$(-\infty; 5)$	-0,47777779	$[5; +\infty)$
Y	0, 1111111111	$(-\infty; 5)$	0,333333343	$[5; +\infty)$

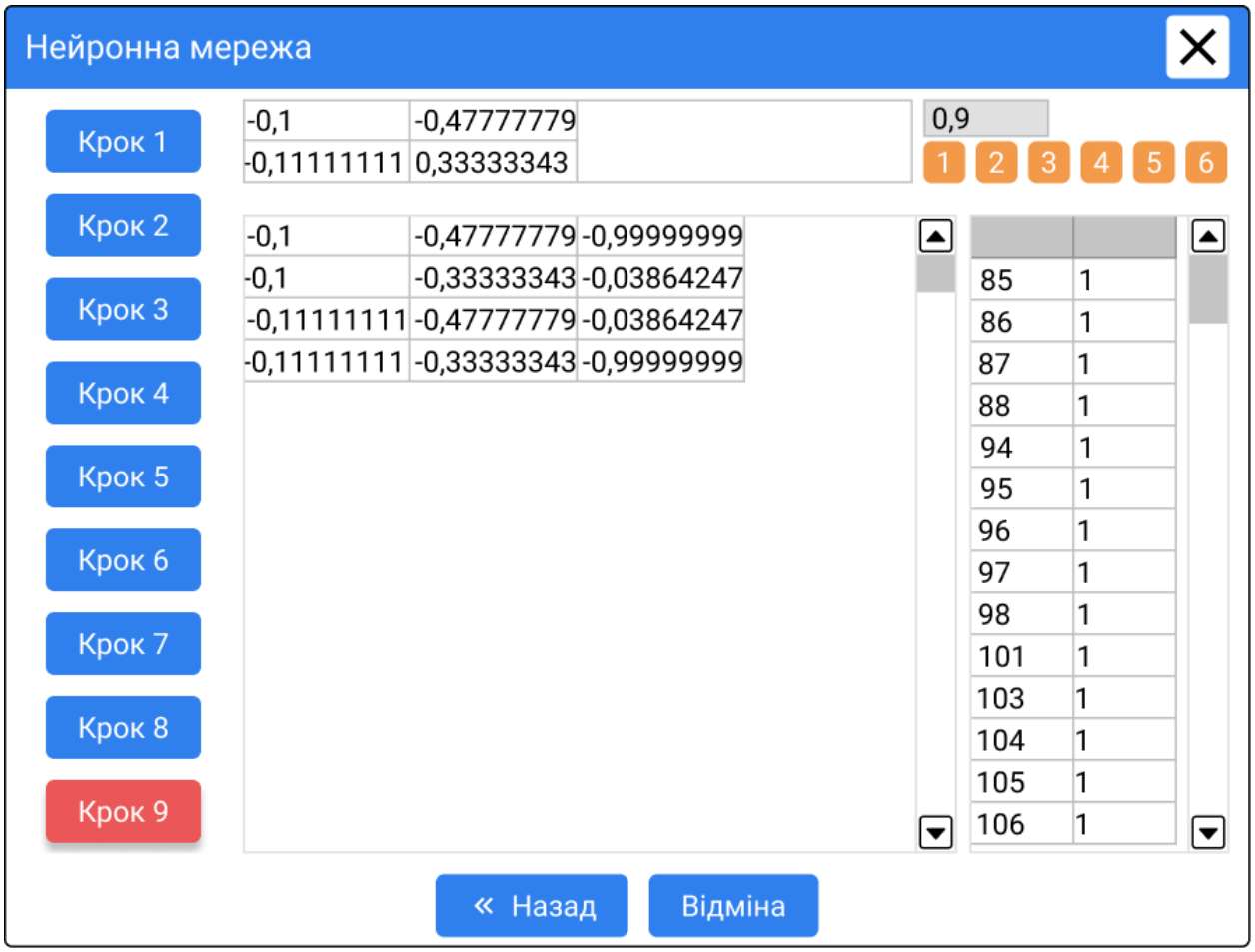


Рис. 4.9. Видобування правил

Таблиця 4.15

Таблиця правил функціонування мережі, де пр – клас із навчальної вибірики, с – клас, передбачений мережею, вірно – кількість навчальних прикладів, що підтверджують вказане правило, кнп – кількість навчальних прикладів, що невірно класифіковані вказаним правилом

№	X		Y		клас			вірно	кнп
					пр		с		
А	ЯКЩО	кластер 1	І	кластер 1	ТО	0	0	24	0
Б	ЯКЩО	кластер 2	І	кластер 2	ТО	1	1	25	0
В	ЯКЩО	кластер 3	І	кластер 1	ТО	1	1	25	0
Г	ЯКЩО	кластер 4	І	кластер 2	ТО	0	0	25	0

Якщо ми замінимо значення відповідних діапазонів замість «кластер» в отриманих правилах (табл. 4.15), ми отримаємо правила, які ми шукали під час постановки задачі:

1) ЯКЩО X в $(-\infty;\ 5)$ І Y в $(-\infty;\ 5)$, ТО 0.
2) ЯКЩО X в $(-\infty;\ 5)$ І Y в $[5;\ +\infty)$, ТО 1.
3) ЯКЩО X в $[5;\ +\infty)$ І Y в $(-\infty;\ 5)$, ТО 1.
4) ЯКЩО X в $[5;\ +\infty)$ І Y в $[5;\ +\infty)$, ТО 0.

Таким чином, застосування запропонованого підходу дозволило: а) навчити нейронну мережу з точки зору 100% розподілу даних на два класи; б) створити повний набір правил, який без винятку пояснює всі наявні приклади, щоб задача працювала у зрозумілій для користувача формі.

Яскравість кожного методу, що використовується для вирішення задачі, можна оцінити з табл. 4.16, яка не містить методів, що не виявляли залежностей, і методів, які неможливо інтерпретувати, а також у табл. 4.17, що систематизує показники ефективності розглянутих систем аналізу даних.

