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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.9

«До захисту допущено»
Завідувач кафедри СПСКС

_____
В.П.Тарасенко_
(підпис) (ініціали, прізвище)
"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Системне програмування

на тему: Евристичний метод тренування штучної нейронної мережі

Виконав: студент II курсу, групи _КВ-73мп_____
(шифр групи)

Карвацький Сергій Сергійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник. доц.каф.СПСКС, к.т.н., доцент Зорін Ю. М. _____
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

Київ – 2018 року

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

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія

Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)

«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Карвацькому Сергію Сергійовичу

1. Тема дисертації: Евристичний метод тренування штучної нейронної мережі, науковий керівник дисертації к.т.н., доцент Зорін Юрій Михайлович, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації 07 грудня 2018 р.

3. Об'єкт дослідження: процес тренування штучної нейронної мережі.

4. Предмет дослідження: евристичний метод тренування штучної нейронної мережі.

5. Перелік завдань, які потрібно розробити

   - опис предметної області досліджень та обґрунтування задачі тренування штучної нейронної мережі;
   - модифікований евристичний алгоритм для тренування штучної нейронної мережі.

– провести порівняльний аналіз розробленого методу з існуючими методами тренування штучнох нейронної мережі

6. Перелік ілюстративного матеріалу: 28 рисунків, 6 таблиць.

7. Перелік публікацій

– «Модифікований алгоритм зозуі для навчання штучної нейронної мережі», X конференція молодих вчених ПМК-2018-1. – 2018;

– «Евристичний метод тренування штучної нейронної мережі», V Міжнародна науково-технічна конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами». – 2018;

8. Дата видачі завдання 5 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення літератури за тематикою проекту	05.09.2017	
2	Аналіз існуючих рішень	20.01.2018	
3	Підготовка матеріалів першого розділу магістерської дисертації	09.03.2018	
4	Підготовка матеріалів другого розділу магістерської дисертації	30.04.2018	
5	Підготовка матеріалів третього розділу магістерської дисертації	10.09.2018	
6	Підготовка графічної частини дипломного проекту	16.10.2018	
7	Оформлення документації дипломного проекту	01.11.2018	
8	Попередній розгляд магістерської дисертації на кафедрі	26.11.2018	

Студент _____ Карвацький С.С
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Зорін Ю.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми.

Штучні нейронні мережі знаходять застосування у наступних сферах: класифікація та розпізнавання образів, системи асоціативної пам'яті, компресія даних, оптимізаційні задачі, теорія керування, розробка нейрокомп'ютерів, наближення функцій з високою точністю, екстраполяція та прогнозування. Відомі алгоритми тренування штучної нейронної мережі працюють відносно довго, та результат тренування з їх використанням не завжди задовільний. Тому розробка удосконалених методів тренування є актуальною та перспективною задачею. Запропонований у даній роботі алгоритм дозволяє ефективніше навчати нейронні мережі майже будь-якої структури, отже може бути застосований у вище наведених сферах.

Об'єктом дослідження є процес тренування штучної нейронної мережі.

Предметом дослідження є евристичні методи для тренування штучної нейронної мережі.

Мета роботи: розробка удосконаленого евристичного алгоритму тренування штучної нейронної мережі, що характеризується вищою швидкодією, ніж відомі методи.

Наукова новизна:

1. Запропоновано удосконалений алгоритм навчання штучної нейронної мережі мережі, що характеризується вищою швидкодією, ніж відомі методи.

2. Виконано порівняльний аналіз запропонованого методу з існуючими, визначено в яких саме ситуаціях потрібно використовувати даний метод, його переваги та недоліки порівняно з існуючими методами навчання штучної нейронної мережі.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований алгоритм дає змогу ефективніше тренувати штучну

нейронну мережу в порівнянні з відомими алгоритмами. Запропонований аглоритм дозволяє отримати приріст в швидкодії тренування штучної нейронної мережі.

Апробація роботи. Запропонований підхід був представлений та обговорений на науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2018 (Київ, 21 – 23 березня 2018 р.), а також у V Міжнародній науково-технічній конференції «Сучасні методи, інформаційне, програмне татехнічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (Київ, 22-23 листопада 2018 р.).

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

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів

У першому розділі розглянуто існуючі методи тренування штучних нейронних мереж, історію виникнення нейронних мереж, та загальні відомості про них.

У другому розділі модифікований евристичний алгоритм, для розв’язання задачі тренування штічної нейронної мережі.

У третьому розділі проведено тестування модифікованого алгоритму, та обґрнтовано доцільність модифікацій.

У четвертому розділі представлено результати тренування штучної нейронної мережі запропонованим алгоритмом та проведено аналіз отриманих результатів.

У висновках підсумовано результати проведеної роботи.

Магістерська дисертація представлена на 80 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: Штучна нейронна мережа, метод рою частинок, градієнтний спуск, метод зворотнього поширення помилки, багатошаровий перцептрон, нейрон.

ABSTRACT

Actuality of theme.

Artificial neural networks are used in the following areas: classification and pattern recognition, associative memory systems, data compression, optimization tasks, control theory, neurocomputer development, high-precision functions approximation, extrapolation and prediction. Known algorithms for training an artificial neural network work relatively long, and the result of training with their use is not always acceptable. Therefore, the development of advanced training methods is an urgent and promising task. The algorithm proposed in this paper allows to effectively teach neural networks of almost any structure, therefore it can be applied in the above mentioned spheres.

The object of the study is the process of training an artificial neural network.

The subject of the study is heuristic methods for training an artificial neural network.

Purpose: the development of an improved heuristic algorithm for training an artificial neural network, characterized by higher speed than the known methods.

Scientific novelty:

1. An improved algorithm for training an artificial neural network of the network, characterized by a higher speed than the known methods, is proposed.

2. A comparative analysis of the proposed method with the existing ones is made, it is determined in which situations it is necessary to use this method, its advantages and disadvantages in comparison with the existing methods of training artificial neural network.

The practical value of the results obtained in the work is that the proposed algorithm makes it possible to train the artificial neural network more efficiently than known algorithms. The proposed algorithm allows you to get an increase in the speed of training an artificial neural network.

Test work. The proposed approach was presented and discussed at the scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kiev, March 21-23, 2018), as well as at the V International Scientific and Technical Conference "Modern Methods, Information, software and technical support for control systems for organizational, technical and technological complexes "(Kyiv, November 22-23, 2018).

Structure and scope of work. This paper consists of an introduction, four chapters and conclusions.

The introduction gives a general description of the work, assesses the current state of the problem, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained

The first chapter examines the existing methods of training artificial neural networks, the history of the emergence of neural networks, and general information about them.

In *the second chapter*, a heuristic algorithm is modified to solve the problem of training the artificial neural network.

In *the third chapter* the modified algorithm is tested, and justified the feasibility of modifications.

In *the fourth section* presented the results of the training of the artificial neural network by the proposed algorithm and analyzes the results.

In *the conclusions* the results of the work were summarized.

The master's dissertation is presented on 80 sheets, contains the list of used literary sources.

Key words: artificial neural network, particle swarm optimization, gradient descent, backpropagation method, multilayer perceptron, neuron.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ШНМ – штучна нейронна мережа

МРЧ (англ. PSO – Particle swarm optimization) – метод рою частинок

ММРЧ – модифікований метод рою частинок

МЗПП – метод зворотнього поширення помилки (англ. backpropagation)

ADALINE (Multiple ADAptive LINear Elements) - Адаптивні Лінійні Елементи

ФА – функція активації

ЦФ – цільова функція – функція, що оптимізується.

ВСТУП

За останні десятиліття відбувся перехід до так званого «інформатизованого суспільства». Його концепція відображає зростання темпів виробництва, розподілу, переробки та споживання інформації. Це спричинило зростання актуальності розробки та модернизації інформаційних технологій. Важливе місце серед них займають алгоритмічні та програмно-апаратні системи і комплекси з елементами штучного інтелекту, призначені розв’язувати інтелектуальні задачі та виконувати функції, які раніше вважалися прерогативою людини.

Також, значного прогресу було досягнуто біологами в області дослідження роботи мозку. Досліджуючи структуру і функції нервової системи людини, вони багато чого дізналися про функціонування мозку.

У процесі досліджень з'ясувалося, що мозок має приголомшуючу складність: мільярди нейронів, кожен з яких з'єднаний з сотнями або тисячами інших, утворюють систему, яка далеко перевершує наші найсміливіші мрії про суперкомп'ютери.

Концепцію «штучна нейронна мережа» вперше було вжито в 40-х роках минулого століття. На рівні логіки, діяльність нервової системи людини і тварин було перенесено на концепцію штучної нейромережі. У 1943-му році формальна модель нейрона була розроблена, хоча в ті часи вона мала певні недоліки, та могла вирішувати невелику кількість задач. Ці недоліки вдалося виправити, об'єднуючи нейрони в мережі по декілька шарів. Такі системи виявилися набагато більш гнучкими: об'єднані в мережу формальні нейрони можуть вирішувати завдання, які традиційно відносяться до області «людської діяльності» (наприклад прийняття рішень на основі неповної інформації і навіть розпізнавання образів). Найромережі виявились здатними запам’ятовувати інформацію та навчатися, що нагадує розумові процеси людини. Саме тому в ранніх роботах по дослідженню нейромереж часто згадувався термін «штучний інтелект».

За останній час швидко зростав інтерес до нейронних мереж. Ними зайнялися фахівці зі, здавалося б не споріднених областей, — техніки, фізіології, психології. Цей інтерес зрозумілий, так як штучна нейронна мережа, по суті, являє собою модель природної нервової системи, тому створення і вивчення таких мереж дозволяє дізнатися багато про функціонування природних систем.

Сама теорія штучних нейронних мереж з'явилася в 40-х роках завдяки останнім на той момент досягнень біології, так як штучні нейрони складаються з елементів, які моделюють елементарні функції біологічних нейронів. Ці елементи організуються за способом, який може відповідати (або не відповідати) анатомії мозку.

Незважаючи на таку поверхневу подібність, штучні нейронні мережі демонструють дивовижні властивості, подібні до властивостей природного мозку. Наприклад, штучна нейронна мережа здатна змінювати свою поведінку в залежності від зовнішнього середовища. Прочитавши пред'явлені їй вхідні сигнали (можливо, разом з необхідними виходами) вона здатна «навчитися» так, щоб забезпечувати необхідну реакцію.

Після тренування мережа не реагує на невеликі зміни вхідних сигналів. Ця здатність бачити образ крізь шум і спотворення дуже корисна, якщо потрібно вирішувати задачі розпізнавання образів.

Варто відзначити, що нейронна мережа робить узагальнення автоматично завдяки своїй структурі, а не за допомогою спеціально написаних програм.

Іншою цікавою властивістю нейромереж є надійність: навіть якщо кілька елементів працюватимуть неправильно або вийдуть з ладу, то мережа все одно буде здатна видавати правильні результати, але з меншою точністю.

Деякі типи нейронних мереж мають здатність генерувати абстрактний образ на основі декількох вхідних сигналів. Наприклад, можна навчити мережу, пред'являючи їй послідовність спотворених зображень букви «А». Після тренування мережа зможе породити букву «А» без спотворень, тобто

мережа може згенерувати дещо унікальне, на основі інформації, здобутої в процесі тренування.

Штучні нейронні мережі застосовуються у різноманітних сферах, зв'язаних з обробкою інформації. Наприклад в таких галузях як: розпізнавання образів, системи пам'яті, що основані на асоціаціях, класифікація, компресія даних, задачі оптимізації, теорія керування, вирішення інженерних задач проектування, екстраполяція та прогнозування. Таким чином, вивчення методів тренування нейронних мереж в даний час є дійсно актуальною темою.

Метою написання роботи є вивчення евристичних методів оцінки тренування штучних нейронних мереж.

При написанні роботи були поставлені наступні завдання:

1. Виконати порівняльний огляд існуючих методів тренування нейронних мереж;

2. Вивчити евристичні методи тренування нейронних мереж та порівняти з іншими методами;

3. Розробити удосконалений евристичний алгоритму тренування штучної нейронної мережі, що характеризується вищою швидкодією, ніж відомі методи.

4. Провести порівняльний аналіз розробленого евристичного алгоритму з існуючими методами тренування штучної нейронної мережі.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ЗАГАЛЬНІ ВІДОМОСТІ ПРО ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

1.1 Історія вивчення нейронних мереж

Для початку розглянемо історію вивчення нейронних мереж.

Перший крок в дослідженні нейромереж був зроблений в 1943 р, коли вийшла статися нейрофізіолога Уоррена Маккалоха (Warren McCulloch) і математика Уолтера Питтса (Walter Pitts), присвячена штучним нейронам, а також реалізації моделі нейронної мережі за допомогою електричних схем.

1949 р.Д. Хебб (D. Hebb) запровадив ідеї про особливості з'єднань нейронів мозку та їх взаємодії між собою, а також запропонував правила тренування нейронної мережі.

1950-і рр. З'явилися програмні моделі перших ШНМ. Одні з перших робіт були проведені Натаніелом Рочестером (Nathanial Rochester) з лабораторії IBM. Ця модель зазнала невдачі, не дивлячись на те, що послідуючі реалізації були успішними.Так вийшло, тому що традиційні обчислення також бурхливо розвивалися в той час, на їх тлі ШНМ не виглядали так перспективно.

1957 г. - Ф. Розенблатт (F. Rosenblatt) вперше в світі пропонує технічну реалізацію нейрокомп’ютера. Заклав постулати роботи та організації перцептронів.

1958 р Джон фон Нейман (John fon Neumann) розробив систему на основі вакуумних трубок, що імітує прості функції нейронів.

1959 р Бернард Відроу (Bernard Widrow) і Марсіан Хофф (Marcian Hoff) розробили моделі ADALINE (Multiple ADAptive LINear Elements, Адаптивні Лінійні Елементи) і MADALINE (Множинні Адаптивні Лінійні Елементи).

MADALINE використовувалася в якості адаптивного фільтра для усунення перешкод на телефонних лініях. Ця нейромережа використовується і до сих пір.

У тому ж році нейробіолог Френк Розенблат (Frank Rosenblatt) почав роботу над моделлю перцептрону. Одношаровий перцептрон, побудований Розенблаттом, в даний час вважається класичною моделлю нейромережі. Розенблатт використовував свій перцептрон, щоб розділяти вхідні сигнали на два класи. На жаль, одношаровий перцептрон міг виконувати лише обмежений клас задач [1].

1969 р Опублікована книга М. Мінського (M. Minsky) і С. Пейперта (S. Papert) «Перцептрони», в якій доводилася принципова обмеженість можливостей перцептронів [2].

Ранні успіхи сприяли тому, що від нейронних мереж стали очікувати занадто багато: більше, ніж дійсно можна було реалізувати в рамках тієї моделі. Надмірний оптимізм, процвітаючий в академічному і технічному співтоваристві, врешті-решт привів безліч фахівців до розчарування, і всі проекти дослідження нейронних мереж були піддані критиці як безперспективні.

В результаті, фінансування досліджень було припинено аж до 80-х років. 1970-1976 рр. В СРСР проводилися активні розробки в області перцептронів.

80-і роки. Завдяки роботам Джона Хопфилда (John Hopfield), відродився інтерес до нейроінформатики. Хопфілд показав, як подолати обмеження нейромереж першого покоління, а також розробив теорію нейронних мереж, які моделюють асоціативну пам'ять.

1985 р З'явилися перші комерційні нейрокомп'ютери, наприклад, Mark III фірми TRW (США).

1986 р Девід Румельхарт з співавторами запропонував алгоритм тренування багатошарового перцептрона. У той же час в Кіото (Японія) відбулася об'єднана американо-японська конференція по нейронним

мережам. На цій конференції нейронні мережі оголосили п'ятим поколінням ЕОМ.

З 1985 р Американський Інститут Фізики почав щорічні зустрічі «Нейронні мережі для обчислень».

1987 р Почалося масштабне фінансування розробок в області штучних нейронних мереж в Японії, США та Західній Європі (японська програма «Human Frontiers» і європейська програма «Basic Research in Adaptive Intelligence and Neurocomputing») [2].

1989 р. Розробки і дослідження в області штучних нейромереж ведуться практично всіма великими електротехнічними фірмами.

Нейрокомп'ютери стають одним з найдинамічніших секторів ринку (за два роки обсяг продажів виріс в п'ять разів). Агентством DARPA (Defence Advanced Research Projects Agency) міністерства оборони США розпочато фінансування програми по створенню надшвидкодіючих зразків нейрокомп'ютерів для різноманітних застосувань.

1990 р. Активізація радянських наукових організацій в галузі дослідження штучних нейронних мереж і нейрокомп'ютерів (Інститут кібернетики ім. Глушкова в Києві, Інститут багатопроцесорних обчислювальних систем в Таганрозі, Інститут нейрокібернетики в Ростові-наДону) [3].

1991 р Річний обсяг продажів на ринку нейрокомпьютерів наблизився до 140 млн. Доларів. Були створені центри дослідження нейрокомп'ютерів в Москві, Києві, Мінську, Новосибірську, Санкт-Петербурзі.

1992 р. Роботи в області нейромереж знаходяться в стадії інтенсивного розвитку.

Щорічно проводяться десятки міжнародних конференцій і форумів по нейронним мережам, число спеціалізованих періодичних наукових видань із зазначеної тематики досягає двох десятків найменувань.

1997 р. Річний обсяг продажів на ринку штучних нейромереж і нейрокомп'ютерів перевищив 9 млрд. Доларів, а щорічний приріст склав 50%.

2000 р. Завдяки переходу на субмікронні і нанотехнології, а також успіхам молекулярної і біомолекулярної технології, з'являються принципово нові архітектурні та технологічні підходи до створення нейрокомп'ютерів.

Побудова будь-якої нейронної мережі передбачає великий обсяг обчислень (тренування мережі зазвичай є ітераційним процесом). Тому тільки з ростом обчислювальної потужності комп'ютерів з'явилася можливість практичного застосування нейронних мереж, що дало потужний поштовх до широкого поширення програм, що використовують принципи нейромережевої обробки даних.

Дослідники досі не дійшли єдиної думки щодо визначення нейронної мережі. У літературі можна зустріти безліч варіантів, наприклад:

«Нейронна мережа - система, що складається з безлічі простих обчислювальних елементів, що працюють паралельно. Результат роботи мережі визначається структурою мережі, силою зв'язків, а також видом обчислень, виконуваних кожним елементом »[4].

«Нейронна мережа - паралельний розподілений процесор, здатний самостійно отримувати дані з інформації, що надходить. Робота такої мережі нагадує роботу мозку, так як знання отримуються за допомогою процесу тренування, а отримані знання зберігаються не в окремому елементі, а розподілені по всій мережі »[5].

«Нейронна мережа - система, що складається з великого числа простих обчислювальних елементів. Результат роботи кожного елемента залежить тільки від його внутрішнього стану. Всі елементи працюють незалежно один від одного, тобто без синхронізації з іншими елементами »[6].

«Штучні нейронні мережі - системи, здатні отримувати, зберігати і використовувати знання» [7].

Проте, більшість дослідників сходяться на тому, що нейронна мережа - це система, що складається з безлічі простих процесорів, кожна з яких має локальну пам'ять. Вміст такої пам'яті прийнято називати станом процесора. Процесори здатні обмінюватися між собою числовими даними. Результат роботи процесора залежить тільки від його стану і даних, які він отримує на вході.

1.1 Загальні поняття про біологічний нейрон і нейронні мережі

Штучні нейронні мережі (ШНМ) є одним з найбільш поширених методів класифікації в інтелектуальному аналізі даних та потужним методом прогнозування. Такі системи навчаються розв’язувати задачі, розглядаючи приклади загалом, без спеціального програмування під конкретну задачу, не маючи апріорних знань про неї.

Штучні нейронні мережі мають дві сильні сторони – розпаралелювання обробки інформації та задатність самонавчатися, тобто створювати абстракції.