Таблиця 4.16

Результати розв’язання задачі за допомогою різних аналітичних систем

<table>
<tr><td>Дерева прийняття рішень (модифіковані)</td><td colspan="6">PolyAnalyst, модуль FL - пошук законів</td></tr>
<tr><td>1)ЯКЩО C0L1>1,5 І C0L1<=5,5 ТО COL3=0 2)ЯКЩО C0L1<=1,5 І COL2>5,5 ТО COL3=0
3) ЯКЩО C0L1>1,5 І COL1>5,5 ТО COL3=1
4) ЯКЩО C0L1<=1,5 І COL2<=5,5 ТО C0L3=1</td><td colspan="6">COL3=(1.63895*COL2*COL2*if(-0.254181 <= 1 *COL1 and 1*COL1 <-0.254181 + 5.19708 ,1 ,5.34572 /COL2)*if(-0.254181<= 1*COL1 and COL1 <-0.254181+5.19708 ,1 ,5.34572 /COL2)-0.168777*COL2*COL2*COL2*if(-0.254181<= 1*COL1 and COL1<-0.254181+5.19708,1 ,5.34572 /COL2)*if(- 0.254181 <= 1*COL1 and COL1<-0.254181+5.19708 ,1 ,5.34572 /COL2)-0.0170399*COL2*COL2*COL2+0.368701*COL2*COL2-29.6449)/ (COL2*COL2* if(-0.254181 <= 1 *C0L1 and 1*COL1<-0.254181+5.19708,1 ,5.34572 /COL2)*if(- 0.254181 <= 1*COL1 and COL1<-0.254181+5.19708,1,5.34572/COL2)*if(- 0.254181 <= I *COL1 and 1*COL1<-0.254181 + 5.19708 ,1 ,5.34572/COL2)-27.74)</td></tr>
<tr><td>Нейронні мережі (Deductor та NeuroPro)</td><td colspan="6">Розроблена система аналізу даних на основі нейронних мереж</td></tr>
<tr><td rowspan="6">Нейрони рівня 1:
H1_1=0,7317115*COL1-0,1982859*COL2+0,04892339...
H1_5=-0,3716678*COL1-0,4985738*COL2+0,6272115
Нейрони рівня 2:
H2_1=O,2403359*H1_1+0,2077693*H1_2+0,03431265...
H2_5=O,1334755*H1_1+O,258269*H1_2-0,2513632*H1_3-...
Вихідні нейрони:
COL3=0,185</td><td></td><td>COL1</td><td></td><td>COL2</td><td></td><td>Результат</td></tr>
<tr><td>ЯКЩО</td><td>(−∞; 5)</td><td>І</td><td>(−∞; 5)</td><td>ТО</td><td>0</td></tr>
<tr><td>ЯКЩО</td><td>(−∞; 5)</td><td>І</td><td>[5; +∞)</td><td>ТО</td><td>1</td></tr>
<tr><td>ЯКЩО</td><td>[5; +∞)</td><td>І</td><td>(−∞; 5)</td><td>ТО</td><td>1</td></tr>
<tr><td>ЯКЩО</td><td>[5; +∞)</td><td>І</td><td>[5; +∞)</td><td>ТО</td><td>0</td></tr>
</table>

Таблиця 4.17

Систематизація показників ефективності розглянутих систем аналізу даних

	Закономірності розпізнано	Кількість розпізнаних значень	Стандартна похибка	Стандартне відхилення	Кількість епох для успішного навчання (для нейронних мереж)
WIZRULE	-	-	-	-	-
See5	-	-	49,5%	-	-
Cubist	-	-	-	-	-
PolyAnalyst 4.5					
Модуль SS	-	-	-	-	-
Модуль LS	+	-	-	0,5025	-
Модуль FD	+	36	17%	0,80	-
Модуль FL	+	-	68,98%	0,3449	-
Модуль CL	+	-	50%	-	
Модуль FS	-	-	-	-	-
Модуль DT	-	-	-	-	-
Модуль MB	+	-	36,82%	0,185	-
Deductor					
SOMap Analyzer	-	-	-	-	-
Neural Analyzer	-	-	55,5%	0,5357	10000
Tree Analyzer	+	54	39%	0,483	-
NeuroPro 0,25	+	86	29,3%	0,265	7000
Розроблена система	+	98	13%	0,122	2000

Отже, запропонована задача не може бути вирішена з достатньою точністю жодною системою, яка використовує підхід, відмінний від даної нейронної мережі, як математичного інструменту. Серед аналітичних систем, заснованих на підході нейронних мереж, аналітична система, розроблена в цій роботі, показала один з найкращих результатів з точки зору здатності навчатися при мінімальній кількості нейронів. Більше того, це єдина система у своєму класі, яка представляє шаблон, який нейронна мережа знаходить у форматі «якщо … то …», найбільш зрозумілому для аналітиків. Звичайно, представлений тест не говорить про перевагу одного підходу над іншим у всіх областях, але він чітко демонструє ефективність та результативність підходу, запропонованого для аналізу даних у попередніх розділах.

Висновки до розділу 4

У розглянутому розділі було розроблено з використанням програмних засобів інтелектуальну систему аналізу даних на основі технологій нейронних мереж.

Під час розробки було проаналізовано можливості сучасних мов програмування, які використовуються для моделювання нейронних мереж. Також обгрунтовано вибір мови програмування Erlang, вказано на переваги запропонованої мови порівняно з існуючими. Надано докладні інструкції з налаштування системи, детально та покроково описано функціональне призначення розробленого програмного забезпечення, надано чіткі інструкції для користувача. Здійснено технічний опис програмного та апаратного забезпечення, яке необхідне для можливості використання системи.

Перевірено ефективність розробленої системи за допомогою порівняння систем інтелектуального аналізу даних з точки зору пошуку логічних закономірностей. Для порівняння було використано аналітичну систему

WIZRULE, See5, Cubist, PolyAnalyst 4.5 з модулями SS, LS, FD, FL, CL, FS, DT, MB, систему Deductor 5.3 з технологіями аналізу даних RawData Analyzer, Cube Analyzer, SOMap Analyzer, Neural Analyzer, Tree Analyzer, систему NeuroPro 0,25 та розроблену.

Після перевірки було отримано висновок, що запропонована задача не може бути вирішена з достатньою точністю жодною системою, яка використовує підхід, відмінний від даної нейронної мережі, як математичного інструменту. Серед аналітичних систем, заснованих на підході нейронних мереж, аналітична система, розроблена в цій роботі, показала один з найкращих результатів з точки зору здатності навчатися при мінімальній кількості нейронів. Більше того, це єдина система у своєму класі, яка представляє шаблон, який нейронна мережа знаходить у форматі «якщо … то …», найбільш зрозумілому для аналітиків.

Застосування запропонованого підходу дозволило: а) навчити нейронну мережу з точки зору 100% розподілу даних на два класи; б) створити повний набір правил, який без винятку пояснює всі наявні приклади, щоб задача працювала у зрозумілій для користувача формі.

Тому можна стверджувати, що створено весь необхідний програмний апарат, що був перевірений на ефективність на тестових та реальних задачах.

ВИСНОВКИ

В дисертаційній роботі вирішувалось наукове завдання з розробки методики побудови інтелектуальної системи аналізу даних на основі нейронних мереж.

Опишемо основні висновки та результати роботи.

Сформульовано вимоги до сучасних систем аналізу даних для обґрунтування вибору математичного апарату ядра аналітичної системи.

Розроблено новий метод нелінійної нормалізації даних, який полягає у поступовій реалізації перетворень над даними із змінним типом нелінійності, що дозволяє збільшити інформаційне наповнення вхідних даних.

Досліджено, які нейронні мережі мають перевагу над традиційними методами математичної статистики та технічного аналізу для визначеної задачі.

Обрано оптимальну топологію, що має структуру багатошарового персептрона, та оптимальну кількість параметрів обраної топології.

Розроблено новий підхід, який полягає у кластеризації вхідних параметрів та активності нейронів, що дозволяє отримати правила залежності даних від навченої нейронної мережі.

Розроблено інформаційну технологію інтелектуального аналізу даних за допомогою апарату нейронної мережі, що дозволяє здійснити перевірку ефективності розроблених алгоритмів.

Отже, дисертація містить вирішення задачі автоматизованого отримання знань з баз даних, отримані наукові результати мають принципове значення для теорії та практики розробки інтелектуальних систем для обробки аналітичної інформації.

Результати дисертиційних досліджень впроваджені у виробничий процес ТОВ «ХУАВЕЙ Україна», ТОВ «ПІПЛ УА», ТОВ «Ай Ті Джи» та в навчальному процесі Державного університету телекомунікацій (Київ) (додаток В).

Можливими напрямками подальших досліджень є удосконалення нейронної мережі, на основі якої працює інформаційна технологія інтелектуального аналізу даних.

ЛІТЕРАТУРА

1. S. Yan, Y. Wang, and J. Liu, “Research on the Comprehensive Evaluation of BusinessIntelligence System Based on BP Neural Network”, in *2nd International Conference on Complexity Science & Information Engineering*, Beijing, China, 2011, pp. 2211-3819.

2. X. Ni, “Research of Data Mining Based on Neural Networks”, *World Academy of Science*, vol. 39, pp. 381-384, 2008.

3. V.M. Kussul, and T.N. Baidyk, “Working out of architecture neural networks for recognition of the form of objects on the image”, *Automatics*, vol. 5, pp. 56-61, 2011.

4. E.H. Dzheffri, “As neural networks are trained”, *In the science world*, vol. 11-12, pp. 103-107, 2013.

5. Ph. Koehn, *Neural Machine Translation*. Cambridge, England: Cambridge University Press, 2020.

6. V.G. Manzhula, and D.S. Fedyashov, “Kohonen neural networks and fuzzy neural networks in data mining”, *Technical sciences*, vol. 4, pp. 108-114, 2013.

7. X.-P. Zhang, “Thresholding neural network for adaptive noise reduction”, in *Proc. IEEE Trans. on Neural Networks*, 2001, pp. 567–584.

8. A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: high confidence predictions for unrecognizable images”, in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 427–436.

9. C. Charu, *Aggarwal. Neural Networks and Deep Learning*. New York, USA: Springer, 2018.

10. U. Michelucci, *Applied Deep Learning a Case-Based Approach to Understanding Deep Neural Networks*. New York, USA: Springer, 2018.

11. V. Savchenko, V. Akhramovych, A. Tushych, I. Sribna, and I. Vlasov, “Analysis of Social Network Parameters and the Likelihood of its Construction”, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no 2, pp. 271-276, 2020.

12. K. Storchak, N. Yakovenko, A. Tushych, I. Sribnaya, and O. Polonevich, “Improving the material quality of network equipment due to a mechanism of surface hardening”, *Scientific discussion*, vol. 1, no. 49, pp. 34-39, 2020.

13. А.М. Тушич, К.П. Сторчак, А.П. Бондарчук, та А.О. Макаренко, “Вимоги до інтелектуальних систем аналізу даних та їх класифікацій”, *Науково-технічний журнал “Телекомунікаційні та інформаційні технології”,* №1, с. 31-36, 2019.

14. К.П. Сторчак, А.М. Тушич, та А.П. Бондарчук, “Кластерний аналіз даних з використанням штучних нейронних мереж”, *Науковий журнал “Зв’язок”,* №6, с. 36-38, 2018.

15. К.П. Сторчак, А.М. Тушич, К.С. Козелкова, та М.М. Степанов, “Інтелектуальний аналіз даних з використанням нейронних мереж”, *Науковий журнал “Зв’язок”,* №4, с. 17-19, 2018.

16. К.П. Сторчак, А.М. Тушич, О.М. Ткаленко, В.М. Чорна, та Т.М. Жила, “Аналіз вимог до проектування хмарної платформи для інтернету речей”, *Науковий журнал “Зв’язок”,* №6, с. 26-34, 2019.

17. О.А. Золотухіна, О.М. Ткаленко, А.М. Тушич, В.М. Чорна, та О.Р. Нікітенко, “Концепція розвитку підсистеми передавання мультимедійних повідомлень IMS”, *Науково-технічний журнал “Телекомунікаційні та інформаційні технології”*, №4, с. 81-89, 2019.

18. І.С. Сиротенко, І.С. Щербина, К.П. Сторчак, та А.М. Тушич, “Аналіз ефективності викристання нейронних мереж на прикладі багатошарового персептрону та мережі Кохонена”, *Науковий журнал “Зв’язок”,* №5, с. 17-19, 2020.

19. Б.В. Шефкін, І.В. Красюк, В.О. Хоменчук, К.П. Сторчак, та А.М. Тушич, “Дослідження та впровадження нейронної мережі на основі TENSORFLOW”, *Науковий журнал “Зв’язок”,* №6, с. 18-20, 2020.

20. К.П. Сторчак, Д.В. Кравець, А.М. Тушич, та Д.В. Сорокін, “Аналіз методів організації прав користувачів у GNU\Linux системах”, *Науковий журнал “Зв’язок”,* №4, с. 38-40, 2020.

21. А.М. Тушич, “Використання штучних нейронних мереж для створення IoT рішення фільтрації VPN трафіку”, на *XI Міжнар. наук.-техн. конф. студентів та аспірантів “Перспективи розвитку інформаційно-телекомунікаційних технологій та систем”*, Київ, 2019, с. 361.

22. А.М. Тушич, “Машинне навчання з використанням TENSORFLOW”, на *XII Міжнар. наук.-техн. конф. студентів та аспірантів “Перспективи розвитку інформаційно-телекомунікаційних технологій та систем”,* Київ, 2020, с. 379.

23. М. Пелепей, та А. Тушич, “Штучний інтелект – друг, ворог чи помічник людини”, на *IX Міжнар. наук.-техн. конф. студентства та молоді “Світ інформації та телекомунікацій”*, Київ, 2019, с. 303-304.

24. А.М. Тушич, “Аналіз доцільності використання автоматизованої системи інтелектуального аналізу даних на основі штучних нейронних мереж”, на *VII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених з автоматичного управління присвячена Дню космонавтики*, Херсон, 2019, с. 76-77.

25. Д.Я. Алтинніков, та А.М. Тушич, “Як штучний інтелект та IOT доповнюють один одного”, на *Всеукр. наук.-техн. конф. “Сучасний стан та перспективи розвитку IOT”*, Київ, 2020, с. 303-304.

26. А.М. Тушич, та П.В. Лебединець, “Дослідження системи моніторингу Zabbix для IT-інфраструктури підприємства”, на *VII Наук.-техн. конф. “Сучасні інфокомунікаційні технології”*, Київ, 2018, с. 20-21.

27. А.М. Тушич, та О.М. Скрипаль. “Безпека функціонування телекомунікаційніх систем та мереж”, на *XII Міжнар. наук.-техн. конф. “Проблеми інформатизації”*, Київ, 2018р. – С.75-76.

28. А.М. Тушич, та П.В. Лебединець, “Дослідження відкритого програмного забезпечення поштового сервера та клієнта”, на *XII Міжнар. наук.-техн. конф “Проблеми інформатизації”*, Київ, 2018, с. 105.

29. Б. Свердлюк, Ю.Каграманова, та А. Тушич, “Інтернет речей”, на *IX Міжнар. наук.-техн. конф. студентства та молоді “Світ інформації та телекомунікацій*”, Київ, 2019, с. 107-108.

30. Y. Zhang, and Z. Teng, *Natural Language processing. A Machine Learning Perspective*. Cambridge, England: Cambridge University Press, 2020.

31. M.J. Zaki, and M. Wagner, *Data Mining and Machine Learning*. *Fundamental Concepts and Algorithms*. Cambridge, England: Cambridge University Press, 2020.