Під терміном астаркції розуміється здатність створювати прийнятний результат на основі даних, які в процесі навчання не були використані. Такі особливості дозволяють нейронним мережам розв’язувати масштабні завдання, які поки вважаються важковирішуваними. Не дивлячись на це, штучні нейронні мережі самі по собі не можуть надавати готові рішення, а потребують інтеграції, в свою чергу, до складніших систем. Наприклад комплексні задачі можна розбивати на більш прості, які будуть прийнятно вирішуватися нейронними мережами. Важливо розуміти, що для створення архітектури, яка буде розв’язувати комплексні завдання на рівні з людським мозком ще необхідно пройти важкий та довгий шлях.

Однак, штучні нейронні мережі володіють такими корисними особливостями:

1. Очевидність відповіді. Якщо розглядати задачу класифікації, то нейронна мережа може бути розроблена таким чином, щоб надавати

інформацію не тільки про клас вхідних даних, а і про достовірність результату. Така інформація надасть змогу не брати до уваги сумнівні рішення нейронної мережі.

2. Нелінійність. Оскільки функція активації кожного нейрону часто вибирається нелінійною, то мережа вцілому має змогу імітувати набагато складніші нелінійні функції. Це дозволяє ШНМ апроксимувати функції майже будь якої складності. Обмеження накладає тільки кількість нейронів та структура зв’язків мережі.

3. Чітка відповідність вхідної інформації до вихідної.

Тренування ШНМ з учителем є одним з найпопулярніших способів тренування. Цей спосіб заключається в зміні вагових коефіцієнтів використовуючи навчальні приклади. Для кожного вхідного прикладу існує бажаний вихідний вектор, відповідний йому. При тренуванні наступний приклад вибираєтсья випадковим чином та нейронна мережа формує вихідний вектор на основі прикладу. Цей вектор порівнюється з бажаним, та за спеціальним алгоритмом відбувається процес модифікації вагових коефіцієнтів, таким чином, щоб наблизитись до бажаного вектору. Приклади, що були використані раніше, можуть бути застосовані в іншому порядку ще раз.

4. Адаптивність. Нейронні мережі навчаються гнучко вирішувати поставлені завдання. Зокрема, при необхідності, вони можуть бути перенавчені досить легко, якщо коливання параметрів середовища незначні. Також висока адаптивність дозволяє надійно використовуати нейронні мережі в нестаціонарних середовищах. Хоча, слід зауважити, що висока адаптивність може призводити до нестійкої поведінки ШНМ. Якщо при зміні коефіцієнтів вона швидко пристосовується до певних подразників, то це може істотно влинути на подальшу продуктивність. Тобто висока адаптивність може призводити до втрати стійкості і навпаки. Зазвичай це називають проблемою пластичності-стійкості.

5. Відмовостійкість. Добре тренована нейронна мережа може нормально працювати, навіть якщо кілька її нейронів вийшли з ладу. Це дозволяє реалізовувати нейронні мережі на фізичному рівні, та бути впевненим в надійності вибраної архітектури. Оскільки інформацію, необхідну для функціонування мережі розподілено в кожному нейроні, то вся мережа може вийти з ладу тільки за умови серйозного пошкодження структури, а незначне пошкодження ніколи серйозно не вплине на працездатність. Це очевидна перевага ШНМ.

6. Здатність до масштабування. Архітектура нейронних мереж дозволяє дуже легко виконувати великий відсоток обчислень паралельно. Це дозволяє витрачати мінімальну кількіст зусиль на перенесення чи масштабування систем, основаних на штучних нейронних мережах.

ШНМ складається з базових модулів – штучних нейронів. Вони моделюють основні функції природних нейронів. В процесі роботи мережі, багато вхідних сигналів одночасно надходять до нейрону. Кожен вхід має свій власний синаптичний ваговий коефіцієнт, який надає входу вплив, що враховується в функції суматора елемента обробки. Ці коефіцієнти моделюють різну пропускну здатність для сигналів в біологічній нейромережі та є мірою впливу одного нейрону на іншого. Сигнал від нейрону, ваговий коефіцієнт якого значний, суттєво підсилюється, і навпаки, відбувається примусове зменшення сигналу від нейрону, якщо його відповідний ваговий воефіцієнт замалий. Розміри вагових коефіцієнтів можуть змінюватись, залежно від архітектури мережі, спеціальних правил тренування чи різноманітних навчальних прикладів.

Вхідні сигнали x_n зважені ваговими коефіцієнтами з'єднання w_n додаються, проходять через передатну функцію, результат якої потрапляє до наступного нейрону мережі, чи безпосередньо на вихід.

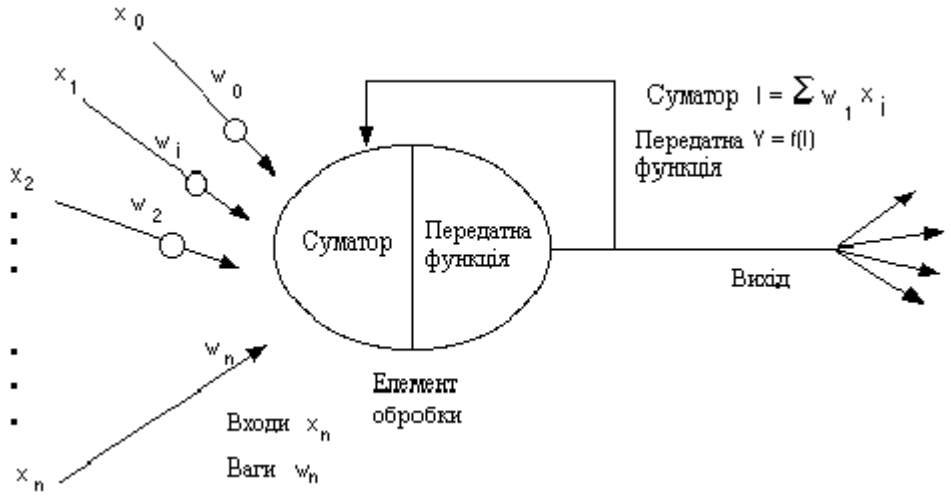


Рисунок 1.1 – Базовий штучний нейрон

В програмних реалізаціях штучні нейрони називають «елементами обробки» або «процесорами» і вкладають в них більше можливостей, ніж в базовому штучному нейроні, що описаний вище.

На рисунку 1.2 схема штучного нейрону зображається детальніше.

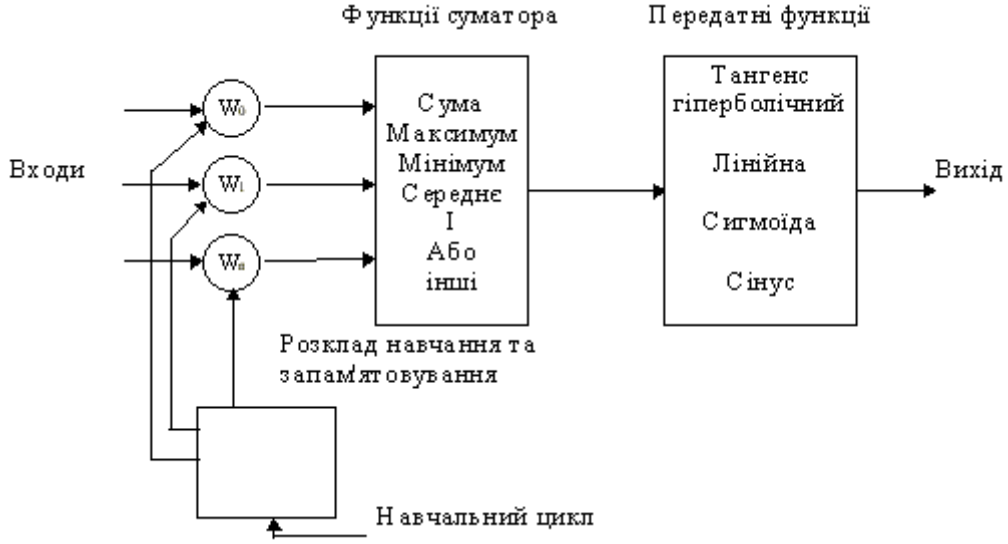


Рисунок 1.2 – Модель "елементу обробки"

Замість функції, що обробляє вхідні сигнали, нейрон може мати і інші функції. Наприклад можна вибирати мінімальний чи максимальний сигнал або знайти середнє арифметичне чи добуток всіх сигналів. В деяких випадках обробкою вхідних сигналів займається окремий алгоритм. Багато програмних реалізацій використовують власні функції суматора, що запрограмовані на мові вищого рівня.

Штучний нейрон (базовий процесорний елемент) - є основою будь-якої штучної нейронної мережі.

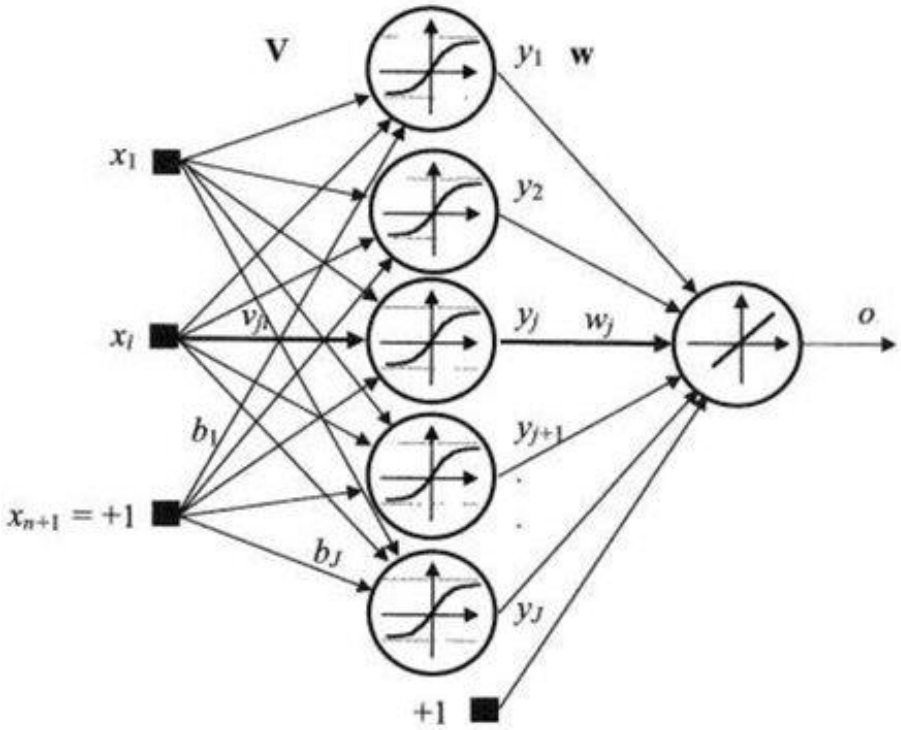


Рисунок 1.3 – Схематичне зображення штучного нейрона

Нейрони являють собою відносно прості, однотипні елементи, що імітують роботу нейронів мозку. Кожен нейрон характеризується своїм поточним станом (збуджена або загальмована) за аналогією з нервовими клітинами головного мозку.

Штучний нейрон, також як і його природний прототип, має групу синапсів (входів), які з'єднані з виходами інших нейронів, а також аксон - вихідний зв'язок даного нейрона - звідки сигнал збудження або гальмування надходить на синапси інших нейронів.

Загальний вигляд нейрона показаний на рисунку 1.1. Штучний нейрон складається з двох елементів - зваженого суматора і нелінійного перетворювача.

На вхід штучного нейрона надходить деяка множина сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід множиться на відповідну вагу, аналогічну синаптичній силі, і всі добутки підсумовуються, визначаючи

рівень активації нейрона. Зважений суматор виробляє підсумовування за формулою:

$$\mathbf{S}_{jl} = \sum_{i=1}^{n} X_i W_{ijl} + W_{0jl}, \quad (1.1)$$

де i - номер входу нейрона;

j - номер нейрона в шарі;

l – номер шару

X_i – вхідні сигнали, сукупність всіх вхідних сигналів нейрона, що утворюють вектор $\mathbf{X}$.

W_i – вагові коефіцієнти, сукупність вагових коефіцієнтів, що утворюють вектор ваг $\mathbf{W}$.

W_0 - вага, що моделює пороговий рівень нейрона

$\mathbf{S}$ – зважена сума вхідних сигналів, значення S – передається на нелінійний елемент.

$$\mathbf{S} = \mathbf{X}^T \mathbf{W}, \quad (1.2)$$

де $\mathbf{X}$ - вектор вхідних сигналів нейрона, що подається на вхід з вагою W_0;

$\mathbf{W}$ - вектор ваг нейрона, що включає $\mathbf{W_0}$.

Нелінійний елемент перетворює вихід суматора за формулою:

$$\mathbf{Y}_{jl} = f(\mathbf{S}_{jl}), \quad (1.3)$$

де f – функція активації, яка підбирається специфікою розв'язуваної задачі, зручністю реалізації нейронної мережі і алгоритмом тренування.

1.2 Функція активації нейрона

Перед надходженням до функції активації вхідні сигнали та вагові коефіцієнти можуть комбінуватись багатьма способами. Алгоритми для комбінування входів нейронів визначають відповідно до мережної архітектури та парадигми.

В деяких нейромережах суматор виконує додаткову обробку, так звану функцію активації, яка зміщує вихід функції суматора в часі. Цю функцію найкраще використовувати як компоненту мережі в цілому, ніж як компоненту окремого нейрона. Часто, ця функція є відсутньою.

Результат функції суматора перетворюється у вихідний сигнал через функцію активації. В функції активації для визначення виходу нейрона загальна сума порівнюється з деяким порогом (зазвичай, це діапазон [0, 1] або [-1,1] або інше) за допомогою певного алгоритму.

Переважно застосовують нелінійну функцію активації, оскільки лінійні функції є обмеженими і вихід є пропорційним до входу. Застосування лінійних функцій активації було проблемою у ранніх моделях мереж, і їх обмеженість та недоцільність була доведена в книзі Мінскі та Пейперта "Перцептрони"[2].

В існуючих нейромережах як функцію активації використовують сигмоїду, синус, гіперболічний тангенс тощо.

Для різних нейромереж можуть вибиратись інші передатні функції. Після обробки сигналу, нейрон на виході має результат передатної функції, який надходить на входи інших нейронів або до зовнішнього з'єднання, як це передбачається структурою нейромережі.

Функція активації (активаційна функція, функція збудження) - функція, що обчислює вихідний сигнал штучного нейрона.

Як аргумент приймає сигнал Y, одержуваний на виході суматора Σ.

Існує багато різних активаційних функцій, але частіше за все використовуються наступні:

Порогова функція або одиничний стрибок (рисунок 1.4) - кусково-лінійна функція. Якщо вхідне значення менше порогового, то значення функції активації мінімальне, в іншому випадку – максимальне.

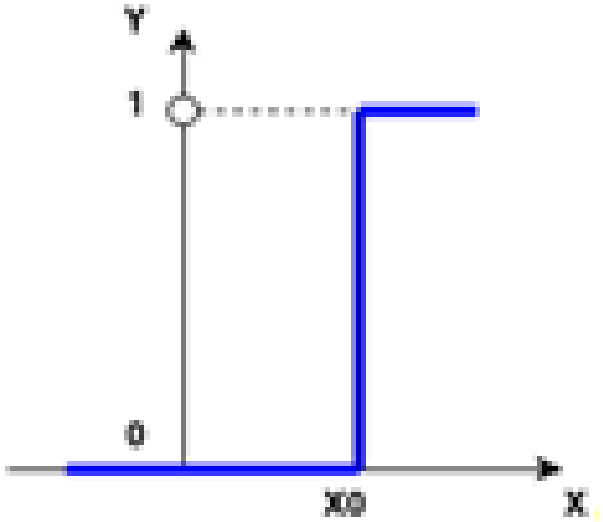


Рисунок 1.4 - Графік порогової функції

Порогова функція описується так:

$$Y(X) = \begin{cases} 0, X \le 0 \\ 1, X > 0 \end{cases}, \tag{1.7}$$

Лінійний поріг або гістерезис (рисунок 1.5) - нескладна кусочно-лінійна функція, яка має дві лінійних ділянки, де функція активації тотожно дорівнює мінімально допустимому і максимально допустимого значенню і є ділянка, на якому функція строго монотонно зростає.

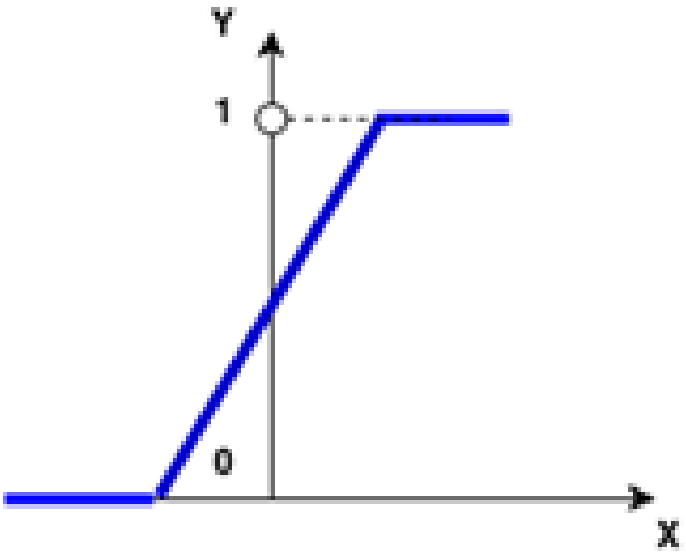


Рисунок 1.5 - Графік функції лінійного порогу

Сигмоїдальна функція (рисунок 1.6) - монотонно зростаюча всюди диференційована S – образна нелінійна функція з насиченням.

Сигмоїдальна функція, є найпоширенішою функцією, яка використовується для створення нейронних мереж.

Це швидко зростаюча функція, яка підтримує баланс між лінійною і нелінійним поведінкою. Часто під сигмоїдальною функцією розуміють логістичну функцію.

Функцію можна описати таким виразом:

$$Y = \frac{1}{1 + \exp(-aY)}, \tag{1.8}$$

де, α – параметр нахилу. Змінюючи цей параметр, можна побудувати функції з різною крутизною.

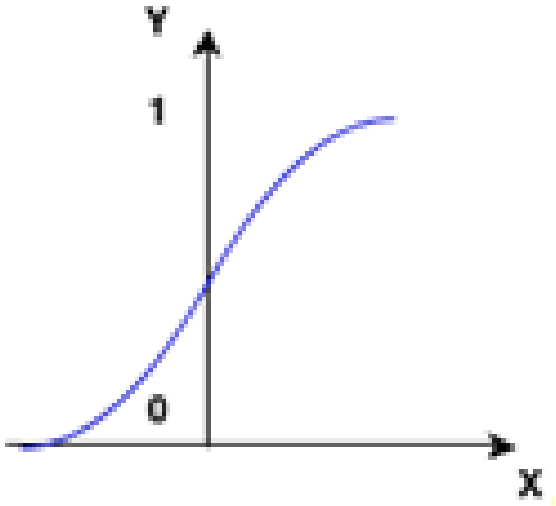


Рисунок 1.6 - Графік сигмоїда

Гіперболічний тангенс (рисунок 1.7). Дану функцію можна представити виразом:

$$Y = th(\frac{Y}{\alpha}), \tag{1.9}$$

де, α – параметр, що впливає на нахил сигмоїдальної функції.

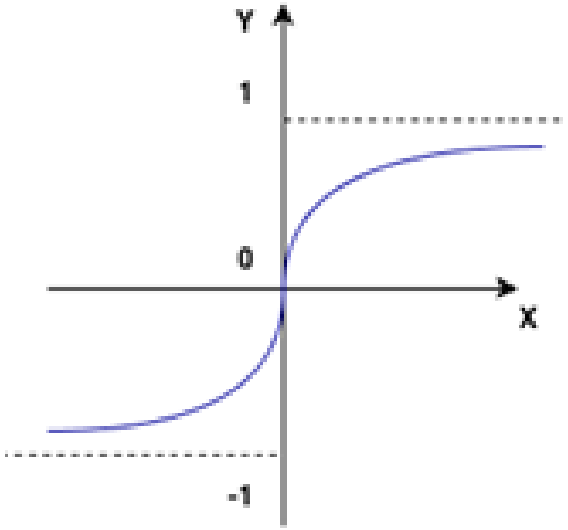


Рисунок 1.7 - Графік гіперболічного тангенса

1.3 Архітектура нейронних мереж

Штучний нейрон – є базовим блоком штучної нейроннї мережі. Велика кількість синаптичних звязків пов’язуть штучні нейрони в одну мережу. Групування нейронів у мозку людини забезпечує обробку інформації інтерактивним та динамічним шляхом.

Біологічні нейронні мережі з мікроскопічних компонентів існують у тривимірному просторі і здатні до різноманітних з'єднань. Але для реалізації штучних мереж присутні фізичні обмеження.

Штучні нейрони можуть утворювати ефективні системи обробки інформації, з’єднуючись у мережі. Вони забезпечують ефективну адаптацію моделі до постійних змін з боку зовнішнього середовища. В процесі роботи мережі відбувається конвертація вхідного вектора сигналів у вихідний. Конкретний вид конвертації визначається архітектурою ШНМ, характеристиками нейронів, засобами синхронізації та керування інформаційних потоків між нейронами.

Встановлення типів зв’язків між нейронами, та оптимальної кількості нейронів є важливою задачею для досягння ефективності мережі.

Для характреристики нейромереж можуть бути використані такі терміни:

- **Структура нейромережі** - спосіб з’єднання нейронів у нейромережі.
- **Архітектура нейромережі** - типи нейронів у нейромережі, та її структура.
- **Парадигма нейромережі** - спосіб тренування та використання.

Різні парадигми можуть бути реалізовані на базі однієї архітекури, і навпаки.

Серед відомих архітектурних рішень виділяють групу **слабозв'язаних нейронних мереж**, у випадку, коли кожний нейрон мережі зв'язаний лише із сусідніми. В **повнозв'язаних нейромережах** входи кожного нейрона зв'язані з виходами всіх решта нейронів.

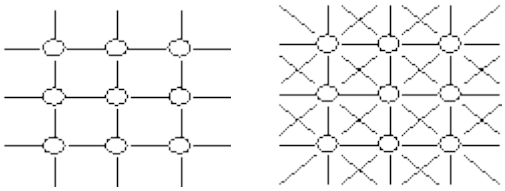


Рисунок 1.8 – Слабозв'язані нейромережі

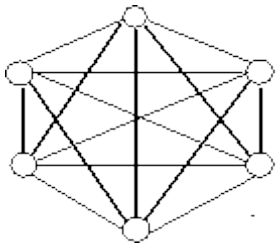


Рисунок 1.9 – Повнозв'язані нейромережі

Найбільш поширеним варіантом архітектури є багатошарові мережі. Нейрони в даному випадку об'єднуються у шари з єдиним вектором вхідних сигналів. Зовнішній вхідний вектор подається на вхідний шар нейронної мережі (рецептори). Виходами нейронної мережі є вихідні сигнали останнього шар (ефектори). Окрім вхідного та вихідного шарів, нейромережа має один або кілька прихованих шарів нейронів, які не мають контактів із зовнішнім середовищем.

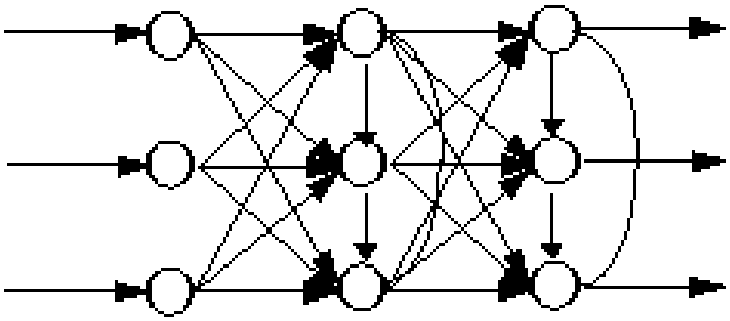


Рисунок 1.10 – Багатошаровий тип з'єднання нейронів

- Зв'язки між нейронами різних шарів називають **проективними**.
- Зв'язки між нейронами одного шару називають **бічними** (латеральними).

На рисунку 1.10 показана типова структура штучних нейромереж. Хоча існують мережі, які містять лише один шар, або навіть один елемент, більшість застосувань вимагають мережі, які містять як мінімум три типи шарів - вхідний, прихований та вихідний. Вхідний шар нейронів отримує безпосередньо дані, що розпізнаються чи класифікуються. Ці данні у деяких випадках можуть бути попередньо нормалізованими, задля прискорення тренування нейронної мережі. Вихідний шар пересилає інформацію безпосередньо до зовнішнього середовища, до вторинного комп'ютерного процесу, або до інших пристроїв. Між цими двома шарами може бути багато прихованих шарів, які містять багато нейронів в різноманітних зв'язаних структурах. Входи та виходи кожного з прихованих нейронів сполучені з іншими нейронами.

Важливим аспектом нейромереж є напрямок зв'язку від одного нейрону до іншого:

- Зв'язки скеровані від вхідних шарів до вихідних називаються **аферентними**,
- Зв’язки в зворотному напрямку називаються **еферентними**.

В більшості мереж кожен нейрон прихованого шару отримує сигнали від всіх нейронів попереднього шару чи від нейронів вхідного шару. Після виконання операцій над сигналами, нейрон передає свій вихід до всіх нейронів наступних шарів, забезпечуючи передачу вперед (*feedforward*) на вихід.

При зворотному зв'язку, вихід нейронів скеровується до нейронів попереднього шару (рисунок 1.11).

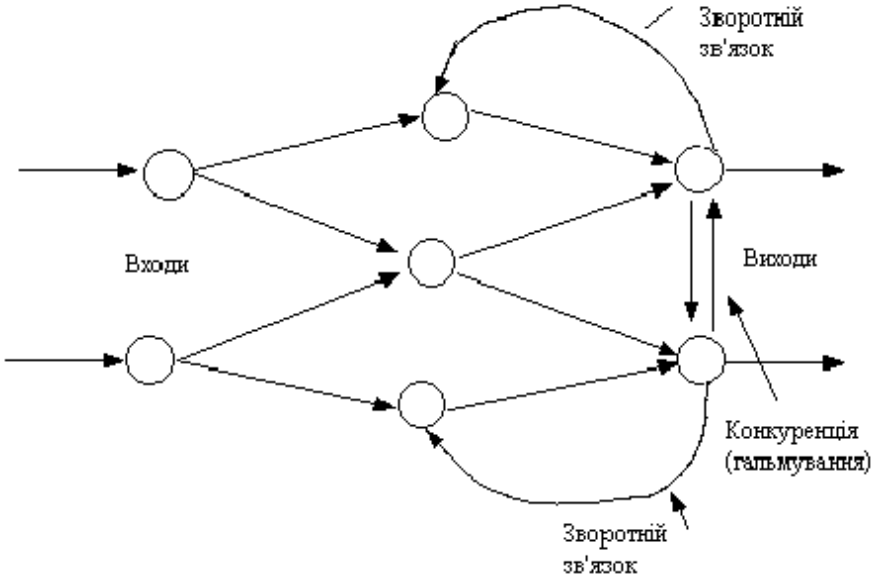


Рисунок 1.11 – ШНМ зі зворотнім зв’язком

Напрямок зв’язків нейронів має значний вплив на роботу мережі. Більшість програмних нейромереж дозволяють користувачу додавати, вилучати та керувати з'єднаннями як завгодно. Корегуючи параметри, можна налаштувати зв'язки як на посилення так і на послаблення величини сигналів.

За архітектурою зв'язків, більшість відомих нейромереж можна згрупувати у два великих класи:

1. Мережі прямого поширення (з односкерованими послідовними зв'язками).
2. Мережі зворотного поширення (з рекурентними зв'язками).

Мережі прямого поширення відносять до статичних, тут на входи нейронів надходять вхідні сигнали, які не залежать від попереднього стану мережі.

Рекурентні мережі вважаються динамічними, оскільки за рахунок зворотних зв'язків (петель) входи нейронів модифікуються в часі, що призводить до зміни станів мережі.

Типові архітектури нейронних мереж:

1. Рекурентні мережі:
    a. Мережа Хемінга

b. Мережа Хопфілда
c. Двоскерована асоціативна пам'ять
d. Мережа адаптивної резонансної теорії

2. Мережі прямого поширення:
    a. Карта Кохонена
    b. Перцептрони
    c. Мережа зустрічного поширення
    d. Мережа Back Propagation

За архітектурою зв'язків ШНМ можуть бути згруповані в два класи: мережі прямого поширення, і рекурентні мережі (зі зворотними зв'язками).

У мережах прямого поширення сигнал по мережі проходить тільки в одному напрямку: від входу до виходу.

Сукупність нейронів, об'єднаних в один шар, називають одношаровою нейромережею. Об'єднання одношарових нейромереж в кілька шарів визначає багатошарову нейромережу.

Розглянемо одношаровий персептрон.

На вхід надходить тільки двійковий сигнал, тобто або 0 або 1. Елемент Σ (суматор) складає вхідні сигнали, зважені щодо відповідних синапсів нейрона.

Якщо ця сума більше заданого порогового значення, вихід дорівнює 1, якщо менше - нулю. Оскільки вихідний сигнал *y* приймає значення «0» або «1», тому персептрон - це нейрон бінарного типу. При функціонуванні персептрона розрізняють два режими його роботи: режим тренування, режим функціонування.

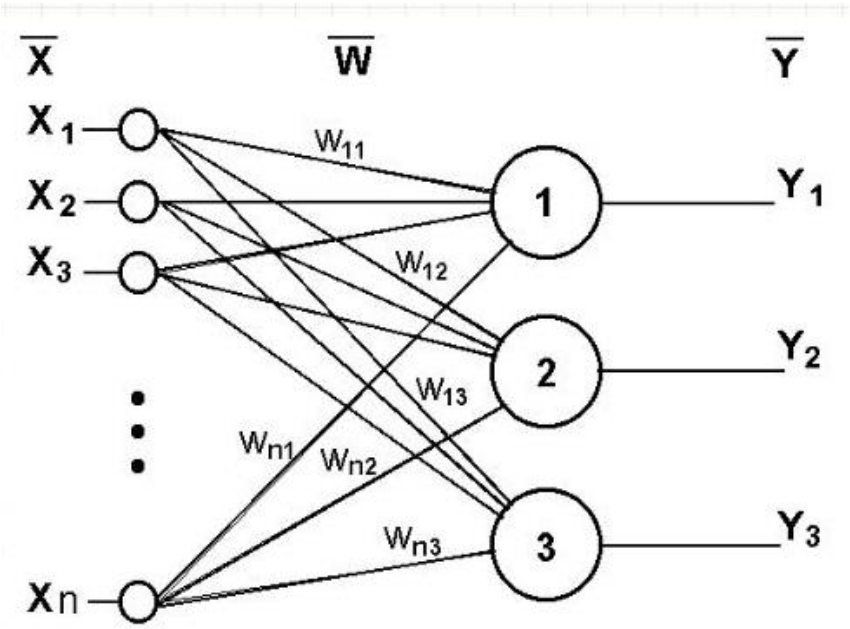


Рисунок 1.12 - Структура одношарового персептрона

Модель персептрона, представлена на рисунку 1.12 складається з одного шару (тобто кількість шарів нейронів між входом X і виходом Y дорівнює одиниці) штучних нейронів, з'єднаних за допомогою вагових коефіцієнтів з множинию входів.

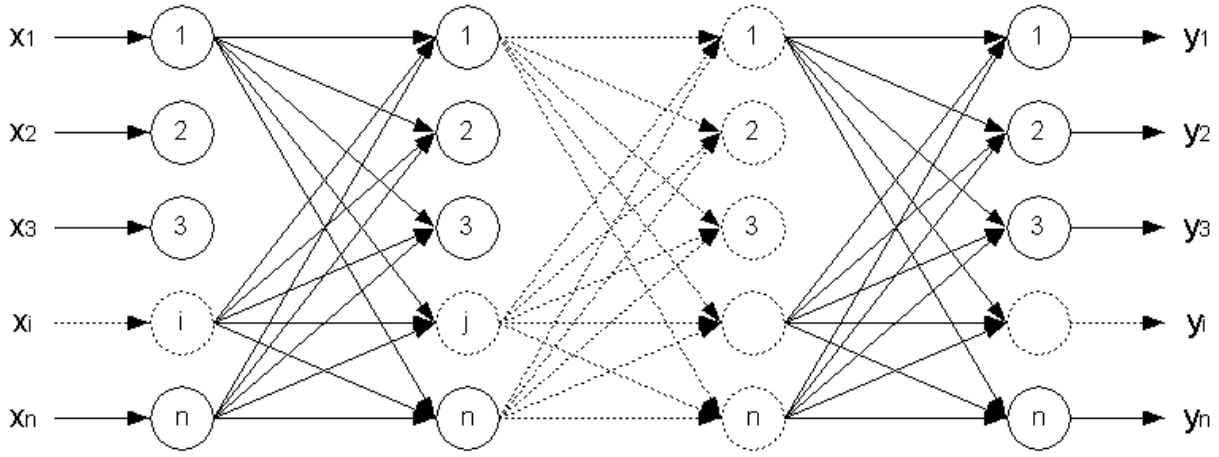


Рисунок 1.13 - Структура багатошарового персептрона

Багатошарова мережа складається з нейронів, які розташовані на різних рівнях. Крім вхідного і вихідного шарів є як мінімум ще один шар - прихований. При завданні функції активації сигмоїдального типу матимемо багатошаровий персептрон (рисунок 1.13).

У багатошаровому персептроні кожен нейрон на даному рівні ієрархії приймає і обробляє сигнали від кожного нейрона нижчого рівня.

Прихований шар нейронів дозволяє мережі навчатися вирішенню складних завдань, послідовно отримуючи найбільш важливі ознаки з вхідного шару.

Багатошарові персептрони виявилися вельми ефективними для вирішення різноманітних завдань управління.

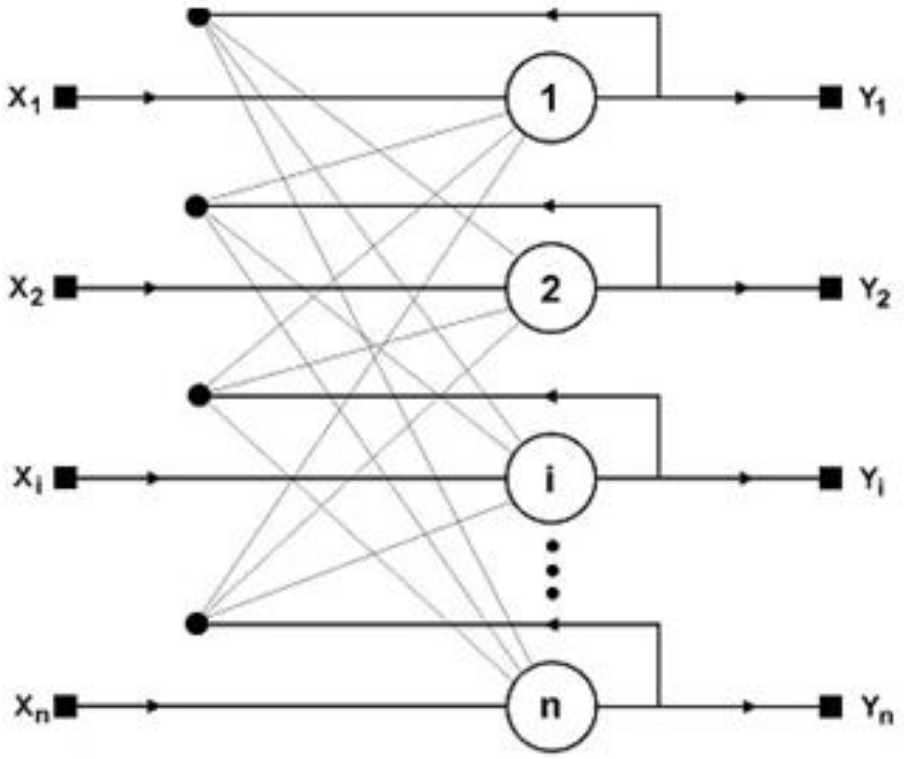


Рисунок 1.14 - Структура мережі Хопфілда

Нейронні мережі, що мають зворотний зв'язок, називаються "рекурентними мережами".

Кожен крок мережі називається ітерацією. Рекурентна мережа може складатися з єдиного шару нейронів, кожен з яких спрямовує свій вихідний сигнал на входи всіх інших нейронів шару. Наприклад, мережа Хопфілда, представлена на рисунку 1.14, не має прихованих нейронів.

Варто відзначити, що на малюнку видно, що відсутні зворотні зв'язки нейронів з самими собою.

На малюнку нижче (рисунок 1.15) представлений інший клас рекурентних мереж з прихованими шарами нейронів. Тут зворотні зв'язки виходять, як з вихідних, так і з прихованих нейронів.

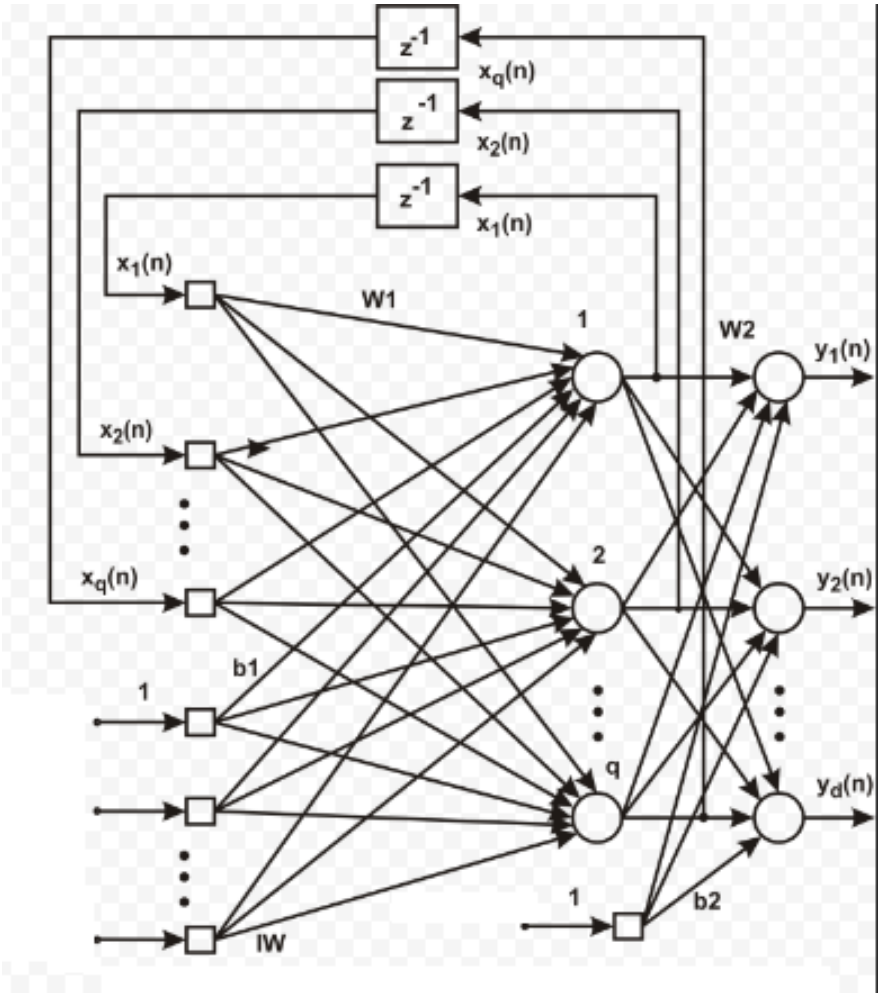


Рисунок 1.15 - Структура рекурентної мережі з прихованими нейронами

Наявність зворотного зв'язку в мережі, показаній на малюнку 1.15, безпосередньо впливає на продуктивність і здатність мережі до тренування.

Більш того, зворотний зв'язок має на увазі використання елементів одиничної затримки (позначені як z-1), що призводить до нелінійної динамічної поведінки, якщо, звичайно, в мережі містяться нелінійні нейрони.