32. L.N. Cooper, *Science and Human Experience*. Cambridge, England: Cambridge University Press, 2014.

33. Sh. Shalev-Shwartz, and Sh. Ben-David, *Understanding Machine Learning. From Theory to Algorithms*. Cambridge, England: Cambridge University Press, 2014.

34. S. Farrell, and St. Lewandowsky, *Computational Modeling of Cognition and Behavior*. Cambridge, England: Cambridge University Press, 2018.

35. K. Frankish, and W.M. Ramsey, *The Cambridge Handbook of Artificial Intelligence*. Cambridge, England: Cambridge University Press, 2014.

36. J. Wu, *Essentials of Pattern Recognition. An Accessible Approach*. Cambridge, England: Cambridge University Press, 2020.

37. T. Hey, and G. Papay, *The Computing Universe. A Journey through a Revolution*. Cambridge, England: Cambridge University Press, 2014.

38. J. Watt, R. Borhani, and A.K. Katsaggelos, *Machine Learning Refined*. Cambridge, England: Cambridge University Press, 2020.

39. D.A. Lieberman, *Learning and Memory*. Cambridge, England: Cambridge University Press, 2020.

40. Br. Efron, and Tr. Hastie, *Computer Age Statistical Inference*. Cambridge, England: Cambridge University Press, 2016.

41. W.A. Kretzschmar, *Language and Complex Systems*. Cambridge, England: Cambridge University Press, 2015.

42. D.A. Lieberman, *Learning and Memory*. Cambridge, England: Cambridge University Press, 2020.

43. S.L. Brunton, and J.N. Kutz, *Data-Driven Science and Engineering*. Cambridge, England: Cambridge University Press, 2019.

44. N. Kasabov, "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data", *Neural Networks*, vol. 52, pp. 62-76, 2014.

45. D.U. Tenk, and D.D. Khopfild, "Collective calculation in neural electronic schemes", *In the science world*, vol. 2, pp. 44-53, 2012.

46. S. Tsuprikov, "Neuron calculations undertake on arms of financiers", *Computerworld*, vol. 7, pp. 57-58, 2011.

47. A.I. Masalovich, "From neuron to neuron-computer", *Computers + programs*, vol. 1, pp. 20-23, 2012.

48. P. Kouveles, and H.L. Lee, "Block angular structures and the loading problem in flexible manufacturing systems", *Oper*, vol. 39, no. 4, pp. 666-676, 2011.

49. M. Christopher, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer science, 2012.

50. M.E. Rimer, T.L. Anderson, and T.R. Martinez, "Improving backpropagation ensembles through lazy training", in *Proc. IEEE Intern. Joint Conf. on Neural Networks*, 2001, pp. 2007–2112.

51. D. Ivanov, I. Kalyaev, and S. Kapustyan, “Method of circles for solving formation task in a group of quadrotor UAVs”, on *2nd International Conference Systems and Informatics (ICSAI),* 2014, pp. 236–240.

52. D. Ivanov, I. Kalyaev, and S. Kapustyan, “Formation task in a group of quadrotors”, in *The 3rd International Conference on Robot Intelligence Technology and Applications*, 2014, pp. 183–191.

53. R. Kostadinova, and C. Adam, “Performance Analysis of the Epidemic Algorithms”, *Intell. Control Autom*, vol. 6, pp. 6675–6679, 2008.

54. R. Aragues, J. Cortes, and C. Sagues, “Distributed consensus on robot networks for dynamically merging feature-based maps”, *IEEE Transactions on Robotics*, vol. 28(4), pp. 840–854, 2012.

55. P. Brass, F. Cabrera-Mora, A. Gasparri, and X. Jizhong, “Multirobot tree and graph exploration”, *IEEE Transactions on Robotics*, vol. 27(4), pp. 707–717, 2011. doi:10.1109/TRO.2011.2121170.

56. W. Burgard, M. Moors, C. Stachniss, and F. Schneider, “Coordinated multi-robot exploration”, *IEEE Transactions on Robotics*, vol. 21(3), pp. 376–386, 2005. doi:10.1109/TRO.2004.839232.

57. H. Choi, L. Brunet, and J. How, “Consensus-based decentralized auctions for robust task allocation”, *IEEE Transactions on Robotics*, vol. 25(4), pp. 912–926, 2009.

58. H. Costelha, and P. Lima, “Robot task plan representation by Petri nets: Modelling, identification, analysis and execution”, *Autonomous Robots*, vol. 33, no. 4, pp. 337–360, 2012. doi:10.1007/s10514-012-9288-x.

59. M. Defoort, T. Floquet, A. Kokosy, and W. Perruquetti, “Sliding-mode formation control for cooperative autonomous mobile robots”, *IEEE Transactions on Industrial Electronics*, vol. 55(11), pp. 3944–3953, 2008. doi:10.1109/TIE.2008.2002717.

60. D. Di Paola, A. Gasparri, D. Naso, and F. Lewis, “Decentralized discrete-event modeling and control of task execution for robotic networks”, in *IEEE 51st Annual Conference on Decision and Control (CDC)*, 2012, pp. 7346–7351. doi:10.1109/CDC.2012.6426687.

61. D. Di Paola, A. Gasparri, D. Naso, G. Ulivi, and F. L. Lewis, “Decentralized task sequencing and multiple mission control for heterogeneous robotic networks”, in *Proceedings of 2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp, 347-352. doi:10.1109/ICRA.2011.5980405.

62. M. Di Rocco, F. La Gala, and G. Ulivi, “Saetta: A small and cheap mobile unit to test multirobot algorithms”, *IEEE Robotics Automation Magazine*, 2012, pp. 52-62. doi:10.1109/MRA.2012.2185991.

63. A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi, “Consensus-based distributed intrusion detection for multi-robot systems”, in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 120–127. doi:10.1109/ROBOT.2008.4543196.

64. M. Zavlanos, and G.J. Pappas, “Distributed connectivity control of mobile networks”, *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, 2008.

65. S. Ansari, *Computer Vision Modeling on the Cloud in Building Computer Vision Applications Using Artificial Neural Networks*. Berkeley, CA, USA: Apress, 2020.

66. M.J.A.M. van Putten, *Neurostimulation in Dynamics of Neural Networks*. Berlin, Germany: Springer, 2020.

67. C. Chang, and C. Lin, “LIBSVM: a library for support vector machines”, *ACM Trans. Intell. Syst. Technol*, vol. 2, no. 3, pp. 1–27, 2011.

68. R. Fan, L. Chang, C. Hsieh, X. Wang, and C. Lin, “LIBLINEAR: a library for large linear classification”, *J. Mach. Learn,* vol. 9, pp. 1871–1874, 2008.

69. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative Adversarial Nets", *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

70. A. Krizhevsky, I. Sutskerver, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

71. U. Michelucci, *Advanced Applied Deep Learning Convolutional Neural Networks and Object Detection*. New York, USA: Springer, 2018.

72. M. Frampton, *Complete Guide to Open Source Big Data Stack*. New York, USA: Springer, 2018.

73. M. Salvaris, D. Dean, and W.H. Tok, *Deep Learning with Azure. Building and Deploying Artificial Intelligence Solutions on the Microsoft AI Platform*. New York, USA: Springer, 2018.

74. R. Abellera, L. Bulusu, *Oracle Business Intelligence with Machine Learning. Artificial Intelligence Techniques in OBIEE for Actionable BI*. New York, USA: Springer, 2018.