1.4 Відомості про тренування нейронних мереж

Оригінальність нейромереж полягає у здібності до тренування за прикладами, що складають навчальну множину, аналогічно біологічному мозку. Процес тренування нейромереж розглядається як визначення архітектури та вибір вагових коефіцієнтів синаптичних зв'язків, які б відповідали даним навчальної множини, щоб ефективно вирішувати поставлену задачу.

Тренування нейромереж буває:

- Контрольоване тренування (тренування з вчителем)
- Неконтрольоване тренування (тренування без вчителя)

Контрольоване тренування

Контрольоване тренування використовується в абсолютній більшості реалізацій нейромереж. При такомунавчанні вихід ШНМ постійно порівнюється з бажаним виходом. Вагові коефіцієнти зв’язків на початку встановлюються за певним псевдо випадковим правилом (ініціалізація мережі). Під час тренування, для досягнення максимально близької відповідності між виходом ШНМ та бажаним виходом, вагові коефіцієнти та коефіцієнти зміщення коректуються. Такий метод тренування націлений на мінімізацію похибок всіх елементів обробки, що вібувається завдяки неперервній зміні синаптичних ваг до досягнення прийнятної точності мережі.

Перед використанням, нейромережа з контрольованим тренуванням повинна бути навченою. Фаза тренування займає певний час. Тренування вважається закінченим при досягненні нейромережею визначеного користувачем рівня ефективності і бажаної статистичної точності. Після тренування вагові коефіцієнти зв’язків фіксуються для подальшого застосування. Деякі типи мереж дозволяють під час використання

продовжувати тренування, і це допомагає мережі адаптуватись до змінних умов.

Навчальні множини повинні бути достатньо великими, щоб містити всю необхідну інформацію для виявлення важливих особливостей і зв'язків. Навчальні приклади повинні містити широке різноманіття даних. Якщо мережа навчається лише для одного прикладу, вагові коефіцієнти, що старанно встановлено для цього прикладу, радикально змінюються у навчанні для наступного прикладу. Попередні приклади при навчанні наступних просто забуваються. В результаті система повинна навчатись всьому разом, знаходячи найкращі вагові коефіцієнти для загальної множини прикладів.

Наприклад, у навчанні системи розпізнавання піксельних образів для десяти цифр, які представлені двадцятьма прикладами кожної цифри, всі приклади цифри "сім" не доцільно представляти послідовно. Краще надати мережі спочатку один тип представлення всіх цифр, потім другий тип і так далі.

Головною компонентою для успішної роботи мережі є представлення і кодування вхідних і вихідних даних. Штучні мережі працюють лише з числовими вхідними даними, отже, необроблені дані, що надходять із зовнішнього середовища повинні перетворюватись. Важливою є нормалізація даних, тобто приведення всіх значень даних до єдиного діапазону. Нормалізація виконується шляхом ділення кожної компоненти вхідного вектора на довжину вектора, що перетворює вхідний вектор в одиничний. Попередня обробка зовнішніх даних, отриманих за допомогою сенсорів, у машинний формат є спільною і легко доступною для стандартних комп'ютерів.

Якщо після контрольованого тренування нейромережа ефективно опрацьовує дані навчальної множини, важливим стає її ефективність при роботі з даними, які не використовувались для тренування. У випадку отримання незадовільних результатів для тестової множини, тренування

продовжується. Тестування використовується для забезпечення запам'ятовування не лише даних заданої навчальної множини, але і створення загальних образів, що можуть міститись в даних.

Неконтрольоване тренування

Неконтрольоване тренування може бути великим надбанням у майбутньому. Воно проголошує, що комп'ютери можуть самонавчатись у справжньому роботизованому сенсі. На даний час, неконтрольоване тренування використовується в мережах відомих, як самоорганізовані карти (self organizing maps). Мережі не використовують зовнішніх впливів для коректування своїх ваг і внутрішньо контролюють свою ефективність, шукаючи регулярність або тенденції у вхідних сигналах та здійснюють адаптацію відповідно до навчальної функції. Навіть без повідомлення правильності чи неправильності дій, мережа повинна мати інформацію відносно власної організації, яка закладена у топологію мережі та навчальні правила.

Алгоритм неконтрольованого тренування скеровано на знаходження близькості між групами нейронів, які працюють разом. Якщо зовнішній сигнал активує будь-який вузол в групі нейронів, дія всієї групи в цілому збільшується. Аналогічно, якщо зовнішній сигнал в групі зменшується, це приводить до гальмуючого ефекту на всю групу.

Основу для тренування формує конкуренція між нейронами. Тренування конкуруючих нейронів підсилює відгуки певних груп на певні сигнали. Це пов'язує групи між собою та відгуком. При конкуренції змінюються ваги лише нейрона-переможця.

Оцінки тренування

Оцінка ефективності тренування нейромережі залежить від кількох керованих факторів, важливими з яких є: ємність, складність зразків і обчислювальна складність.

- **Ємність** показує, скільки зразків може запам'ятати мережа, і які межі прийняття рішень можуть бути на ній сформовані.
- **Складність зразків** визначає кількість навчальних прикладів, необхідних для досягнення здатності мережі до узагальнення.
- **Обчислювальна складність** напряму пов'язана з потужністю комп’ютера.

1.5 Багатошаровий перцептрон та опис методу зворотнього поширенн помилки

Багатошаровий перцептрон є найбільш поширеною моделлю для тренування мереж прямого поширення. Основна мета мережі прямого поширення – апроксимувати деяку функцію f. Наприклад для задачі класифікації ця функція перетворює вхдні данні в категорію, до якої, імовірно, ці вхідні данні належать. Мережа прямого поширення визначає відображення $y = f(x, \theta)$. В процесі тренування ШНМ знаходить таке значення θ, щоб максимально близько апроксимувати цю функцію. Ці мережі називаються мережами прямого поширення, оскільки при їх тренуванні та використанні сигнал проходить з вхідного шару, через кілька прихованих шарів, відразу до вихідного шару. Тобто у данної моделі немає зворотніх зв’язків, в яких виходи певних нейронів повертаються до попередніх шарів.

На рисунку 1.16, ми можемо побачити, яким чином багатошаровий перцептрон може подолати проблему нелінійності функції, яку необхідно апроксимувати в процесі тренування.

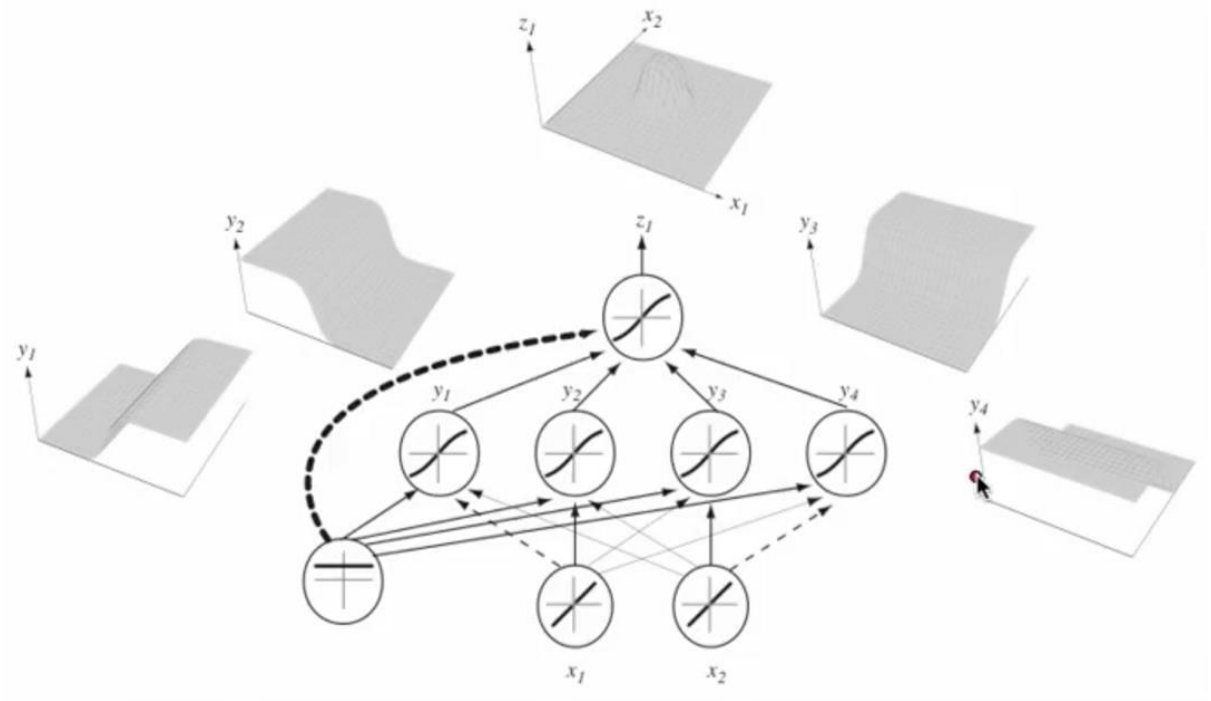


Рисунок 1.16 – Комбінація функцій різних нейронів в результуючу функцію

Одним з найпоширеніших методів тренування багатошарового перцептрона є метод зворотного поширення помилки (МЗПП)(англ. backpropagation). Цей алгоритм використовується для мінімізації функції помилки ШНМ та отримання бажаного результату на виході. Це градієнтний ітеративний алгоритм. У процесі тренування мережі сигнали помилки поширюються від віходів ШНМ до її входів. В зворотньому напрямку, у порівнянні з поширенням сигналів в звичайному режимі роботи. Матод зворотнього поширення помилки вимагає, щоб функція активації нейронів була диференційовною на всій області визначення.

Тренування ШНМ є типовою задачею оптимізації. Процес тренування ШНМ можна представити як процес мінімізації функції оцінки (англ. cost function). Така функція має вказувати кількісно наскільки добре мережа вирішує поставлені їй завдання. Функція оцінки неявно залежить від всіх параметрів мережі і явно залежить відвихідного результату мережі. Найпростіший та найбільш поширений приклад оцінки — сума квадратів відстаней від вихідного результату мережі до їх необхідних значень:

$H = \frac{1}{2}\sum_{x \in out}\left(Z(x) - Z^*(x)\right)^2$, де $Z^*(x)$ — бажане значення вихідного сигналу.

Мінімізація середьоквадратичної відстані не завжди є найкращим підходом для тренування ШНМ. Правильно підібрана функція оцінки дозволяє підвищити ефективність тренування ШНМ на порядки. Деякі функції оцінки також можуть надавати інформацію про імовірність, що розв'язок, наданий нейромережею виявиться правильним.

1.5.1 Опис методу зворотнього розповсюдження помилки

Позначимо множину входів мережі як $x_1, \dots, x_n$, множина виходів - Outputs. Пронумеруємо наскрізно всі вузли числами від 1 до N. Позначимо через w_{ij} вагу зв'язку, що з'єднує i-й і j-й вузли, вихід i-го вузла - o_i. Бажані результати мережі відомі — $t_k, k \in Outputs$, оскільки вони є данними для тренування ШНМ. Функція помилки, отримана за методом найменших квадратів, буде виглядати так:

$$E(\{w_{ij}\}) = \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2$$

Суть стохастичного градієнтного спуску полягає в тому, що після кожного тестового прикладу, чи міні набору прикладів, вагові коефіцієнти мережі будуть незначним чином виправлятися. Таким чином буде реалізовано рух в багатовимірному просторі в сторону, протилежну до напрямку вектору градієнту. Зміна вагового коефіцієнту:

$\Delta w_{ij} = -\eta \frac{dE}{dw_{ij}}$, де $0 < \eta < 1$ — множник, що відповідає розміру кроку.

Порядок розрахунку похідної такий:

Спочатку ваговий коефіцієнт, який нас цікавить входить до нерону з вихідного шару, тобто $j \in Outputs$. w_{ij} впливає на на вихід мережі лише як частина суми $S_j = \sum_i w_{ij} x_i$ – сума входів j-того вузла. Тому

$$\frac{dE}{dw_{ij}} = \frac{dE}{dS_j}\frac{dS_j}{dw_{ij}} = x_i \frac{dE}{dS_j}$$

Отже S_j впливає на загальну помилку тільки в рамках j-того вузла. Якщо функція активації нейронів являє собою експоненціальну сигмоїду:

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

тоді:

$$\frac{dE}{dS_j} = \frac{dE}{do_j}\frac{do_j}{dS_j} = \left(\frac{d}{do_j}\frac{1}{2}\sum_{k\in Outputs}(t_k - o_k)^2)\right)\left(\frac{d\sigma(S_j)}{dS_j}\right)$$

$$= \left(\frac{1}{2}\frac{d}{do_j}(t_j - o_j)^2\right)\left(o_j(1-o_j)\right) = -o_j(1-o_j)(t_j - o_j)$$

Якщо ж у j-того вузла є виходи, то

$$\frac{dE}{dS_j} = \sum_{k\in Children(j)} \frac{dE}{dS_k}\frac{dS_k}{dS_j}$$

також:

$$\frac{dS_k}{dS_j} = \frac{dS_k}{do_j}\frac{do_j}{dS_j} = w_{ij}\frac{do_j}{dS_j} = w_{ij}o_j(1-o_j)$$

$\frac{dE}{dS_k}$ — це поправка аналогічна, але обчислена для вузла наступного шару.

Позначимо її як δ_k. Вона відрізняється від Δ_k тільки відсутністю множника $-\eta x_{ij}$. Алгоритм тренування можна створити на основі того, що

було показано як обчислювати поправку для вузлів вихідного шару та виразити поправку для вузла нижчого шару через поправки більш високого. Цей алгоритм називається методом зворотнього поширення помилки саме через цю особливість обчислення поправок.

Отже для вузла останнього рівня:

$$\delta_j = -o_j(1-o_j)(t_j - o_j)$$

Для внутрішнього вузла мережі:

$$\delta_j = -o_j(1-o_j)\sum_{k \in Children(j)} \delta_k w_{jk}$$

Для всіх вузлів:

$$\Delta w_{ij} = -\eta \delta_j x_i$$

Порядок тренування штучної нейронної мережі наведено нижче.

Розглянемо процес ініціалізації початкових синаптичних вагових коефіцієнтів. Припускаючи відсутність апріорної інформації, генеруємо синаптичні ваги і порогові значення за допомогою рівномірно розподілених чисел із середнім значенням 0. Дисперсія вибирається таким чином, щоб стандартне відхилення індукованого локального поля нейронів доводилася на лінійну частину сигмоїдальной функції активації (і не досягала області насичення). Вибір праильної дисперсії для вагових коефіцієнтів – дуже важливий етап тренування ШНМ. Якщо початкові вагові коефіцієнти мережі будуть замалими, то сигнал буде неминуче зменшуватись, під час проходження через кожен з шарів мережі. Таким чином, сигнал потрібної сили не зможе дійти до вихідного шару. Якщо ж початкові синаптичні вагові коефіцієнти вибрано завеликими, то результуючий сигнал вийде масивним, та імовірно зможе активувавти забагато нейронів.

Для вибору правильних початкових вагових коефіцієнтів можна використати метод Xavier. Його суть полягає у виборі дисперсії залежно від кількості нейронів n_{in}, виходи яких входять до поточного нейрону *W:*

$$Var(W) = \frac{1}{n_{in}}$$

Цей метод ініціалізації дозволяє зберегти синаптичні вагові коефіцієнти випадковими, а мережу – асиметричною. І при цьому він гарантує що сигнал у початковій мережі буде нормально доходити до останнії шарів, незначно змінюючи прицьому свою амплітуду.

Пред'явлення прикладів тренування. У мережу подаються образи з навчальної множини (епохи).

Прямий прохід (forward computation). Нехай приклад тренування представлений парою $(x(n), d(n))$, де $x(n)$ - вхідний вектор, який пред'являється вхідному шару сенсорних вузлів; $d(n)$ - бажаний відгук, що надається вихідному прошарку нейронів для формування сигналу помилки. Обчислюємо індуковані локальні поля і функціональні сигнали мережі, проходячи по ній пошарово в прямому напрямку.

Індуковане локальне поле нейрона j шару l обчислюється за формулою:

$$y^l(n) = \sum \omega_{ij}^l(n)\, y_i^{l-1}(n), \qquad (2.2)$$

де $y_i^{(l-1)}(n)$ - вихідний (функціональний) сигнал нейрона i, розташованого в попередньому шарі $l-1$, На ітерації n;

$w_{ji}^{(l)}(n)$ - синаптична вага зв'язку нейрона j шару l з нейроном i шару $l-1$.

для $i = 0$ $y_0^{(l-1)}(n) = +1$, а $w_{j0}^{(l)}(n) = b_j^l(n)$ - поріг, застосовуваний до нейрона j шару l.

Якщо використовується сигмоїдальна функція, то вихідний сигнал нейрона j шару l виражається наступним чином:

$$y_j^{(l)}(n) = \varphi_j(v_j(n)). \qquad (2.3)$$

Якщо нейрон j знаходиться в першому прихованому шарі (тобто $l = 1$), То

$$y_j^{(0)}(n) = x_j(n), \tag{2.4}$$

де, $x_j(n)$- j-й елемент вхідного вектора x (n) .Якщо нейрон j знаходиться в вихідному шарі (тобто $l = L$, де L-глибина мережі), то

$$y_j^{(L)}(n) = 0_j(n). \tag{2.5}$$

Обчислюємо сигнал помилки

$$e_j(n) = d_j(n) - 0_j(n) \tag{2.6}$$

де $d_j(n)$ – j-й елемент вектора бажаного відгуку $d(n)$.

Зворотний прохід. Обчислюємо локальні градієнти вузлів мережі за такою формулою:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)), \\ \varphi_j'(v_j^{(l)}(n))\sum_k \delta_k^{(l+1)}(n)\delta_{kj}^{(l+1)}(n), \end{cases} \tag{2.7}$$

для нейрона j вихідного шару L, для нейрона j прихованого шару l, де штрих до функцій $\varphi_j'(\cdot)$ позначає диференціювання по аргументу.

Зміна синаптичних ваг шару l мережі виконується відповідно до узагальненого дельта-правила:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)y_j^{(l-1)}(n) \tag{2.8}$$

де η-параметр швидкості тренування; α - постійна моменту.

Ітерації. Послідовно виконуємо прямий і зворотний проходи, пред'являючи мережі всі приклади тренування з епохи, поки не буде досягнуто критерію зупину.

Порядок подання прикладів тренування може випадковим чином змінюватися від епохи до епохи. Параметри моменту і швидкості тренування налаштовуються (і зазвичай зменшуються) в міру зростання кількості ітерацій.

Метою тренування є знаходження такого набору вагових коефіцієнтів мережі, який забезпечує рішення даного конкретного завдання.

Розглянемо тренування мережі даними алгоритмом на прикладі тришарової нейронної мережі, представленої на рисунку 2.3.

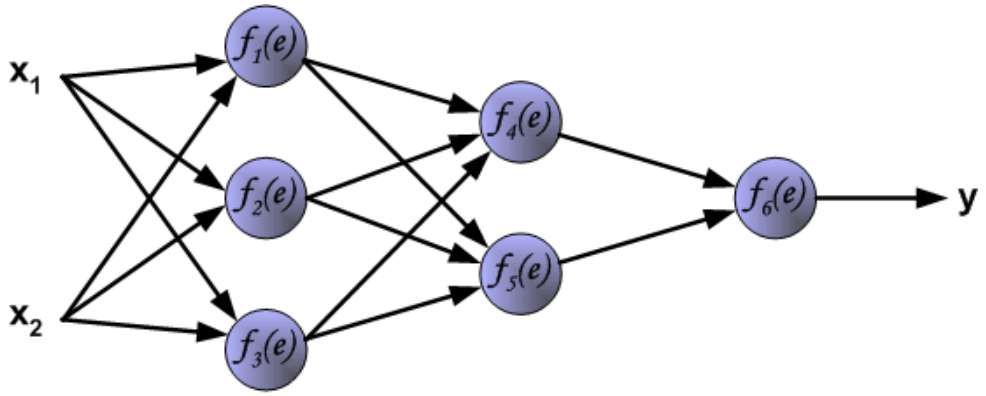


Рисунок 2.3 - Схема багатошарової нейронної мережі

Для тренування нейронної мережі ми повинні підготувати навчальні приклади.