75. Q. Zhao, H. Yin, and H. Zhu, "Simultaneous recognition of two channels of optical packet headers utilizing reservoir computing subject to mutual-coupling optoelectronic feedback", *Optik*, vol. 157, pp. 951-956, 2018.

76. J. Zhu, and P. Sutton, "FPGA Implementations of neural networks - a survey of a decade of progress", in *Proc. International conference on field programmable logic and applications*, 2003, pp. 1062-1066.

77. C. Zhao, J. Li, L. Liu, L.S. Koutha, J. Liu, and Y. Yi, "Novel spike based reservoir node design with high performance spike delay loop", in *Proc. the 3rd ACM international conference on nanoscale computing and communication*, 2016, p. 14.

78. Q. Zhao, K. Nakajima, H. Sumioka, H. Hauser, and R. Pfeifer, "Spine dynamics as a computational resource in spine-driven quadruped locomotion", in *Proc.*

IEEE/RSJ international conference on intelligent robots and systems (IROS), 2013, pp. 1445-1451.

79. Y. Zhang, P. Li, Y. Jin, and Y. Choe, "A digital liquid state machine with biologically inspired learning and its application to speech recognition", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 2635-2649, 2015.

80. O. Yilmaz, "Machine learning using cellular automata based feature expansion and reservoir computing", *Journal of Cellular Automata*, vol. 10, pp. 435-472, 2015.

81. O. Yilmaz, "Symbolic computation using cellular automata-based hyperdimensional computing", *Neural Computation*, vol. 27, pp. 2661-2692, 2015.

82. I.B. Yildiz, H. Jaeger, and S.J. Kiebel, "Re-visiting the echo state property", *Neural Networks*, vol. 35, pp. 1-9, 2012.

83. X. Yang, W. Chen, and F.Z. Wang, "Investigations of the staircase memristor model and applications of memristor-based local connections", *Analog Integrated Circuits and Signal Processing*, vol. 87, pp. 263-273, 2016.

84. Y. Yi, Y. Liao, B. Wang, X. Fu, F. Shen, and H. Hou, "FPGA Based spike-time dependent encoder and reservoir design in neuromorphic computing processors", *Microprocessors and Microsystems*, vol. 46, pp. 175-183, 2016.

85. T. Yamazaki, and S. Tanaka, "The cerebellum as a liquid state machine", *Neural Networks*, vol. 20, pp. 290-297, 2007.

86. Y. Xia, C. Jahanchahi, and D.P. Mandic, "Quaternion-valued echo state networks", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 663-673, 2015.

87. T. Yamane, Y. Katayama, R. Nakane, G. Tanaka, and D. Nakano, "Wave-based reservoir computing by synchronization of coupled oscillators", in *Proc. International conference on neural information processing (ICONIP),* 2015, pp. 198-205.

88. S. Wolf, D. Awschalom, R. Buhrman, J. Daughton, S. Molnar, and M. Roukes, “Spintronics: a spin-based electronics vision for the future”, *Science*, vol. 294, pp. 1488-1495, 2001.

89. R.J. Williams, and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks”, *Neural Computation*, vol. 1, pp. 270-280, 1989.

90. G.M. Wojcik, and W.A. Kaminski, “Liquid state machine and its separation ability as function of electrical parameters of cell”, *Neurocomputing*, vol. 70, pp. 2593-2597, 2007.

91. P.J. Werbos, “Backpropagation through time: what it does and how to do it”, *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, 1990.

92. Q. Wang, Y. Li, and P. Li, “Liquid state machine based pattern recognition on fpga with firing-activity dependent power gating and approximate computing”, in *Proc. IEEE international symposium on circuits and systems (ISCAS)*, 2016, pp. 361-364.

93. Q. Wang, Y. Li, B. Shao, S. Dey, and P. Li, “Energy efficient parallel neuromorphic architectures with approximate arithmetic on FPGA”, *Neurocomputing*, vol. 221, pp. 146-158, 2017.

94. Q. Wang, Y. Jin, and P. Li, “General-purpose lsm learning processor architecture and theoretically guided design space exploration”, in *Proc. IEEE biomedical circuits and systems conference (BioCAS)*, 2015, pp. 1-4.

95. Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, and M. Haelterman, “High-performance photonic reservoir computer based on a coherently driven passive cavity”, *Optica*, vol. 2, pp. 438-446, 2015.

96. F. Walter, F. Röhrbein, and A. Knoll, “Neuromorphic implementations of neurobiological learning algorithms for spiking neural networks”, *Neural Networks*, vol. 72, pp. 152-167, 2015.

97. P. Vincent-Lamarre, G. Lajoie, and J.-P. Thivierge, “Driving reservoir models with oscillations: a solution to the extreme structural sensitivity of chaotic networks”, *Journal of Computational Neuroscience*, vol. 41, pp. 305-322, 2016.

98. D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, “Isolated word recognition with the liquid state machine: a case study”, *Information Processing Letters*, vol. 95, pp. 521-528, 2005.

99. D. Verstraeten, B. Schrauwen, M. d’Haene, and D. Stroobandt, “An experimental unification of reservoir computing methods”, *Neural Networks*, vol. 20, pp. 391-403, 2007.

100. K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, and P. Bienstman, “Toward optical signal processing using photonic reservoir computing”, *Optics Express*, vol. 16, pp. 11182-11192, 2008.

101. O. Breuleux, Y. Bengio, and P. Vincent, “Quickly generating representative samples from an RBM-derived process”, *Neural Computation*, vol. 23, no. 8, pp. 2053–2073, 2011.

102. G. Hinton et al., “Deep neural networks for acoustic modeling in speech recognition”, *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

103. G. E. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets”, Neural Computation, vol. 18, pp. 1527–1554, 2006.

104. K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?”, in *Proc. International Conference on Computer Vision (ICCV’09)*, 2009, pp. 2146–2153.

105. G. Zhou, K. Sohn, and H. Lee, “Online incremental feature learning with denoising autoencoders”, in *Proc. International conference on artificial intelligence and statistics*, 2012, pp. 1453-1461.

106. T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, “Error-driven incremental learning in deep convolutional neural network for large-scale image

classification", in *Proc. of the ACM International Conference on Multimedia*, 2014, pp. 177-186.

107. K. Weiss, T.M. Khoshgoftaar, and D.D. Wang, "A survey of transfer learning", *Journal of Big Data*, vol. 3, no. 9, pp. 205-214, 2016.

108. D. Tse et al., "Schema-dependent gene activation and memory encoding in neocortex", *Science*, vol. 333, pp. 891-895, 2011.

109. B.E. Stein, T.R. Stanford, and B.A. Rowland, "Development of multisensory integration from the perspective of the individual neuron", *Nature Reviews Neuroscience*, vol. 15, no. 8, pp. 520-535, 2014.

110. S.F. Sorrells et al., "Human hippocampal neurogenesis drops sharply in children to undetectable levels in adults", *Nature*, vol. 555, pp. 377-381, 2018.

111. A. Soltoggio, "Short-term plasticity as cause-effect hypothesis testing is distal reward learning", *Biological Cybernetics*, vol. 109, pp. 75-94, 2015.

112. F.M. Richardson, and M.S. Thomas, "Critical periods and catastrophic interference effects in the development of self-organising feature maps", *Developmental Science*, vol. 11, no. 3, pp. 371-389, 2008.

113. B. Ren, H. Wang, J. Li, and H. Gao, "Life-long learning based on dynamic combination model", *Applied Soft Computing*, vol. 56, pp. 398-404, 2017.

114. G. Quadrato, M.Y. Elnaggar, and S. Di Giovanni, "Adult neurogenesis in brain repair: Cellular plasticity vs. cellular replacement", *Frontiers in Neuroscience*, vol. 8, no. 17, pp. 675-683, 2014.

115. J.D. Power, and B.L. Schlaggar, "Neural plasticity across the lifespan", *Wiley Interdisciplinary Reviews: Developmental Biology*, vol. 6, pp. 216-222, 2016.

116. G.I. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of humans actions with deep neural network self-organization", *Neural Networks*, vol. 96, pp. 137-149, 2017.

117. G.I. Parisi, S. Magg, and S. Wermter, “Human motion assessment in real time using recurrent self-organization”, in *Proc. of the IEEE international symposium on robot and human interactive communication*, 2016, pp. 71-79.

118. S.J. Pan, and Q. Yang, “A survey on transfer learning”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2010.

119. L. Mici, G.I. Parisi, and S. Wermter, “A self-organizing neural network architecture for learning human-object interactions”, *Neurocomputing*, vol. 307, pp. 14-24, 2018.

120. P. Li, I. Farkas, B. McWhinney, “Early lexical development in a self-organising neural network”, *Neural Networks*, vol. 17, pp. 1345-1362, 2004.

121. Y. LeCun, Y. Bengio, G. Hinton, “Deep learning”, *Nature*, vol. 521, pp. 436-444, 2015.

122. D. Kumaran, D. Hassabis, J.L. McClelland, “What learning systems do intelligent agents need? Complementary learning systems theory updated”, *Trends in Cognitive Sciences*, vol. 20, no. 7, pp. 512-534, 2016.

123. J. Wielemaker, and V.S. Costa, “On the Portability of Prolog Applications”, in *International Symposium on Practical Aspects of Declarative Languages PADL 2011: Practical Aspects of Declarative Languages*, 2011, pp. 69-83.

124. L. De Koninck, T. Schrijvers, and B. Demoen, “A flexible search framework for CHR”, *LNCS*, vol. 5388, pp. 16-47, 2008.

125. P. Szabo, and P. Szeredi, “Improving the ISO prolog standard by analyzing compliance test results”, *LNCS*, vol. 4079, pp. 257-269, 2006.

126. G. van Noord, “At Last Parsing is Now Operational”, in *13e Conference sur Le Traitement Automatique des Langues Naturelles*, 2006, pp. 20–42.

127. A.F. Gad, *Practical Computer Vision Applications Using Deep Learning with CNNs with Detailed Examples in Python Using TensorFlow and Kivy*. New York, USA: Springer, 2018.

128. S. Nagar, *Introduction to Python for Engineers and Scientists*. New York, USA: Springer, 2018.

129. S. Mukhopadhyay, *Advanced Data Analytics Using Python with Machine Learning, Deep Learning and NLP Examples*. New York, USA: Springer, 2018.

130. L. Allison, "Models for Machine Learning and Data Mining in Functional Programming", *Journal of Functional Programming*, vol. 15, no. 1, pp. 15-32, 2005.

131. H. Bauer, Y. Goh, S. Schlink, and C. Thomas, "The Supercomputer in Your Pocket", *McKinsey on Semiconductors*, vol. 112, pp. 14-27, 2012.

132. D. Chen, and H. Zhao, "Data Security and Privacy Protection Issues in Cloud Computing", In *Proc. of the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, 2012, pp. 647-651.

133. R. Coppola, and M. Morisio, "Connected Car: Technologies, Issues, Future Trends", *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, pp. 1-36, 2016.

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. V. Savchenko, V. Akhramovych, A. Tushych, I. Sribna, and I. Vlasov, “Analysis of Social Network Parameters and the Likelihood of its Construction”, *International Journal of Emerging Trends in Engineering Research*, vol. 8, no 2, pp. 271-276, 2020.

2. K. Storchak, N. Yakovenko, A. Tushych, I. Sribnaya, and O. Polonevich, “Improving the material quality of network equipment due to a mechanism of surface hardening”, *Scientific discussion*, vol. 1, no. 49, pp. 34-39, 2020.

3. А.М. Тушич, К.П. Сторчак, А.П. Бондарчук, та А.О. Макаренко, “Вимоги до інтелектуальних систем аналізу даних та їх класифікацій”, *Науково-технічний журнал “Телекомунікаційні та інформаційні технології”,* №1, с. 31-36, 2019.

4. К.П. Сторчак, А.М. Тушич, та А.П. Бондарчук, “Кластерний аналіз даних з використанням штучних нейронних мереж”, *Науковий журнал “Зв’язок”,* №6, с. 36-38, 2018.

5. К.П. Сторчак, А.М. Тушич, К.С. Козелкова, та М.М. Степанов, “Інтелектуальний аналіз даних з використанням нейронних мереж”, *Науковий журнал “Зв’язок”,* №4, с. 17-19, 2018.

6. К.П. Сторчак, А.М. Тушич, О.М. Ткаленко, В.М. Чорна, та Т.М. Жила, “Аналіз вимог до проектування хмарної платформи для інтернету речей”, *Науковий журнал “Зв’язок”,* №6, с. 26-34, 2019.

7. О.А. Золотухіна, О.М. Ткаленко, А.М. Тушич, В.М. Чорна, та О.Р. Нікітенко, “Концепція розвитку підсистеми передавання мультимедійних

повідомлень IMS", *Науково-технічний журнал "Телекомунікаційні та інформаційні технології"*, №4, с. 81-89, 2019.

8. І.С. Сиротенко, І.С. Щербина, К.П. Сторчак, та А.М. Тушич, "Аналіз ефективності використання нейронних мереж на прикладі багатошарового персептрону та мережі Кохонена", *Науковий журнал "Зв'язок",* №5, с. 17-19, 2020.

9. Б.В. Шефкін, І.В. Красюк, В.О. Хоменчук, К.П. Сторчак, та А.М. Тушич, "Дослідження та впровадження нейронної мережі на основі TENSORFLOW", *Науковий журнал "Зв'язок",* №6, с. 18-20, 2020.

10. К.П. Сторчак, Д.В. Кравець, А.М. Тушич, та Д.В. Сорокін, "Аналіз методів організації прав користувачів у GNU\Linux системах", *Науковий журнал "Зв'язок",* №4, с. 38-40, 2020.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

11. А.М. Тушич, "Використання штучних нейронних мереж для створення IoT рішення фільтрації VPN трафіку", на *XI Міжнар. наук.-техн. конф. студентів та аспірантів "Перспективи розвитку інформаційно-телекомунікаційних технологій та систем"*, Київ, 2019, с. 361.

12. А.М. Тушич, "Машинне навчання з використанням TENSORFLOW", на *XII Міжнар. наук.-техн. конф. студентів та аспірантів "Перспективи розвитку інформаційно-телекомунікаційних технологій та систем",* Київ, 2020, с. 379.

13. М. Пелепей, та А. Тушич, "Штучний інтелект – друг, ворог чи помічник людини", на *IX Міжнар. наук.-техн. конф. студентства та молоді "Світ інформації та телекомунікацій"*, Київ, 2019, с. 303-304.