Є приклади, що складаються з вхідних сигналів (X_1 і X_2) і бажаного результату Z.

У кожній ітерації вагові коефіцієнти нейронів підганяються з використанням нових даних з навчальних прикладів. Зміна вагових коефіцієнтів синаптичного зв'язку і становлять суть алгоритму.

Кожен крок тренування починається з впливу вхідних сигналів з навчальних прикладів. Після цього можна визначити значення вихідних сигналів для всіх нейронів в кожному шарі мережі.

На рисунку 2.4 показано напрямок сигналу в мережі.

$W(X_{mn})$ - вага зв'язку між входом X_m і нейроном n у вхідному шарі.

$Y(n)$ - вихід нейрона n.

$$\begin{aligned} y_1 &= f_1(w_{(x_1)1}x_1 + w_{(x_2)1}x_2), \\ y_2 &= f_2(w_{(x_1)2}x_1 + w_{(x2)2}x_2), \\ y_3 &= f_3(w_{(x_1)3}x_1 + w_{(x_2)3}x_2). \end{aligned} \tag{2.9}$$

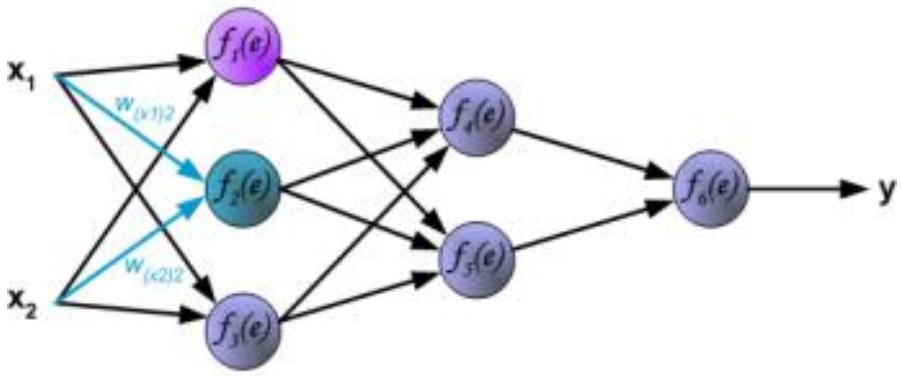


Рисунок 2.4 - Поширення сигналу від входу до 1-го прихованого шару

W_{mn} - ваги зв'язків між виходом нейрона m і входом нейрона n в наступному шарі.

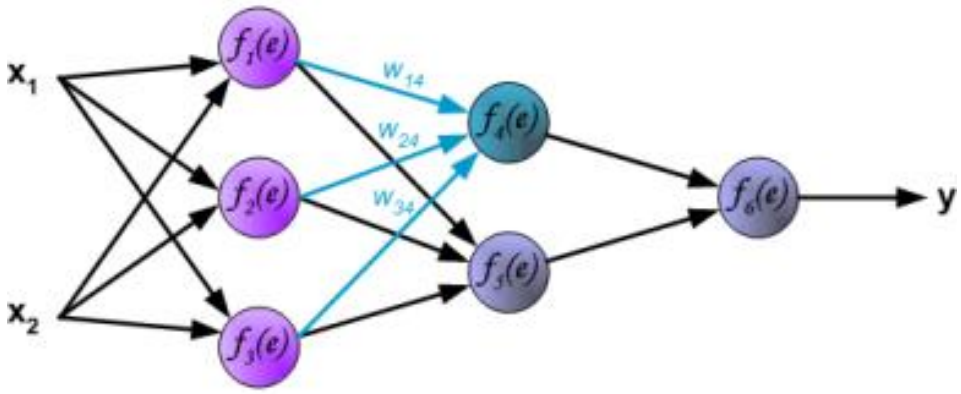


Рисунок 2.5 - Поширення сигналу до 2-го прихованого шару

На рисунку 2.4 показано як поширюється сигнал від першого шару до другого. У такому напрямку обчислюються виходи всіх нейронів, в тому числі і вихідного:

$$\begin{aligned} y_4 &= f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3), \\ y_5 &= f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3). \end{aligned} \tag{2.10}$$

Далі йде прохід сигналу через вихідний шар:

$$y = f_6(w_{46}y_4 + w_{56}y_5). \tag{2.11}$$

Наступний етап алгоритму полягає в наступному: вихідний сигнал y порівнюється з бажаним виходом мережі z.

δ - помилка (різниця) між сигналами z і y: $\delta = z - y$.

На рисунку 2.6 показано розподіл помилки по мережі

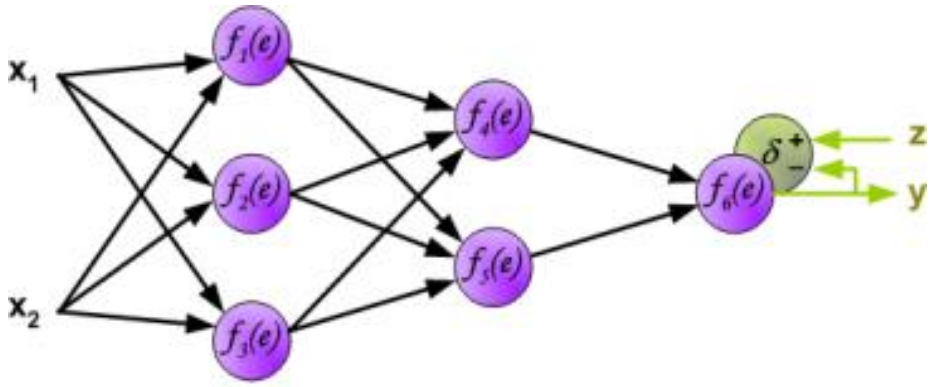


Рисунок 2.6 - Поширення помилки

Далі сигнал δ поширюється в зворотному напрямку на всі нейрони, чиї вихідні сигнали були вхідними для останнього нейрона.

$$\begin{aligned} \delta_4 &= w_{46}\delta, \\ \delta_5 &= w_{56}\delta. \end{aligned} \qquad (2.12)$$

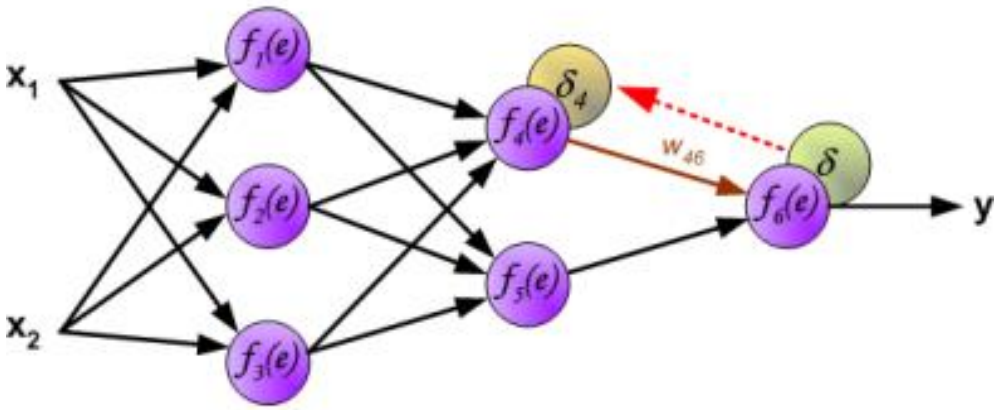


Рисунок 2.7 - Поширення помилки від вихідного шару до 2-го прихованого шару

Вагові коефіцієнти W_{mn} рівні тим же коефіцієнтам, що використовувалися під час обчислення вихідного сигналу y.

Змінюється напрямок потоку даних (сигнали передаються від виходу до входу). Процес повторюється для всіх верств мережі.

Якщо помилка прийшла від декількох нейронів - вона підсумовуються (рисунок 2.7):

$$\begin{aligned} \delta_1 &= w_{14}\delta_4 + w_{15}\delta_5, \\ \delta_2 &= w_{24}\delta_4 + w_{25}\delta_5, \\ \delta_3 &= w_{34}\delta_4 + w_{35}\delta_5. \end{aligned} \tag{2.13}$$

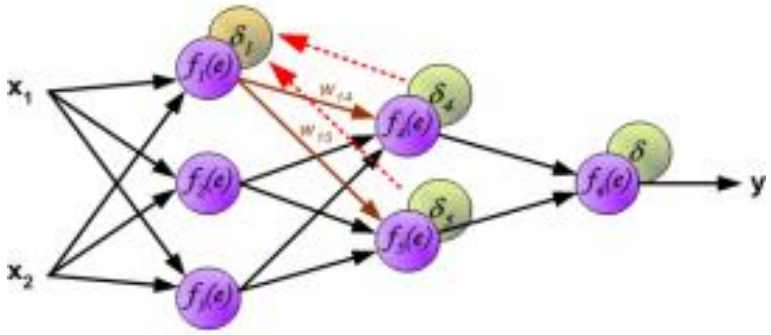


Рисунок 2.8 - Поширення помилки δ в зворотному напрямку в 1-ій прихованому шарі

Існує можливість перенавчання(англ. Overfitting) мережі. Весь набір образів, наданих до тренування, буде вивчено мережею, але будь-які інші образи, навіть дуже схожі, можуть бути класифіковані невірно.

Коли обчислюється помилка δ для кожного нейрона - можна скорегувати вагові коефіцієнти кожного нейрона (рисунок 2.9).

$$\begin{aligned} w'_{(x_1)1} &= w_{(x_1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1, \\ w'_{(x_2)1} &= w_{(x_2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2. \end{aligned} \tag{2.14}$$

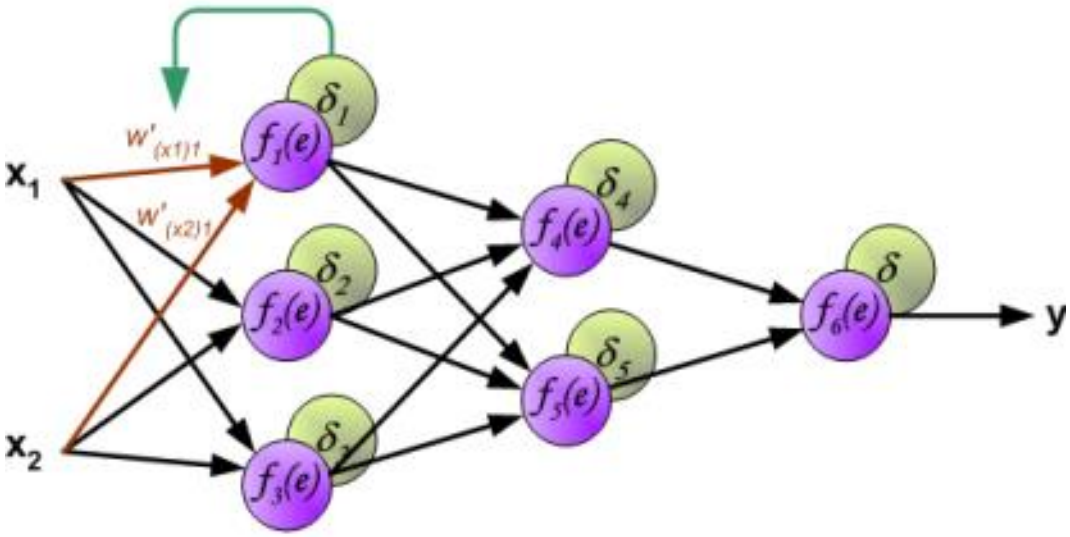


Рисунок 2.9 - Коригування ваг в 1-му шарі

де, $\frac{df(e)}{de}$ - похідна від функції активації нейрона (того, чиї ваги налаштовуються),

η - коефіцієнт, що впливає на швидкість тренування.

$$w'_{(x_1)2} = w_{(x_1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1,$$
$$w'_{(x_2)2} = w_{(x_2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2. \quad (2.15)$$

так само відповідно обчислюємо $w'_{(x_1)3}$ і $w'_{(x_2)3}$

На рисунку 2.9 зображено, в якому напрямку проходить корекція ваг в вихідному шарі.

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1,$$
$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2, \quad (2.16)$$
$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3.$$

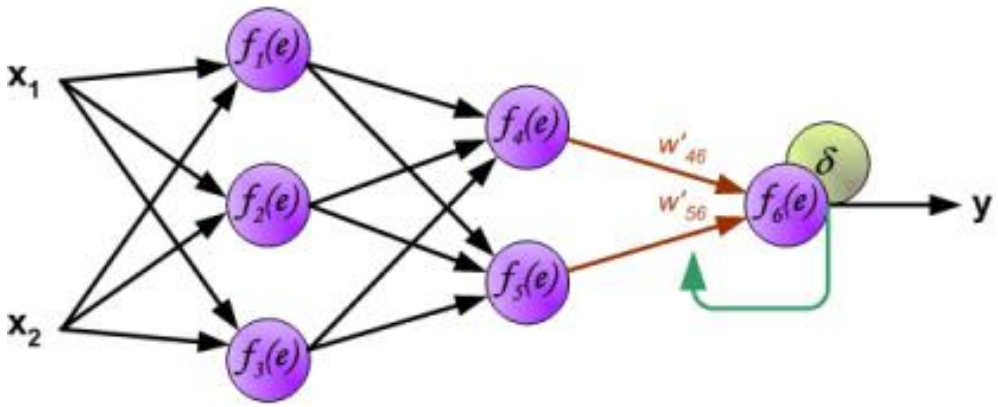


Рисунок 2.10 - Коригування ваг в вихідному шарі

Однак даний алгоритм має ряд недоліків. Серед таких недоліків можна виділити наступні:

Найбільше неприємностей приносить невизначено довгий процес тренування нейронної мережі, при цьому вона може і не навчитися зовсім. Причини можуть бути наступними:

- параліч мережі. У процесі тренування мережі значення ваг можуть в результаті корекції стати дуже великими величинами. Більшість нейронів будуть функціонувати в області, де похідна стискаючої функції дуже мала;

- локальні мінімуми. Зворотне поширення використовує різновид градієнтного спуску - спуск вниз по поверхні помилки, - безперервно підлаштовуючи ваги в напрямку до мінімуму.

Мережа може потрапити в локальний мінімум, коли поруч є набагато більш глибокий мінімум;

- розмір кроку. Корекції ваг передбачаються нескінченно малими. Це нездійсненно на практиці, так як веде до нескінченного часу тренування. Однак розмір кроку повинен братися кінцевим.

На вирішення цих недоліків будуть спрямовані евристичні методи, які будуть розглянуті далі.

Існує безліч градієнтних методів тренування нейронних мереж.

Метод градієнтного спуску.

Метод градієнтного спуску в "чистому" вигляді може "застрявати" в локальних мінімумах функції втрати E, для боротьби з цим використовується кілька доповнень для цього методу. Перше доповнення - це застосування стратегії mini-batch, а друге це так звані "моменти".

Метод моментів можна порівняти з поведінкою важкої кульки, яка скотилася по схилу в найближчу низину на деяку відстань, де здатна рухатися вгору за інерцією, вибираючись таким чином з локальних мінімумів. Формально це виглядає як добавка для зміни ваг мережі.

$$\Delta Wt: = \eta \cdot \nabla E + \mu \cdot \Delta Wt\text{-}1 \qquad (2.17)$$

де η - коефіцієнт швидкості тренування,

∇E - градієнт функції втрати,

μ- коефіцієнт моменту,

ΔWt-1 зміна ваг на попередній ітерації.

Ще одним удосконаленням методу оптимізації буде застосування регуляризації.

Регуляризація "накладає штраф" на надмірний ріст значень ваг. Це допомагає боротися з перенавчанням. Формально це виглядає як ще одна добавка для зміни ваг мережі.

$$\Delta Wt: = \eta \cdot (\nabla E + \rho \cdot Wt\text{-}1) + \mu \cdot \Delta Wt\text{-}1\Delta Wt, \qquad (2.18)$$

де η - коефіцієнт швидкості тренування,

∇E - градієнт функції втрати,

μ - коефіцієнт моменту,

ΔWt-1 зміна ваг на попередній ітерації,

ρ - коефіцієнт регуляризації,

Wt-1 - значення ваг на попередній ітерації.

Так само ми будемо змінювати коефіцієнт швидкості тренування η на кожній ітерації t в залежності від зміни помилки E наступним чином:

$$\begin{cases} \eta_t = \alpha\eta_{t-1}, \triangle E > 0 \\ \beta\eta_{t-1} \end{cases}, \qquad (2.19)$$

де $\eta_0 = 0.01$- початкове значення швидкості тренування,

ΔE = Et-γ - зміна помилки,

α = 0.99, β = 1.01, γ = 1,01- константи

Таким чином, при істотному зростанні помилки E крок зміни параметрів η, зменшується, в іншому випадку - крок η збільшується. Це доповнення може збільшити швидкість збіжності алгоритму.

Зберемо все разом, наш алгоритм набуває такого вигляду.

1. форматувати ваги W (випадковими малими значеннями)
2. форматувати нулями початкові значення зміни ваг ΔW
3. обчислюємо помилку E (h (X, W), C, W)) і її зміну ΔE на контрольному наборі
4. якщо результат задовільний, то виконуємо підсумковий тест і кінець роботи
5. випадковим чином вибрати підмножину (X, C) L з навчального набору
6. обчислюємо значення градієнта функції втрати ∇E на обраній підмножині
7. обчислюємо зміну параметрів: ΔW: = η·(∇E + ρ·W) + μ·ΔW
8. коригуємо параметри: W: = W-ΔW
9. коригуємо швидкість тренування η
10. перехід на п.3

Далі розглянемо метод quick Prop.

Від базового методу він відрізняється тим, що параметр моменту μ і коефіцієнт швидкості тренування η задаються індивідуально для кожного параметра. Зміна параметрів описується наступним співвідношенням.

$$\Delta W: = \eta\cdot(\nabla E + \rho\cdot W) + \mu\cdot\Delta W, \qquad (2.20)$$

де ρ - коефіцієнт регуляризації.

Параметр швидкості тренування обчислюється таким чином:

$$\eta = \begin{cases} \eta_0, (\Delta W = 0) \vee (-\Delta W \cdot S > 0) \\ 0 \end{cases}, \qquad (2.21)$$

де $\eta_0 \in (0.01,0.6)$- константа, S =∇E + ρW.

Параметр моменту виглядає наступним чином:

$$\begin{cases} \mu = \mu_{max}, (\beta > \mu\max) \vee (\gamma < 0) \\ \beta \end{cases}. \quad (2.22)$$

де μmax = 1.75 - константа,

S =∇E + ρW

β = S (t) / (S (t-1) -S (t))

γ = S·(-ΔW)·β.

Розглянемо метод r Prop.

У разі rProp моменти і регуляризація не використовуються, застосовується проста стратегія full-batch. Ключову роль відіграє параметр швидкості тренування η, він розраховується для кожної ваги індивідуально:

$$\eta(t) = \begin{cases} \min(\eta\max, a \cdot \eta(t-1)), S > 0 \\ \max(\eta\min, b \cdot \eta(t-1)), S < 0 \\ \eta(t-1), S = 0 \end{cases}. \quad (2.23)$$

де S =∇E (t-1)·∇E (t) - добутки значень градієнта на цьому і попередньому кроці,

ηmax = 50, ηmin = 10-6, a = 1.2, b = 0.5- константи

Зміна параметрів виглядає наступним чином.

$$\Delta Wt: = \eta \cdot (\text{Sign}(\nabla E) + \rho \cdot Wt\text{-}1) + \mu \cdot \Delta Wt\text{-}1. \quad (2.24)$$

Далі розглянемо метод сполучених градієнтів.

Особливість цього методу - спеціальний вибір напрямку зміни параметрів. Він вибирається таким чином, щоб бути ортогональним до попередніх напрямів. Повна зміна ваг виглядає наступним чином:

$$\Delta W: = \eta \cdot (P + \rho \cdot W) + \mu \cdot \Delta, \quad (2.25)$$

де η- коефіцієнт швидкості тренування,

p- напрямок зміни параметрів,

μ- коефіцієнт моменту,

ΔW- зміна ваг на попередній ітерації,

ρ- коефіцієнт регуляризації,

W- значення ваг на попередній ітерації.