14. А.М. Тушич, "Аналіз доцільності використання автоматизованої системи інтелектуального аналізу даних на основі штучних нейронних мереж",

на *VII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених з автоматичного управління присвячена Дню космонавтики*, Херсон, 2019, с. 76-77.

15. Д.Я. Алтинніков, та А.М. Тушич, “Як штучний інтелект та IOT доповнюють один одного”, на *Всеукр. наук.-техн. конф. “Сучасний стан та перспективи розвитку IOT”*, Київ, 2020, с. 303-304.

16. А.М. Тушич, та П.В. Лебединець, “Дослідження системи моніторингу Zabbix для IT-інфраструктури підприємства”, на *VII Наук.-техн. конф. “Сучасні інфокомунікаційні технології”*, Київ, 2018, с. 20-21.

17. А.М. Тушич, та О.М. Скрипаль. “Безпека функціонування телекомунікаційніх систем та мереж”, на *XII Міжнар. наук.-техн. конф. “Проблеми інформатизації”*, Київ, 2018р. – С.75-76.

18. А.М. Тушич, та П.В. Лебединець, “Дослідження відкритого програмного забезпечення поштового сервера та клієнта”, на *XII Міжнар. наук.-техн. конф “Проблеми інформатизації”*, Київ, 2018, с. 105.

19. Б. Свердлюк, Ю.Каграманова, та А. Тушич, “Інтернет речей”, на *IX Міжнар. наук.-техн. конф. студентства та молоді “Світ інформації та телекомунікацій*”, Київ, 2019, с. 107-108.

ДОДАТОК Б

```
%%%-------------------------------------------------------------------
%%% @author atushych
%%% @copyright (C) 2020, <COMPANY>
%%% @doc
%%%
%%% @end
%%% Created : 14. Feb 2020 22:15
%%%-------------------------------------------------------------------
-module(helper).
-author("atushych").

%% API
-export([loop/1, new/2, test/0, print/1, stop/1, pulse/2, true_output/2]).


new(Layers, Memory_length) ->

  Receptors_size = lists:nth(1, Layers),
  L1 = generate_layer(lists:duplicate(Receptors_size, 0)),

  M = lists:flatten(lists:map(fun(P) -> L = generate_layer(lists:seq(0, Memory_length)), neurons:register_link(P, L, 1),
    L end, L1)),

  Hidden_layers = lists:sublist(Layers, 2, length(Layers) - 2),
  L2 = generate_hidden_layers(Hidden_layers),

  Effectors_size = lists:last(Layers),
  L3 = generate_layer(lists:duplicate(Effectors_size, 0)),

  T = generate_layer(lists:duplicate(2, 0)),


  recorder:recording_layer_to_layer(M, lists:nth(1, L2)),
  recorder:recording_between_layers(L2),
  recorder:recording_layer_to_layer(lists:last(L2), L3),
  recorder:recording_layer_to_layer(T, lists:flatten(L2 ++ L3)),

  Data = #{
    hidden_layers => L2,
    receptors => L1,
    effectors => L3,
    memory => M,
    tone => T,
    last_out => []

  },
  spawn(net, loop, [Data]).


test() ->
  Net1 = new([4, 3, 2, 1], 5),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),
```

```
  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  pulse(Net1, [0.9, 0.8, 0.7, 0.1]),
  true_output(Net1, [1]),

  true.


print(Net) when is_pid(Net) ->
  Net ! {request, self(), print}.

stop(Net) when is_pid(Net) ->
  Net ! {request, self(), stop}.

pulse(Net, Powers) when is_pid(Net), is_list(Powers) ->
  Net ! {request, self(), {pulse, Powers}},
  receive
    {reply, Net, {effect, Reply}} -> Reply
  end.

true_output(Net, L) when is_pid(Net), is_list(L) ->
  Net ! {request, self(), {true_output, L}},
  receive
    {reply, Net, {confirm_true_output, ok}} -> ok
  end.



generate_layer(A) ->
  generate_layer(A, []).
generate_layer([], Acc) ->
  Acc;
generate_layer([H | T], Acc) ->
  N = neurons:new(H),
  generate_layer(T, Acc ++ [N]).


generate_hidden_layers(L) ->
  generate_hidden_layers(L, []).
generate_hidden_layers([], Acc) ->
  Acc;
generate_hidden_layers([H | T], Acc) ->
  L = generate_layer(lists:duplicate(H, 0)),
  generate_hidden_layers(T, Acc ++ [L]).
```

```
pulse_to_layer([], []) ->
  true;
pulse_to_layer([NH | NT], [PH | PT]) when is_pid(NH) ->
  neurons:pulse(NH, PH),
  pulse_to_layer(NT, PT).



get_effect(Effectors, EffectAcc) when is_list(Effectors), is_map(EffectAcc) ->
  case {length(Effectors), maps:size(EffectAcc)} of
    {_X, _X} ->
      R = lists:map(fun(P) -> maps:get(P, EffectAcc) end, Effectors),
      R;
    {_X, _Y} ->
      receive
        {reply, Pid, {effect, Value}} ->
          NewEffectAcc = maps:put(Pid, Value, EffectAcc),
          get_effect(Effectors, NewEffectAcc)
      after
        25000 ->
          {error, timeout}
      end
  end.

confirm_back_error(Effectors, Acc) when is_list(Effectors), is_map(Acc) ->
  case {length(Effectors), maps:size(Acc)} of
    {_X, _X} ->
      {ok, ok};
    {_X, _Y} ->
      receive
        {reply, Pid, {confirm_back_error, Value}} ->
          NewAcc = maps:put(Pid, Value, Acc),
          confirm_back_error(Effectors, NewAcc)
      after
        25000 ->
          {error, timeout}
      end
  end.


loop(Data) ->
  receive
    {reply, _, ok} ->
      loop(Data);
    {request, _, stop} ->
      true;
    {request, Pid, print} ->
      io:format("Net~w ~w~n", [self(), Data]),
      io:format("Receptor layer:~n"),
      neurons:print(maps:get(receptors, Data)),
      io:format("Memory layer:~n"),
      neurons:print(maps:get(memory, Data)),
      io:format("Hidden layer:~n"),
      lists:foreach(fun(P) -> neurons:print(P) end, maps:get(hidden_layers, Data)),
      io:format("Effector layer:~n"),
      neurons:print(maps:get(effectors, Data)),
```

```
    io:format("Tone layer:~n"),
    neurons:print(maps:get(tone, Data)),
    loop(Data);
  {request, Pid, {pulse, PowerList}} ->
    Receptors = maps:get(receptors, Data),
    Tone = maps:get(tone, Data),
    pulse_to_layer(Receptors, PowerList),
    pulse_to_layer(Tone, [-1, 1]),
    Effect = get_effect(maps:get(effectors, Data), #{}),
    Pid ! {reply, self(), {effect, Effect}},
    NewData1 = maps:put(last_out, Effect, Data),
    io:format("Net~w: send effect ~w to ~w~n", [self(), Effect, Pid]),
    loop(NewData1);

  {request, Pid, {back_error, Errors}} when is_list(Errors) ->
    io:format("Net~w: request error = ~w ~n", [self(), Errors]),

    Effectors = maps:get(effectors, Data),

    Zip = lists:zip(Effectors, Errors),

    lists:foreach(fun(P) -> {N, E} = P, N ! {request, self(), {back_error, E}} end, Zip),

    confirm_back_error(Effectors, #{}),
    Pid ! {reply, self(), {confirm_back_error, ok}},
    loop(Data);

  {request, Pid, {true_output, L}} ->

    Last_out = maps:get(last_out, Data),
    Effectors = maps:get(effectors, Data),
    Errors = lists:zipwith(fun(X, Y) -> X - Y end, L, Last_out),

    Zip = lists:zip(Effectors, Errors),
    lists:foreach(fun(P) -> {N, E} = P, N ! {request, self(), {back_error, E}} end, Zip),
    confirm_back_error(Effectors, #{}),

    io:format("Net~w: Errors = ~w ~n", [self(), Errors]),

    Pid ! {reply, self(), {confirm_true_output, ok}},
    loop(Data)

after
  25000 ->
    true
end.
```