При цьому коефіцієнт швидкості тренування η вибирається на кожній ітерації, шляхом вирішення задачі оптимізації:

$$\min E(\Delta W(\eta)). \quad (2.26)$$

Напрямок зміни параметрів вибирається в такий спосіб:

$$p = \nabla E + \beta \cdot p. \quad (2.27)$$

Початковий напрямок вибирається як p0: =∇E.

Ключовим моментом є обчислення коефіцієнта сполучення β.

Його можна обчислювати по крайній мірі двома різними способами. Перший спосіб - формула Флетчера-Рівса.

$$\beta = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}}, \quad (2.28)$$

Другий спосіб - формула Полак-Рібьєр:

$$\beta = \frac{g_t^T (g_t - g_{t-1})}{g_{t-1}^T g_{t-1}}, \quad (2.29)$$

де g =∇E- градієнт функції втрати на цій і попередній ітераціях.

Для компенсації накопиченої похибки обчислень метод передбачає скидання сполученого напрямку, тобто (B: = 0, p: =∇E) через кожні n циклів, де n вибирається залежно від кількості параметрів W.

Розглянемо далі так званий метод NAG.

Тут ми розглянемо модифікацію методу градієнтного спуску NAG, де градієнт обчислюється щодо зсунутих на значення моменту ваг:

$$\Delta Wt: = \eta \cdot (\nabla E(Wt-1 + \mu \cdot \Delta Wt-1) + \rho \cdot Wt-1) + \mu \cdot \Delta Wt-1, \quad (2.30)$$

де η- коефіцієнт швидкості тренування,

∇E- градієнт функції втрати,

μ- коефіцієнт моменту,

ΔWt-1- зміна ваг на попередній ітерації,

ρ- коефіцієнт регуляризації,

Wt-1- значення ваг на попередній ітерації.

Розглянемо метод AdaGrad.

Метод AdaGrad враховує історію значень градієнта таким чином

$$g_t = \frac{\nabla Et}{\sqrt{\sum_{i=1}^{t} \nabla E_i^2}}, \quad (2.31)$$

де η- коефіцієнт швидкості тренування,

∇E- градієнт функції втрати,

μ- коефіцієнт моменту,

ΔWt-1- зміна ваг на попередній ітерації,

ρ- коефіцієнт регуляризації,

Wt-1- значення ваг на попередній ітерації.

Метод AdaDelta.

Метод AdaDelta враховує історію значень градієнта і історію зміни ваг наступним чином.

$$t: = \alpha \cdot St\text{-}1 + (1-\alpha) \cdot \nabla E^2 t;\ S0: = 0, \quad (2.32)$$

$$Dt: = \beta \cdot Dt\text{-}1 + (1-\beta) \cdot \Delta W^2 t\text{-}1;\ D0: = 0, \quad (2.33)$$

$$\Delta Wt: = \eta \cdot (Gt + \rho \cdot Wt\text{-}1) + \mu \cdot \Delta Wt\text{-}1, \quad (2.34)$$

де η- коефіцієнт швидкості тренування,

∇E- градієнт функції втрати,

μ- коефіцієнт моменту,

ΔWt-1- зміна ваг на попередній ітерації,

ρ- коефіцієнт регуляризації,

Wt-1- значення ваг на попередній ітерації,

α = β = 0.9.

Розглянемо метод Adam.

Метод Adam перетворює градієнт наступним чином.

$$St: = \alpha \cdot St\text{-}1 + (1-\alpha) \cdot \nabla E^2 t;\ S0: = 0, \quad (2.35)$$

$$Dt: = \beta \cdot Dt\text{-}1 + (1-\beta) \cdot \nabla Et;\ D0: = 0, \quad (2.36)$$

$$\Delta Wt: = \eta \cdot (Gt + \rho \cdot Wt\text{-}1) + \mu \cdot \Delta Wt\text{-}1, \quad (2.37)$$

де η- коефіцієнт швидкості тренування,

∇E- градієнт функції втрати,

μ- коефіцієнт моменту,

ΔW_{t-1}- зміна ваг на попередній ітерації,

ρ- коефіцієнт регуляризації,

W_{t-1}- значення ваг на попередній ітерації,

$\alpha = 0.999$, $\beta = 0.9$

Як вказується в [25], градієнтні методи володіють тим же недоліком, що і розглянутий вище метод: при наявності ярів на поверхні збіжність методу дуже повільна.

Однак далі в дипломній роботі будуть показані результати моделювання і буде проведено порівняння з евристичними методами тренування.

2. ОПИС ЕВРИСТИЧНОГО АЛГОРИТМУ ТРЕНУВАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Евристичним називають метод розв’язання задачі оптимізації, який знаходить приблизне рішення, часто основуючись на випадкових процесах, але робить це шидше ніж відомі точні алгоритми. Часто швидкість досягається за рахунок програшу в оптимальності, повноцінності чи точності.

Основна ціль евристичного алгоритму – знайти розв’язок задачі оптимізації за прийнятний час. Цей розв’язок може бути не найкращим, чи він може просто апроксимувати більш точний розв’язок. Але такий розв’язок залишається цінним, оскільки на його пошуки витрачається значно менше часу. Результати дослідження NP-складних задач в теоритичних компютерних науках роблять евристичні алгоритми єдиною можливою опцією для розвязання деяких задач оптимізації, які потрібно вирішувати швидко, або навіть в реальному часі. Евристичні алгоритим також знаходять використання у області штучного інтелекту, чи комп’ютерних симуляціях мислення, а також в галузях, де не існує відомих точних алгоритмів.

Незважаючи на те, що традиційні методи тренування ШНМ (наприклад, метод зворотного поширення помилки чи класичний градієнтний спуск) широко використовуються у багатьох областях, вони мають певні недоліки. Ці недоліки стали основним вузьким місцем, яке обмежує їх подальший розвиток. У більшості випадків метод градієнтного спуску, що використовується в нейронних мережах прямого поширення, має такі основні недоліки:

1) Повільне тренування. Для остаточного визначення вагових коефіцієнтів може бути потрібно багато ітерацій в методі градієнтного спуску, особливо якщо функція помилки спадає повільно та має схожість з функцією Розенброка.
2) Глобальний мінімум може бути не знайдено, оскільки класичний алгоритм градієнтного спуску легко потрапляє до локального мінімуму.
3) Дуже чутливий до початкових значень коефіцієнтів. Якщо вибір коефіцієнтів некоректний, конвергентна швидкість алгоритму буде дуже малою, і процес тренування займе багато часу.

Тому, щоб підвищити ефективність тренування ШНМ, потрібно позбутися наведених проблем, що і стало предметом даного дослідження.

Задача тренування ШНМ зазвичай складна та має високу розмірність. Багато дослідників вважають за краще використовувати алгоритми, аналогічні алгоритму зворотного розповсюдження помилки для тренування ШНМ. МЗПП працюють, розраховуючи вихідну похибку та градієнт цієї похибки відносно кооефіцієнів ШНМ. Після цього відбувається корекція вагових коефіцієнтів і коефіцієнтів зміщення ШНМ в напрямку, протилежному градієнту. Ці градієнтні методи оцінюють помилку в результуючому векторі, що згенерелувала мережа, в порівнянні з учителем і розповсюджують похибку на коефіцієнти по всій мережі. Таким чином, одна з основних перешкод пов'язана з тим, що пошук оптимальних коефіцієнтів сильно залежить від початкових коефіцієнтів. Якщо вони розташовані біля

локального мінімуму, алгоритм буде може зійтися до неоптимального розв’язку.

Отже, звичайний метод пошуку градієнта схильний до конвергенції в локальних екстремумах. Запропоновані численні рішення, запропоновані нейромережевими дослідниками для подолання повільної швидкості конвергенції та потрапляння в місцевий мінімум. Існує кілька потужних алгоритмів оптимізації, які базуються на алгоритмі простого градієнтного спуску [14]. Наприклад: кон'югатний градієнтний спуск, масштабований градієнтний спуск, Квазі-Ньютоновські методи. Вони дозволяють підвищити швидкість конвергенції. Оскільки евристичні алгоритми не використовують градієнтну інформацію, це вигідно для проблем, коли така інформація недоступна або якщо її дуже важко отримати. Ці переваги роблять їх більш надійними та привабливими, ніж багато інших алгоритмів пошуку [15]. МРЧ як евристичний алгоритм, простий в реалізації та добре працює над багатьма проблемами оптимізації. Як і інші евристичні методики, МРЧ також може бути використаний у навчанні нейронних мереж.

2.1 Метод рою часток

Метод рою часток, МРЧ (англ. Particle Swarm Optimization, PSO) — метод чисельної оптимізації, для використання якого не потрібно знати точного градієнта оптимізованої функції. МРЧ був доведений Кеннеді, Еберхартом і Ші [2] і спочатку призначався для імітації соціальної поведінки. Алгоритм був спрощений, і було зауважено, що він придатний для виконання оптимізації. Книга Кеннеді й Еберхарта описує багато філософських аспектів МРЧ і так званого ройового інтелекту. Велике дослідження застосувань МРЧ зроблене Полі. МРЧ оптимізує функцію, підтримуючи популяцію можливих розв'язків, називаних частками, і переміщаючи ці частки в просторі розв'язків згідно із простою формулою. Переміщення підпорядковуються принципу

найкращого знайденого в цьому просторі положення, що постійно змінюється при знаходженні частками вигідніших положень [4].

Колекти́вний інтеле́кт (англ. Swarm intelligence) описує комплексну колективну поведінку децентралізованої системи, що самоорганізується. Розглядається в теорії штучного інтелекту як метод оптимізації. Термін був уведений Херардо Бені й Ван Цзином в 1989 році, у контексті системи клітинних роботів. Інколи колективний інтелект ще називають ройовим інтелектом.

Із точки зору інформатики, колективний інтелект — це частина комп'ютерних наук, яка проектує та вивчає ефективні числові методи розв’язання проблем у спосіб, схожий з поведінкою «колективу» живих організмів. Досягнення в цій галузі, а це власне розроблені алгоритми, застосовуються перш за все в задачах комбінаторної оптимізації та для розв'язування задачі комівояжера.

Системи колективного інтелекту, як правило, складаються із множини агентів (багатоагентна система), що локально взаємодіють між собою й із навколишнім середовищем. Самі агенти зазвичай досить прості, але всі разом, локально взаємодіючи, створюють так званий колективний інтелект. Прикладом у природі може служити колонія мурах, рій бджіл, зграя птахів, косяк риб.

Методи колективного руху агентів у колонії використовуються при проектуванні систем координованої роботи роботів. Розподілена взаємодія між агентами спонукала до створення декількох кластерних алгоритмів та алгоритмів упорядкування. Моделі розподілу праці між агентами колонії були використані для регулювання спільної роботи робототехніки.

Дизайнери використовують технології рою як засіб створення складних інтерактивних систем і моделювання натовпу. «Розбиваючи лід» — був перший фільм, що використовував технології колективного інтелекту для візуалізації, реалістичного зображення руху груп риб і птахів із використанням Boids системи. Тім Бертон створив фільм «Бетмен

повертається» також з використанням технології колективного інтелекту для демонстрації руху груп кажанів. У фільмі «Володар Кілець» використовували подібну технологію, відому як технологія масивів, під час батальних сцен. Такі технології є особливо привабливими, тому що використання колективного інтелекту — це дешевий, надійний і простий спосіб.

Авіакомпанії використовують теорію колективного інтелекту під час моделювання пасажирів перед посадкою в літак. Дослідник Дуглас Лоусон використовував мурах на основі комп'ютерного моделювання та визначив існування лише шести правил взаємодії пасажирів та спромігся оцінити час посадки з використанням різних методів посадки[8].

Колективний інтелект може бути використаний у цілому ряді програм. Збройні сили США використовують методи колективного інтелекту для управління безпілотними транспортними засобами. Європейське космічне агентство думає про «орбітальний рій» для самостійної збірки і інтерферометрії. NASA досліджує використання технології колективного інтелекту для створення планетарних карт. У 1992 році робота M. Anthony Lewis and George A. Bekey довела можливість використання розвідки роєм, за допомогою колективного інтелекту для контролю нанороботів у тілі з метою знищення ракових пухлин. Також колективний інтелект застосовується для інтелектуального аналізу даних.

Використання колективного інтелекту в телекомунікаційних мережах також досліджували, у вигляді основ мурашиної маршрутизації. Це було вперше відкрито Dorigo та Hewlett Packard в середині 1990-х років із низкою змін з тих пір. В основному це використання імовірнісних таблиць маршрутизації з використанням «нагородження» — зміцнення успішно пройденого маршруту кожної «мурашки» (невеликий пакет управління), який проходить у мережі. Посилення маршруту вперед, у зворотному напрямку, і обидва одночасно були досліджені. У зворотному напрямку зміцнення вимагає симетричної мережі і пар в обох напрямках разом. Мобільні засоби

масової інформації і нові технології можуть змінити поріг для колективних дій у зв'язку з ростом інтелекту систем.

Авіакомпанії використовували мурашину маршрутизацію в призначенні воріт для літака, що прибуває в аеропорт. В авіакомпанії Southwest програма використовує колективний інтелект, тобто теорію, що колонія мурашок працює краще, ніж поодинці. Кожен пілот працює як мураха в пошуках найкращих воріт в аеропорт. «Пілот вчиться на власному досвіді, що найкраще для нього, і виявляється, що це найкраще рішення для авіакомпанії» пояснює Дуглас Лоусон. У результаті роботи колонії, кожен пілот завжди прямує до вільних воріт.

Нехай $f: R^n \to R$— цільова функція, яку потрібно мінімізувати, S — кількість часток у рою. У алгоритмі МРЧ, кожна i-та частинка має позицію $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ в просторі пошуку, та швидкість $v_i = (v_{i1}, v_{i2}, \dots, v_{in})$. Позиція кожної частинки позначає один з розв'язків цілевої функції. Швидкість та позиція кожної частинки оновлюється на кожній ітерації алгоритму, беручи до уваги кращу позицію поточної частинки p_i та кращий глобалтний розв'язок, знайдений всією популяцією g. Ці залежності представленні таким чином:

$$v_i(t+1) = \omega\, v_i(t) + \varphi_p r_p \times (p_i - x_i(t)) + \varphi_g r_g \times (g - x_i(t)) \qquad (2)$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Параметри ω, φ_p, і φ_g є коєфіцієнтами інерції, когнітивним та соціальним відповідно.

Тоді загальний вигляд методу рою часток такий:

1. Для кожної частки i = 1, …, S зробити:
   1.1. Згенерувати початкове положення частки за допомогою випадкового вектора $x_i \sim U(b_{lower}, b_{upper})$, що має

багатовимірний рівномірний розподіл. b_{lower}, b_{upper}— нижня й верхня границі простору розв'язків відповідно.

1.2. Присвоїти найкращому відомому положенню частки його початкове значення: $p_i \leftarrow x_i$.

1.3. Якщо $\left(f(p_i) < f(g)\right)$, то оновити найкращий відомий стан рою: $g \leftarrow p_i$.

1.4. Присвоїти значення швидкості частки: $v_i \sim U\left(-(b_{upper} - b_{lower}),\ (b_{upper} - b_{lower})\right)$.

2. Поки не виконаний критерій зупинки (наприклад, досягнення заданого числа ітерацій або необхідного значення цільової функції), повторювати для кожної частки *i = 1, …, S*:

2.1. Згенерувати випадкові вектори $r_p, r_g \sim U(0,1)$.

2.2. Оновити швидкість частки: $v_i \leftarrow \omega\, v_i + \varphi_p r_p \times (p_i - x_i) + \varphi_g r_g \times (g - x_i)$, де операція $\times$ означає покомпонентне множення.

2.3. Оновити положення частки переносом x_i на вектор швидкості: $x_i \leftarrow x_i + v_i$. Зауважимо, що цей крок виконується незалежно від поліпшення значення цільової функції.

2.4. Якщо $\left(f(x_i) < f(p_i)\right)$, то робити:

2.4.1. Оновити найкраще відоме положення частки: $p_i \leftarrow x_i$.

2.4.2. Якщо $\left(f(p_i) < f(g)\right)$, то оновити найкращий відомий стан рою в цілому: $g \leftarrow p_i$.

3. Тепер g містить найкраще зі знайдених розв'язків.

2.2 Модифікації методу рою частинок

В оригінальному методі рою частинок, швидкість кожної частинки залежить лише від глобального кращого розв'язку та історично збереженого кращого розв'язку конкретної частинки. Таким чином, якщо глобальний карщий розв'язок виявиться одним з локальних мінімумів, алгориму буває складно вийти з такої ситуації та знайти глобальний мінімум. Щоб вирішити таку проблему в цьому розділі запропоновано три стратегії вдосконалення.

Модифікація МРЧ з вибором кращого сусіднього розв'язку

Припускається, що на швидкість збіжності рою може впливати не тільки краща позіція частинки, та глобальний кращий розв'язок рою але і краща позиція сусідньої частинки p_{Ni}^{best}. Імовірно, що в деяких випадках така позиція може позитивніше вплинути на наступну позицію частинки, ніж p_i. Для визначення підходящих частинок-сусідів можна використати різні підходи. У данному дослідженні використано перервірку середньої Евклідової відстані між частинками. Нехай $d(i,j)$ – Евклідова відстань між і-тою та j-тою частинками та md_i – середня Евклідова відстань для і-тої частинки. Тоді:

$$md_i = \frac{\sum_{j=1}^{S} d(i,j)}{S-1}$$

Якщо

$$d(i,j) < md_i, \tag{3}$$

то j-та частинка може бути прийнятою за сусідню до і-тої. Такий підхід можна узагальнити додавши фактор радіусу пошуку сусідніх частинок r. Тоді (3) може бути замінено на $d(i,j) \leq r \times md_i$. Якщо $r = 0$ то алгоритм працюватиме як звичайний МРЧ, без модифікації. Після вибору сусідньої

частинки, у випадку якщо $f\left(p_{Ni}^{best}\right) < f(p_i)$, то p_{Ni}^{best} використовується замість p_i у формулі (2), тобто

$$v_i(t+1) = \omega\, v_i(t) + \varphi_p r_p \times \left(p_{Ni}^{best} - x_i(t)\right) + \varphi_g r_g \times (g - x_i(t)), \quad (4)$$

інакше використовується формула (2). При використанні даної модифікації, на кожній ітерації алгоритму, перед визначенням швидкості частинки відбуається вибір найкращого сусіда, та у випадку, якщо його позиція краще ніж p_i, вона використовується для визначенні нової швидкості (4).

Модифікація МРЧ з заміною непривабливих розв'язків

Щоб уникнути передчасної сходимості до локального мінімуму можна запропонувати модифікацію алгоритму, яка заключається у видаленні імовірно поганих розв'язків, та заміні їх на випадково згенеровані. Нехай L – деяке число, що визначає швидкість видалення старих розв'язків. На початку роботи алгоритму, для кожної частинки ініціалізуємо $l_i(t) = 0\ (i = 1, \dots, S, t = 0)$. На кожній ітерації алгоритму, якщо позиція x_i не може бути змінена в кращу сторону, то модифікуємо:

$$l_i(t+1) = l_i(t) + 1,$$

інакше зкинемо значення до нуля:

$l_i(t+1) = 0$

Якщо розв'язок x_i не може бути покращено L разів, тобто якщо $l_i(t) = L$, то позиція x_i буде видалена, та замінена на x_{new}:

$$x_{new} = x_t + sE_t\ , \quad (4)$$

де E_t взято з нормального розподілу з нульовим середнім і єдиним стандартним відхиленням для випадкових блукань, або взяті зі стійкого

розподілу для польоту Lévy. Тут розмір кроку *s* визначає, наскільки далеко ходок може піти за фіксоване число ітерацій. В генерації Леві визначення розміру кроку складне, тому також часто використовують алгоритм Мантени [14]. Якщо *s* дуже велике, то новий згенерований розв'язок буде занадто далеко від старого (або може навіть не належати до допустимих розв’язків). Тоді такий крок не буде прийнято. Якщо *s* занадто мале, зміни занадто малі, щоб бути значними, і, отже, такий пошук не є ефективним. Таким чином, для найбільшої ефективності пошуку розмір кроку має дуже велике значення. Також розмір кроку значно залежить від області визначення функції та обмежень, накладених на неї для пошуку.

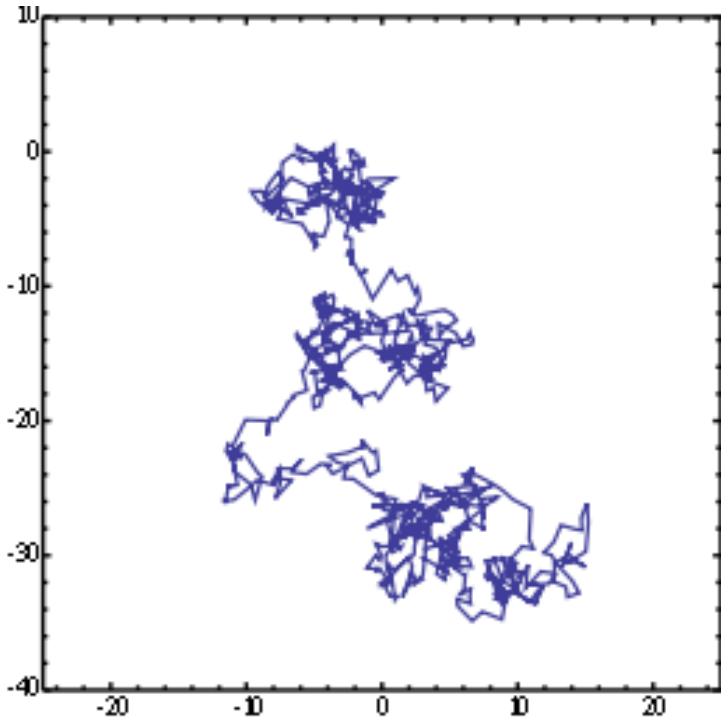


Рис. 2.1 Приклад розв’язків, згенерованих розподіленням Lévy у двовимірному просторі.

Модифікація МРЧ з додаванням хаотичного оператору пошуку

Щоб покращити глобальну конвергентну швидкість алгоритму ММРЧ, було додано хаотичний оператор пошуку. Для генерації хаотичної складової використано залежність:

$ch_{i+1} = 4 * ch_i * (1 - ch_i), 1 \leq i \leq K,$ де K – довжина хаотичної послідовності, та $ch_0 \in (0,1)$ – випадкова величина.

Таким чином, новий розв'язок може бути згенеровано за формулою:

$$x_{new} = (1 - \lambda) * g + \lambda * ch, \quad (5)$$

де λ – фактор розміру кола пошуку. Розмір кола пошуку будемо лінійно зменшувати, залежно від номеру ітерації алгоритму:

$$\lambda = \frac{maxcycle - t + 1}{maxcycle}$$

Таким чином, з плином ітерації алгоритму, хаотична складова буде все менше впливати на новий розв'язок. Це дозволяє залишити здібність алгоритму знаходити більш точний розв'язок при наближенні до останньої ітерації, але додасть можливість знайти нові розв'язки на початкових ітераціях. В свою чергу, така можливість виявляється корисною для запобігання потрапляння алгоритму до локального мінімуму.

Тоді модифікований метод рою частинок (ММРЧ) буде виглядати так:

1. Для кожної частки i = 1, …, S зробити:
   1.1. Згенерувати початкове положення частки за допомогою випадкового вектора $x_i \sim U(b_{lower}, b_{upper})$, що має багатовимірний рівномірний розподіл. b_{lower}, b_{upper}— нижня й верхня границі простору розв'язків відповідно.
   1.2. Присвоїти найкращому відомому положенню частки його початкове значення: $p_i \leftarrow x_i$.

1.3. Якщо $\left(f(p_i) < f(g)\right)$, то оновити найкращий відомий стан рою: $g \leftarrow p_i$.

1.4. Присвоїти значення швидкості частки: $v_i \sim U\left(-(b_{upper} - b_{lower}),\ (b_{upper} - b_{lower})\right)$.

2. Поки не виконаний критерій зупинки (наприклад, досягнення заданого числа ітерацій або необхідного значення цільової функції), повторювати для кожної частки *i = 1, …, S*:

2.1. Згенерувати коефіцієнт інерції $\omega \leftarrow \omega_{max} - (\omega_{max} - \omega_{min}) * \frac{t}{maxcycle}$

2.2. Згенерувати випадкові вектори $r_p, r_g \sim U(0,1)$.

2.3. Ініціалізувати $p_{local} \leftarrow p_i$

2.4. Для кожного з *k* сусідів, що задовільняють $d(i,j) \leq r \times md_i$

2.4.1. Якщо $\left(f(p_{local}) < f(p_j)\right)$, то $p_{local} \leftarrow p_j$

2.5. Оновити швидкість частки: $v_i \leftarrow \omega\, v_i + \varphi_p r_p \times (p_{local} - x_i) + \varphi_g r_g \times (g - x_i)$, де операція × означає покомпонентне множення.

2.6. Оновити положення частки переносом x_i на вектор швидкості: $x_i \leftarrow x_i + v_i$. Зауважимо, що цей крок виконується незалежно від поліпшення значення цільової функції.

2.7. Якщо значення цільової функції поліпшилось, то $l_i \leftarrow 0$, інакше $l_i \leftarrow l_i + 1$.

2.8. Якщо $l_i = L$, то x_i замінюється на новий розв'язок згідно формул (4) та (5).

2.9. Якщо $\left(f(x_i) < f(p_i)\right)$, то робити:

2.9.1. Оновити найкраще відоме положення частки: $p_i \leftarrow x_i$.

2.9.2. Якщо $\left(f(p_i) < f(g)\right)$, то оновити найкращий відомий стан рою в цілому: $g \leftarrow p_i$.

3. Тепер g містить найкраще зі знайдених розв’язків.

3. ТЕСТУВАННЯ МОДИФІКОВАНОГО АЛГОРИТМУ РОЮ ЧАСТИНОК

Було проведено серію тестових експериментів для порівняння модифікованого методу рою частинок та методу без модифікацій.

Цільова функція	Розмірність	Область визначення	Глобальний мінімум
$f_1 = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	0
$f_2 = \sum_{i=1}^{n} i x_i^2$	30	[-10, 10]	0
$f_3 = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	30	[-30, 30]	0
$f_4 = \sum_{i=1}^{n} \left(x_i^2 - 10 \cos \left(2\pi x_i \right) + 10 \right)$	30	[-5.12, 5.12]	0
$f_5 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	[-600, 600]	0
$f_6 = \sum_{i=1}^{n} \left| x_i \right| + \prod_{i=1}^{n} \left| x_i \right|$	30	[-10, 10]	0
$f_7 = -20 \exp \left(-0.2 * \sqrt{\sum_{i=1}^{n} \frac{z_i^2}{n}} \right) - \exp \left(\sum_{i=1}^{n} \cos \left(\frac{2\pi z_i}{n} \right) \right) + 20 + e$	30	[-32, 32]	0
$f_8 = \frac{1}{4000} \sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos \left(\frac{z_i}{\sqrt{i}} \right) + 1$	30	[-600, 600]	0
$f_9 = \sum_{i=1}^{n} z_i^2$	30	[-100, 100]	0
$f_{10} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j \right)^2$			

	30	[-100, 100]	0

Таблиця 1 – Список цільових функцій для тестування.

Тестові експерименти були проведені на ноутбуці з процесором Inter® Core(TM) i5-3230M 2.60 GHz. Він має можливість виконувати паралельно тільки чотири потоки.

У якості цільових функцій було використоно десять функцій, наведених в таблиці 1. Ці функції відібрано з рекомендованого списку функцій для тестування алгоритмів оптимізації. Деякі з них мають склдну структуру та багато локальних мінімумів. Знаходження мінімуму цих функцій складає певну складність для будь якого алгоритму оптимізації. Наприклад на рисунку 3 зображено тривимірний графік однієї з цільових функцій (f_7). На рисунку можна побачити, що функція має дуже багато локальних мінімумів, що значно ускладнює збіжність алгоритму.

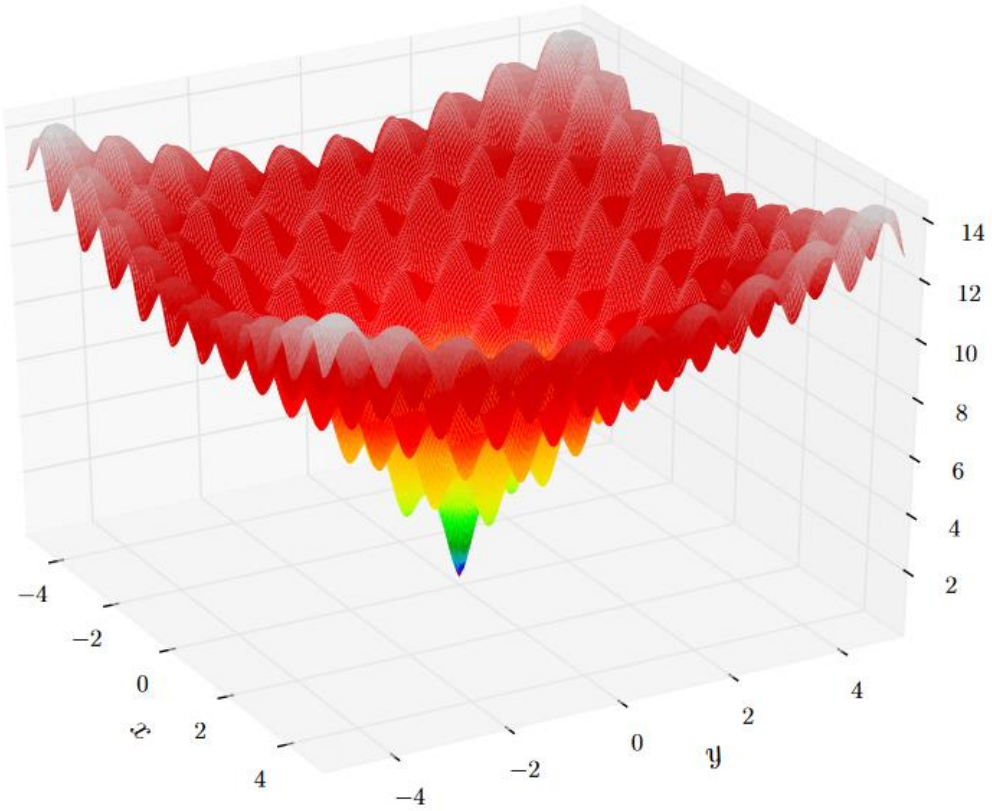


Рисунок 3.1 Тривимірний графік функції Аклі (Ackley's function)

Спільні парамерти обох алгоритмів:

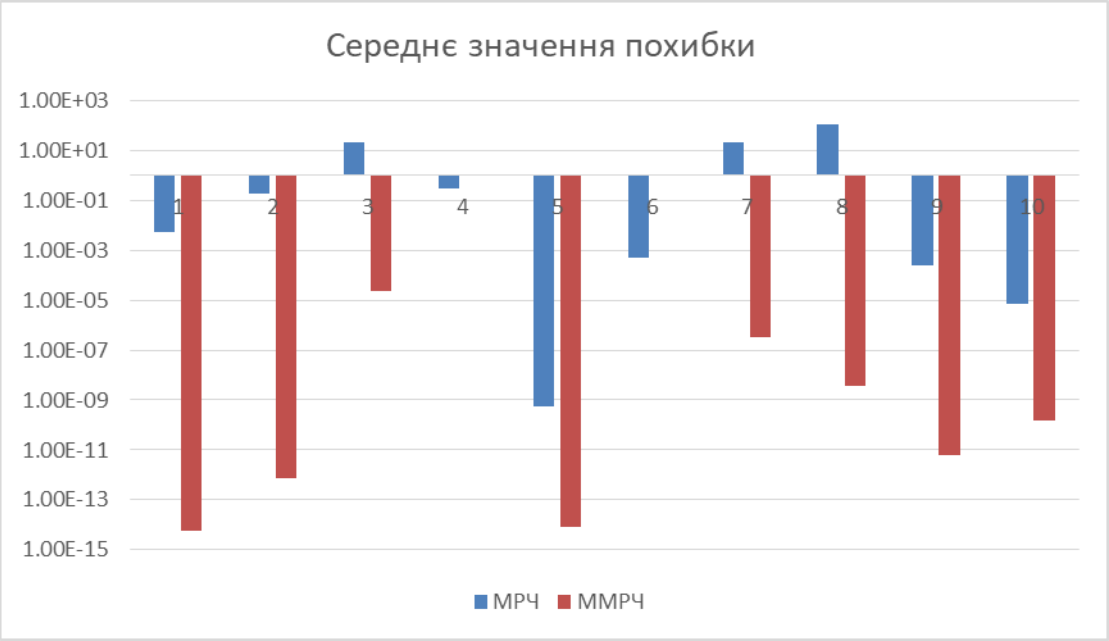
Розмірність n= 30, кількість частинок S = 30, кількість ітерацій *maxcycle* = 2000. Параметри, що були підібрані експериментальним шляхом для ММРЧ $\omega_{min} = 0.4$, $\omega_{max} = 0.9$, L = 5, r = 0.01. Кожен експкримент бло повторено 10 разів незележно один від одного.

Цільова функція	Алгоритм	Середнє	SD	Мінімальне
f_1	МРЧ	5.07e-03	1.25e-03	4.11e-03
	ММРЧ	5.67e-15	7.34e-16	6.23e-16
f_2	МРЧ	1.81e-01	4.32e-01	8.45e-02
	ММРЧ	7.23e-13	9.28e-14	6.67e-16
f_3	МРЧ	2.21e+01	3.24e-01	1.30e+01
	ММРЧ	2.22e-05	4.93e-05	9.38e-13
f_4	МРЧ	2.90e-01	0	2.90e-01
	ММРЧ	0	0	0
f_5	МРЧ	5.32e-10	3.24e-10	1.30e-10
	ММРЧ	8.32e-15	8.23e-15	9.76e-16
f_6	МРЧ	4.97e-04	4.65e-04	1.91e-04
	ММРЧ	0	0	0
f_7	МРЧ	2.03e+01	1.22e-02	2.02e+01
	ММРЧ	3.29e-07	3.76e-10	1.75e-08
f_8	МРЧ	1.05e+02	9.24e-02	1.04e+02
	ММРЧ	3.68e-09	6.27e-09	6.00-12
f_9	МРЧ	2.32e-04	2.03e-05	2.30e-04

	ММРЧ	6.30e-12	1.04e-11	1.29e-14
f_{10}	МРЧ	7.42e-06	1.03e-06	6.23e-06
	ММРЧ	1.43e-10	2.32e-10	4.65e-12

Таблиця 2 – Порівняння результатів роботи МРЧ та ММР

З проведених експериментів видно, що модифікації алгориму покращують загальну сходимість на різних функціях. Модифікований алгорим знаходить глобальний мінмум функції значно точніше, ніж алгоритм без модифікацій. Різницю в похибці алгоритмів яскраво видно на графіку нижче:



Бачимо, що у випадку f_4 та f_6 алгоритму ММРЧ вдалось знайти мінімум абсолютно точно. Це може бути зв’язано з відносною простотою функцій, та з меншою областю визначення ніж у інших функцій. Вцілому алгоритм ММРЧ показує значно кращі та стабільніші результати, ніж алгоритм без модифікацій, за незмінної кількості ітерацій.

4. ТРЕНУВАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Як приклад ШНМ розглянемо багатошаровий перцептрон, оскільки така модель є одною з найпоширеніших та широко використовується для розв’язування найрізноманітніших задач.

Багатошаровий перцептрон — це перцептрон з додатковими шарами штучних нейронів, розташованими між вхідним та вихідним шаром [1]. Виходи нейронів кожного шару з’єднані з входами всіх нейронів наступного шару. Таким чином, вхідний сигнал поширюється в одному напрямку, утворюючи нейронну мережу прямого спрямування.

Кількість нейронів вхідного та вихідного шарів визначається з умов задачі, для розв’язування якої будується ШНМ. Кількість вхідних нейронів залежить від розмірності задачі, а вихідних від кількості ознак чи кількості класів об'єктів. Кількість додаткових (прихованих) шарів та число нейронів у кожному шарі залежить від складності класифікації та часто підбирається для кожної задачі окремо.

Основна задача в процесі тренування ШНМ — це підбір коректних вагових коефіцієнтів для кожного зв’язку між нейронами, що будуть забезпечувати максимальний відсоток бажаних значень на виході мережі.

4.1 Розробка архітектури ШНМ для розв’язання задачі розпізнавання рукописних чисел

Проблему класифікації рукописних чисел можна розділити на дві підпроблеми. По-перше необхідно розбити зображення рукописного числа на складові зображення цифер. По-друге необхідно класифікувати кожне зображення цифри в один з десяти класів.

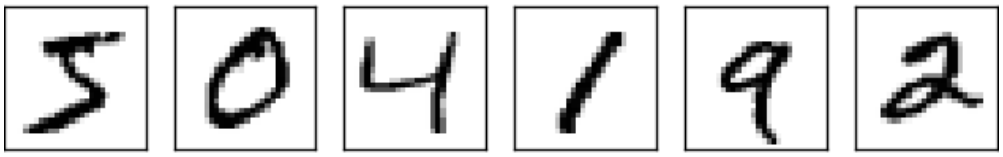


Рисунок 1 – приклади рукописних цифр для розпізнавання

У данному дослідженні увагу було сфокусовано на другій частині проблеми, а саме на класифікації конкретних цифр. Оскільки якщо знайдено надійний спосіб класифікації зображень конкретних цифр, то задачу

розділення зображення числа на цифри розв’язати стає набагато простіше. Один із способів – розділити цільове зображення на велику кількість зображень різних розмірів, та спробувати класифіувати ці зображення. Надалі відсіяти завеликі та замалі зображення цифр, які будуть мати низьку точність класифікації. Основна ідея такого методу заключається в тому, що якщо класифікатор має багато проблем у визначенні цифри на якомусь із розбитихзображень, то імовірно, на ньому тільки частина цифри, більше одної цифри, або взагалі немає цифри. Такий чи інший підхід, з деякими модифікаціями, може бути використано для розбиття цілевого зображення на складові цифри. Тож сконцентруємо увагу на ШНМ для розпізнавання однієї цифри.