ДОДАТОК В

ЗАТВЕРДЖУЮ
Директор
ТОВ "Хуавей Україна"
_____ Ма Ци
(підпис, ініціали, прізвище)
«__» _____ 2021 р.

Акт

про реалізацію результатів наукових досліджень
Тушич Аліни Миколаївни

Матеріали роботи використано в науково-дослідних і дослідно-конструкторських роботах, що ведуться в ТОВ "Хуавей Україна", а саме, при вдосконаленні сімейства All-Flash систем зберігання даних Huawei OceanStor.

В ТОВ "Хуавей Україна" використані наступні результати:

1. Розроблений підхід до виявлення принципів роботи тестової системи на навченій нейронній мережі, який характеризується угрупуванням вхідних параметрів і активності нейронів мережі, дозволив покращити процедуру виявлення правил в даних, що зберігаються.
2. Розроблений метод нелінійної нормалізації даних, що базується на послідовному виконанні перетворень змінного типу нелінійності, дозволив збільшити точність і швидкодію обробки даних у системах зберігання All-Flash.

Матеріали роботи «Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж» дозволять вдосконалити модулі та алгоритми прискорювача ШІ системи зберігання даних All-flash OceanStor Dorado, що реалізує інтелектуальні операції і технічне обслуговування (O&M).

Директор ТОВ "Хуавей Україна" Ма Ци

ЗАТВЕРДЖУЮ

ФОП «Бордіян Олег Васильович»

_____ О.В. Бордіян

(підпис, ініціали, прізвище)

«17» 12 2020 р.

Акт

про реалізацію результатів наукових досліджень
Тушич Аліни Миколаївни

Матеріали роботи використано в науково-дослідних роботах, що ведуться у ФОП «Бордіян Олег Васильович», профілем яких є системи безпеки, «Розумний будинок», IoT, а саме, при створенні програмного забезпечення для IoT рішень, зокрема систем безпеки.

У даному ФОП використані наступні результати:

1. Розроблена інформаційна технологія визначення закономірностей в накопичених даних на основі методу нелінійної нормалізації, методик навчання та виявлення принципів роботи системи, яка дозволить покращити можливості IoT рішень, зокрема додасть функцію аналізу даних у програмному забезпеченні систем обліку робочого часу.
2. Розроблений алгоритм навчання нейронної мережі, що базується на методі задання адаптивних параметрів, дозволив збільшити швидкість навчання нейронної мережі, на основі якої працює термінал розпізнавання обличь, на 2,1%.

Матеріали роботи «Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж» дали змогу удосконалити етапи обробки даних в IoT рішеннях.

ФОП «Бордіян Олег Васильович» О.В. Бордіян

ЗАТВЕРДЖУЮ
Директор
ТОВ "Ай Ті Джи"
_____ Ю.А. Заславський
(підпис, ініціали, прізвище)
«13» 12 2020 р.

Акт

про реалізацію результатів наукових досліджень
Тушич Аліни Миколаївни

Матеріали роботи використано в науково-дослідних роботах, що ведуться у ТОВ "Ай Ті Джи", а саме, при створенні програмного забезпечення для IoT пристроїв.

У ТОВ "Ай Ті Джи" використані наступні результати:

1. Розроблений метод нелінійної нормалізації даних, що базується на послідовному виконанні перетворень змінного типу нелінійності, який дозволив істотно збільшити точність на 1,3% і швидкодію обробки даних на 2,3%, що у промислових масштабах є достатнім показником для впровадження.
2. Розроблено алгоритм навчання нейронної мережі, що базується на методі задання адаптивних параметрів, який дозволив збільшити швидкість навчання мережі на 1,7%. Тому, в представленому програмному забезпеченні ведуться роботи зі зміни алгоритму навчання нейронної мережі, що використовується у останніх стабільних версіях програмного забезпечення.

Матеріали роботи «Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж» дали змогу автоматизувати етапи обробки даних, що збираються з IoT пристроїв.

Директор
ТОВ "Ай Ті Джи" Ю.А. Заславський

„ЗАТВЕРДЖУЮ"

Проректор з науково-педагогічної роботи ДУТ,
доктор технічних наук, професор

_____ Беркман Л.Н.

"31" 08 2022 р.

АКТ

використання у навчальному процесі Навчально-наукового інституту інформаційних технологій результатів дисертаційної роботи старшого викладача кафедри Інформаційних систем та технологій Державного університету телекомунікацій Тушич Аліни Миколаївни на тему: «Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж» на здобуття наукового ступеня доктора філософії за спеціальністю 123 – комп'ютерна інженерія.

Комісія у складі:

голова – директор Навчально-наукового інституту інформаційних технологій, доктор технічних наук, професор Бондарчук А.П.; члени комісії – завідувач кафедри Комп'ютерної інженерії, доктор технічних наук, доцент Ткаченко О.М.; завідувач кафедри Штучного інтелекту, кандидат технічних наук, доцент Зінченко О.В.; завідувач кафедри Інженерії програмного забезпечення, кандидат технічних наук, доцент Негоденко О.В.

розглянула дисертаційну роботу Тушич Аліни Миколаївни на тему: «Методика побудови інтелектуальної системи аналізу даних на основі нейронних мереж» та публікації автора за матеріалами дисертаційної роботи. Результати впроваджено в начальний процес Навчально-наукового інституту інформаційних технологій, а саме:

1. Розроблений метод нелінійної нормалізації даних, що базується на послідовному виконанні перетворень змінного типу нелінійності.

2. Розроблений алгоритм навчання нейронної мережі, що базується на методі задання адаптивних параметрів.

3. Розроблена інформаційна технологія визначення закономірностей в накопичених даних на основі методу нелінійної нормалізації, методик навчання та виявлення принципів роботи системи.

На основі аналізу представлених матеріалів комісія установила, що отримані наукові результати використовуються в навчальному процесі Державного університету телекомунікацій при викладанні навчальних дисциплін: «Машинне навчання та обробка даних в IoT» та «Штучний інтелект».

Голова комісії

Директор ННІІТ
доктор технічних наук, професор А.П. Бондарчук

Члени комісії

Завідувач кафедри Комп'ютерної інженерії,
доктор технічних наук, доцент О.М. Ткаченко

Завідувач кафедри Штучного інтелекту,
кандидат технічних наук, доцент О.В. Зінченко

Завідувач кафедри Інженерії програмного забезпечення,
кандидат технічних наук, доцент О.В. Негоденко