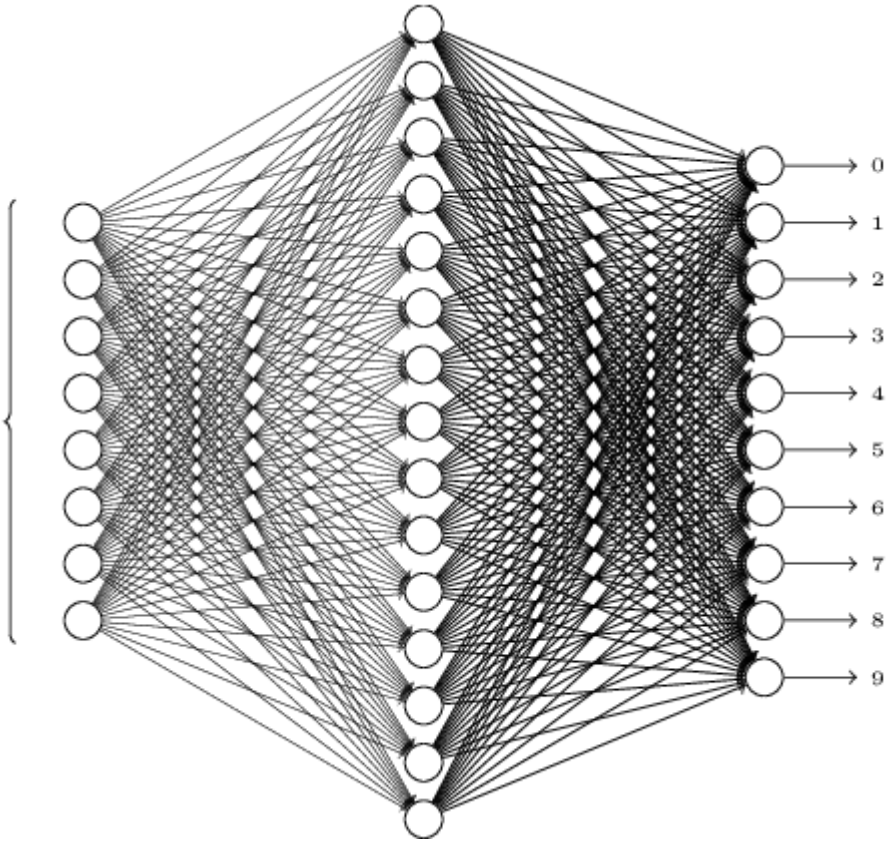
	Прихований шар (15 нейронів)	Вихідний шар (10 нейронів)
Вхідний шар (784 нейрони)		

Рисунок 2 – модель ШНМ

Кожен нейрон вхідного шару отримує сигнал, що залежить від інтенсивності відповідного пікселя в тестовому зображенні. Всі зображення в тестовому наборі данних попередньо було приведено до монохромного вигляду. Перед подачею до вхідного шару значення відповідного пікселю нормалізується до проміжку [0, 1]. Всі тестові зображення мають розширення 28x28 пікселів, отже вхідний шар мусить містити 784 нейрона. Вихідний шар містить десять нейронів, отже, якщо значення сигналу відповідного нейрону вихідного шару наближається до одиниці, то можна говорити про те, що мережа класифікує вхідне зображення як відповідну цифру під номером цього нейрона. Наприклад, якщо значення сигналу приблизно дорівнює одиниці на четверому нейроні, то мережа класифікує вхідне зображення як зображення цифри чотири і т.д.

Прихований шар містить тільки 15 нейронів. Його розмір було підібрано експериментальним шлхом. Збільшення кількості нейронів прихованого шару може призводити до незначного збільшення точності класифікації, але значно ускладнює тренування ШНМ. Оскільки кожен новий нейрон в прихованому шарі додає ще 784 + 10 вагових коефіцієнтів, що необхідно підібрати в процесі тренування. Оскільки це збільшує розмірність задачі, збільшується і час тренування, незалежно від вибраного методу тренування ШНМ.

Тим не менш, деяка кількість нейронів в прихованому шарі все ж необхідна. Без прихованого шару нейронна мережа не буде знаходити спільні аспекти в данних для тренування, а буде просто запам’ятовувати данні. Таким чином вона може добре справлятись із зображеннями, на яких проводилось тренування, але показувати погані результати на нових зображеннях.

Наприклад можна припустити, що деякі чотири нейрони прихованого шару займаються класифікацією певних ознак різних цифр (Рисунок 3).

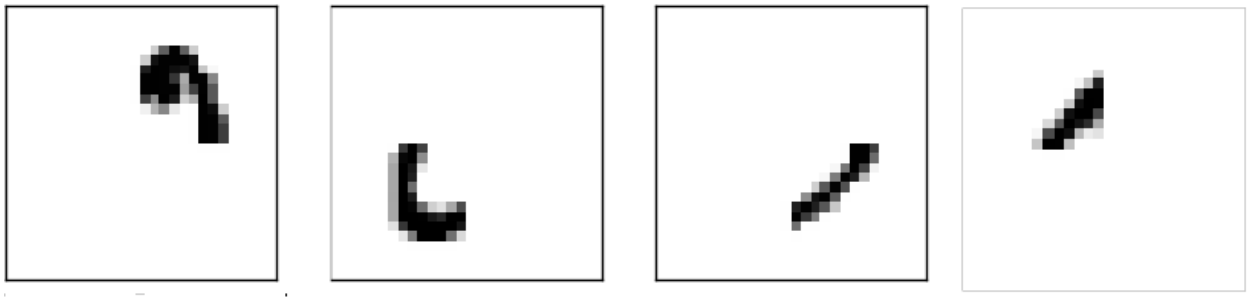


Рисунок 3 – Можливі ознаки різних цифр

Такого результату можна досягти, якщо вагові коефіцієнти відповідних нейронів вхідного шару будуть достатньо великими, щоб активувати нейрон прихованого шару, тільки у випадку, якщо є певна схожість з відповідною ознакою якоїсь цифри. Не важко здогадатись, що якщо активуються всі чотири описані вище нейрони прихованого шару, то має активуватись нейрон під номером нуль у вихідному шарі. Оскільки всі чотири з навединих вище ознак є тільки у цифри нуль (Рисунок 4).

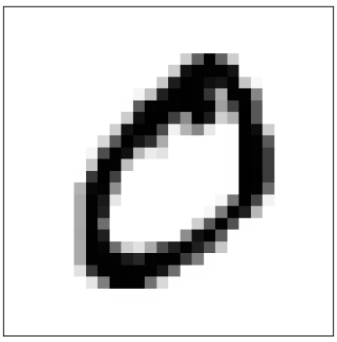


Рисунок 4 – Поєднання чотирьох ознак

Саме для класифікації таких проміжних ознак необхідний прихований шар нейронів. Складність та структура прихованого шару має відповідати кількості різноманітних ознак, що можуть бути використані для класифікації основгого об’єкту. Немає ніяких причин вважати, що ці ознаки мають бути саме такими, які наведено у прикладі (Рисунок 3). Але в процесі тренування нейронна мережа має визначити якись список ознак, та основувати подальшу класифікацію саме на ньому.

4.2 Вибір набору данних для тренування ШНМ

Перше, з чим необхідно визначитись для тренування ШНМ — це набір данних. У данному дослідженні буде використано набір данних MNIST, що

включає десятки тисяч відсканованих зображень рукописних цифр. Кожне зображення має розширення 28x28 пікселів та зберігається у відтінках сірого. Всі зображення попередньо класифіковано. Декілька зображень з цього набору данних можна побачити на рисунку 1. Набір данних включає 60 000 картинок, що були використанні для тренування нейронної мережі. Тестовий набір включає 10 000 картинок, що, для чистоти експерименту, було взято від людей, які не брали участь в написанні набору для тренування ШНМ. Цей факт всиляє впевненість в тому, що тренована ШНМ зможе розпізнавати рукописні цифри, які вона не зустрічала в тестовому наборі.

4.3 Тестування розробленої нейронної мережі

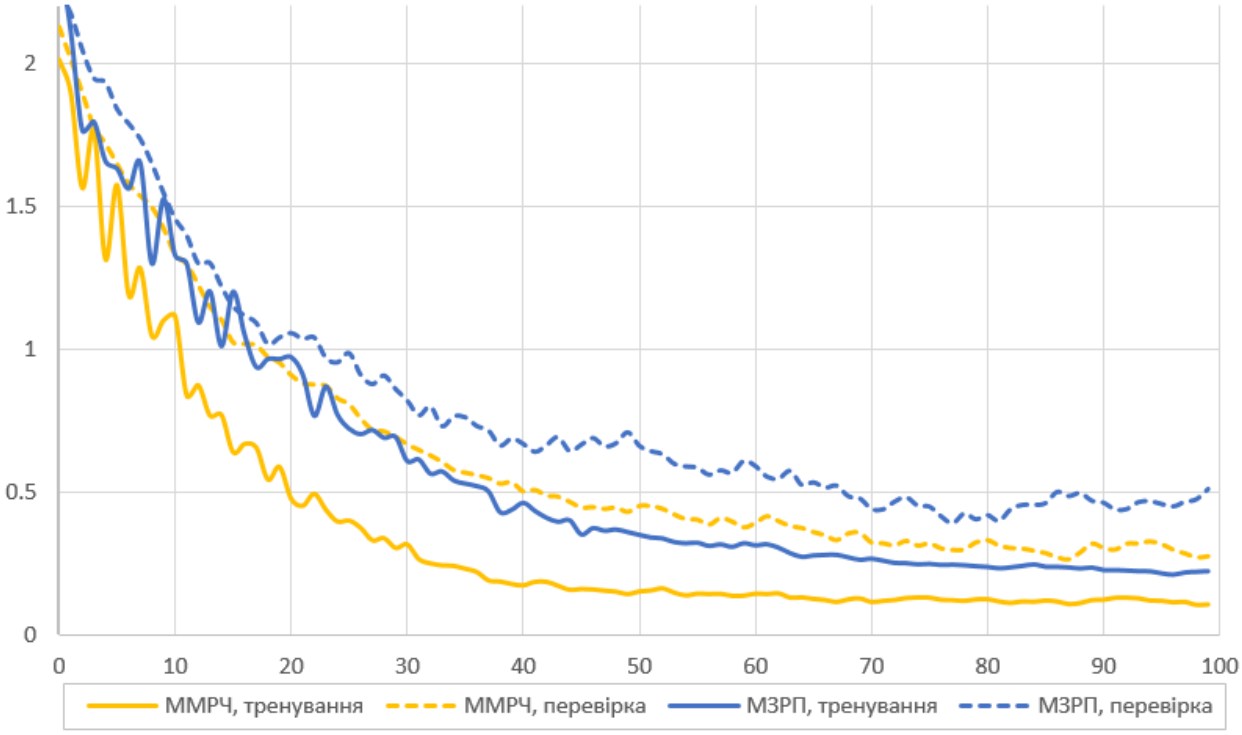
Було розроблене тестове програмне забезпечення на мові програмування C++. Програмне забезпечення включає реалізацію двох алгоритмів тренування ШНМ. У процесі розробки, для імпортування архітектури ШНМ було підключено бібліотеку tinyDNN, оскільки вона є зручною у використанні та підтримує популярні формати зберігання нейромереж. Для розбиття та нормалізації вхідних данних було розроблено кілька допоміжних консольних додатків на мові програмування Python.

Алгоритм ММРЧ було використано для мінімізації цільової функції середньоквадратичної помилки змодельованої нейронної мережі. Для порівняння результатів, було розроблено стандартний алгоритм зворотнього розповсюдження помилки (МЗПП). Кожна частинка алгоритму ММРЧ містить розв’язок нейронної мережі, тобто набір синаптичних коєфіцієнтів, необхідних для класифікації вхідного зразка. Значення функції середньоквадратичної помилки підраховується протягом тренування ШНМ на міні-вибірці з усього набору данних. Кожна міні-вибірка має розмір 40 зображень, та підбирається випадково, але так, щоб гарно представляти весь набір данних. Мається на увазі, що в кожній міні-вибірці підбирається

однакова кількість зображень з кожного класу. Тобто у данному випадку по чотири зображення кожної цифри.

На вибраному наборі данних MNIST було проведено 100 епох тренування кожного алгоритму.

Графік залежності середньоквадратичної помилки від епохи тренування:



Аналізуючи графік залежності помилки від епохи тренування, можна зробити висновок, що ММРЧ навчається швидше. Вже на 50-тій епосі ММРЧ показує результат, який звичайний МЗРП здобуває тільки на сотій епосі тренування. На кожній епосі тренування було виділено спеціальну тестову вибірку, на якій не продилось тренування, а проводилась тільки перевірка середньоквадратичної помилки. Графіки з тестових вибірок зображені пунктирною лінією.

Результати тренування мережі:

	Середньоквадартична помилка	Точність розпізнавання	Час роботи
ММРЧ,	0.10576	91.6%	328хв

тенування			
МЗРП, тренування	0.22198	89.3%	259хв
ММРЧ, перевірка	0.27425	87.8%	—
МЗРП, перевірка	0.50953	85.2%	—

З таблиці вище видно, що результуюча ШНМ, що тренувалась алгоритмом ММРЧ, має вищу точність класифікації на вибірці для тренування та на тестовій вибірці. Одним з недоліків алгоритму ММРЧ виявився час роботи. 100 епох алгоритму пройшло за 328хв, на відмміну від алгоритму МЗРП, який спрацював за 259хв. Але, якщо розглянути отримані данні більш детально, то можна побачити, що порівнювати одну епоху різних алгоритмів не зовсім коректно. Оскільки ММРЧ за одну епоху встигає обробити більше варіантів синаптичних вагових коефіцієнтів, тому витрачає більше часу на одну епоху.

Порівняємо середньоквадратичну помилку на різних епохах різних алгоритмів:

	Епоха	Середньоквадартична помилка	Час роботи
ММРЧ, тенування	50	0.15121	164хв
МЗРП, тренування	100	0.22198	259хв
ММРЧ, перевірка	50	0.45027	—
МЗРП, перевірка	100	0.50953	—

Отже, якщо представити данні в такому вигляді, то чітко видно, що ММРЧ справляється з задачею тренування нейронної мережі значно краще. За менший час вдалося отримати меннш середньоквадратичну помилку на всіх наборах данних.

ВИСНОВОК

У даній дипломній роботі було розглянуто модифікований метод рою частинок, як метод тренування штучної нейронної мережі. Було запропоновано декілька модифікацій до загальноприйнятого евристичного алгоритму оптимізації — методу рою частинок. Було досліджено, що запропоновані модифікації вигідно відрізняють отриманий метод від загальноприйнятого. Даний метод також було адаптовано для тренування штучної нейронної мережі, та проведено порівняльний аналіз з існуючим загальновідомим методом тренування – методом зворотнього розповсюдження помилки.

В процесі дослідження отримано такі результати:

1. Показано, що метод рою частинок інколи може збігатися до локального мінімуму. Запропоновано модифікації алгоритму, для запобігання такої ситуації, та показано їх дієвість, порівнюючи результати роботи двох алгоритмів.
2. Розроблено тестову архітектуру нейронної мережі для розпізнавання рукописних цифр та підібрано дані для тренування мережі.
3. Проведено експериментальні дослідження, які показали, що розроблений алгоритм є конкурентним та може давати прийнятні результати при тренуванні нейронних мереж.

Експериментальні дослідження доводять, що запропонований алгоритм ММРЧ гарно себе проявляє в тренуванні штучних нейронних мереж. Він здатен працювати швидше, та давати більш надійні результати, ніж загальноприйнятий алгоритм зворотнього розповсюдження помилки. ШНМ, що була тренована з допомогою алгоритму ММРЧ здатна правильно розпізнати 87.8% картинок рукописних цифр, що є кращим результатом, ніж ШНМ, що була тренована з допомогою алгоритму зворотнього розповсюдження помилки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Xabier Basogain Olabe. Redes Neuronales Artificiales y sus Aplicaciones Formato Impreso: Publicaciones de la Escuela de Ingenieros, 1998 – 79 p.
2. Минский М., Пейперт С. Персептроны. М.: Мир, 1971 - 261 с.
3. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления: Учеб. Пособие для вузов - М.: Высш. шк. 2002. - 183 с.: ил.
4. Хайкин С. Нейронные сети: полный курс, 2-е изд., испр. : Пер. с англ. - М. : ООО "И.Д. Вильямс", 2006. - 1104 с.
5. Дивеев А.И., Софронова Е.А. “Основы генетического программирования Учебно-методическое пособие” - М.: Изд-во РУДН, 2006;
6. Васенков Д.В. Методы обучения искусственных нейронных сетей //Компьютерные инструменты в образовании. - СПб.: Изд-во ЦПО "Информатизация образования", 2007, №1, С. 20-29.
7. Круг П.Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры». - М.: Издательство МЭИ, 2002. – 176 с.
8. Каллан, Р. Основные концепции нейронных сетей : Пер. с англ. - М. : Издательский дом "Вильямс"', 2001;
9. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. - М.: Горячая линия - Телеком, 2001. - С. 382.
10. Мочалов И.А. Искусственные нейронные сети в задачах управления и обработки информации Ч.1 - М.: 2004. -145 с.
11. Осовский С. Нейронные сети для обработки информации – М.: Финансы и статистика, 2002. – 344 с.
12. Пупков К.А., Егупов Н.Д. «Методы классической и современной теории автоматического управления»: Учебник в 5-и тт.; 2-е изд., перераб. и доп. Т.3: Синтез регуляторов систем автоматического управления / Под ред.

К.А. Пупкова и Н.Д. Егупова. - М.: Издательство МГТУ им. Н.Э. Баумана, 2004. - 616 с.; ил.

13. Кочладзе З.Ю., Оганезов А.Л. Об одном возможном подходе к проблеме распознаванияплоских фигур, Университетский журнал. Тбилиси, 2006.

14. Купарадзе М.Р. Структура и функции нервной системы, Москва, Сборник докладов, 1962

15. Чумбуридзе И.Ш., Нуцубидзе М.А., Микашавидзе Г.Н., Береникашвили Н.Н. К вопросу об организации нейронных сетей, способных к обу ению классификации событий. Тезисы докладов V Всесоюзной конференции по нейрокибернетике, РГУ Ростов на-Дону, 1973.

16. Аркадьев А. Г., Браверманн Э. М. Обучение машины классификации объектов, Москва,

“Наука”, 1971.

17. Бахтадзе Н.М., Кадагишвили Л.Г., Кецба М. . Методы определения параметров феномена бессознательной памяти, Материалы международ ой объединенной конференции по ИИ, Тбилиси, 1976.

18. Мегрелидзе К.Р. Основные проблемы социологии мышления, “Мецниэреба”, Тбилиси, 1973.

19. Воронцов, К.В. Оптимизационные методы линейной и монотонной коррекции в алгебраическом подходе к проблеме распознавания. ЖВМ и МФ, Том 40, № 1, С. 166-176, 2000.

20. Воронцов, К.В., Каневский, Д.Ю. Коэволюционный метод обучения алгоритмических композиций. Таврический вестник информатики и математики, № 2, С. 51-66, 2005.

21. Грешилов, А.А., Стакун, В.А., Стакун, А.А. Математические методы построения прогнозов. – М.: Радио и связь, 1997. – 112 с.

22. Журавлёв, Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации. Проблемы кибернетики, Том 33, С. 5-68, 1978.

23. Журавлёв, Ю.И. Экстремальные алгоритмы в математических моделях для задач распознавания и классификации. Доклады АН СССР, математика, Т. 231, № 3, 1976.

24. Кохонен, Т. Ассоциативные запоминающие устройства. М.: Мир, 1982. – 384 с.

25. Курейчик, В.М. Генетические алгоритмы и их применение. Таганрог: Издво ТРТУ, 2002.

26. Мазуров, В.Д. Комитеты системы неравенств и задача распознавания. Кибернетика, № 3, 1971.

27. Мазуров, В.Д. Метод комитетов в задачах оптимизации и классификации. – М.: Наука, 1990.

28. Boser, B., Guyon I., Vapnik, V. A training algorithm for optimal margin classifiers. In D. Haussler, editor, proceedings of the 5th Annual ACM Workshop on COLT, pp. 144-152, Pittsburgh, 1992.

29. Breiman, L. Bagging predictors. Machine Learning, Vol. 24 (2), pp. 123-140, 1996.

30. Shoghian, Sh., Kouzehgar, M. A Comparison among Wolf Pack Search and Four other Optimization Algorithms. World Academy of Science, Engineering & Technology, Vol. 6, Issue 72, pp. 418-423, 2012.

31. Zahadat, P., Schmickl, T. Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation. In proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO'2014), pp. 241-248, 2014.

32. Lia, C.-M., Duc, Y.-C., Wua, J.-X., Lind, C.-H., Hoe, Y.-R., Lina, Y., Chen, T. Synchronizing chaotification with support vector machine and wolf pack search algorithm for estimation of peripheral vascular occlusion in diabetes mellitus. Biomedical Signal Processing and Control, Vol. 9, pp. 45-55, 2014.

33. Yang, X.S. Firefly algorithms for multimodal optimization. In proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications, pp. 169-178, 2009.

34. Yang, X.S., Deb, S. Cuckoo search via Levy flights. In proceedings of World Congress on Nature & Biologically Inspired Computing, pp. 210-214, 2009.

35. Yang, X.S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization, Springer, SCI 284, pp. 65-74, 2010.

36. Auger A. Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB2009 function testbed. Rothlauf [22]. P. 2447–2452.

37. Hansen N., Auger A., Ros R., Finck S. Real-parameter black-box optimization benchmarking 2009: Experimental setup // Technical Report RR-6828, INRIA, 2009.