Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Адаптивна поліноміальна функція у глибинних нейронних мережах та алгоритм її навчання
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-2
Слепанська В.Д.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Чала Л.Е.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Слепанській Валерії Дмитрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Адаптивна поліноміальна активаційна функція в глибинних нейронних мережах та алгоритм її навчання

затверджена наказом університету від 29 березня 20 21 р. № 390Ст

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 21 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані відомих наукових проектів щодо розробки активаційних функцій, дані статей, результати експериментальних досліджень

4. Перелік питань, що потрібно опрацювати в роботі 1 Аналіз предметної галузі та формалізована постановка задачі; 2 Багатошарові нейронні мережі та алгоритми їх навчання; 3 Адаптивна поліноміальна активаційна функція в глибинних нейронних мережах; 4 Імітаційне моделювання.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – Модель Маккалоха-Пітса; Рисунок 2 – Модель сигмоїдального нейрону; Рисунок 3 – Графік сигмоїдальної функції; Рисунок 4 – Графік похідної від сигмоїдальної функції; Рисунок 5 – Графік впливу моменту на процес навчання; Рисунок 6 – Схема нейрону адаліна; Рисунок 7 – Мережа мадалайн; Рисунок 8 – Нейрон Хебба; Рисунок 9 – Структура одношарової сигмоїдальної НМ; Рисунок 10 – Сукупність даних для навчання функції XOR; Рисунок 11 – Ілюстрація неможливості лінійного розділення даних; Рисунок 12 – Рішення проблеми нелінійного розділення; Рисунок 13 – Структура ШНМ, яка виконує XOR; Рисунок 14 – Узагальнені структура двошарової НМ; Рисунок 15 – Ілюстрація застосування формування і збудження графа; Рисунок 16 – Ілюстрація застосування методу графів; Рисунок 17 – Схема нейрона з AdPReLU; Рисунок 18 – Squashing function; Рисунок 19 – Схема нейрона з APAF; Рисунок 20 – Змінення помилки з вихідним сигналом (4.1) Рисунок 21 – Змінення помилки з вихідним сигналом (4.2); Рисунок 22 – Змінення помилки з вихідним сигналом (4.3).

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Чала Л.Е.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	29.03.2021	виконано
2	Аналіз предметної галузі і постановка завдання	30.03.2021 31.03.2021	виконано
3	Дослідження існуючих моделей нейронів	01.04.2021 02.04.2021	виконано
4	Аналіз існуючих проблем даної галузі	02.04.2021 03.04.2021	виконано
5	Дослідження існуючих алгоритмів навчання	04.04.2021 05.04.2021	виконано
6	Створення імітаційної моделі	06.04.2021 07.04.2021	виконано
7	Тестування та опрацювання імітаційної моделі	7.04.2021 10.04.2021	виконано
8	Оформлення пояснювальної записки	11.04.2021 17.04.2021	виконано
9	Оформлення графічних матеріалів	17.04.2021 18.04.2021	виконано
10	Попередній захист	15.05.2021	виконано
11	Захист перед ЕК	18.05.2021	виконано

Дата видачі завдання 29 березня 20 21 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 93 с., 22 рис., 2 дод., 22 джерела.

АДАПТИВНА АКТИВАЦІЙНА ФУНКЦІЯ, ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, ДЕЛЬТА-ПРАВИЛО, НАВЧАННЯ З ВЧИТЕЛЕМ, СИНАПТИЧНІ ВАГИ, СІМЕЙСТВО ЛІНІЙНИХ ВИПРЯМЛЯЧІВ ФОРМАЛЬНИЙ НЕЙРОН.

Дана робота присвячена дослідженню та розробці адаптивної активаційної функції та алгоритму її навчання.

Метою роботи є створення адаптивної поліноміальної активаційної функції – adaptive polynomial activation function (APAF), параметри якої налаштовуються в процесі навчання подібно синаптичним вагам нейрону, покращуючи апроксимуючі властивості не тільки окремого нейрону, а й нейронної мережі в цілому.

Методи дослідження – математичний апарат обчислювального інтелекту, теорія оптимізації, лінійної алгебри, методи регуляризації та псевдо обернення.

Об'єкт дослідження – процес створення адаптивної поліноміальної активаційної функції та алгоритмів її налаштування.

Предмет дослідження – адаптивні поліноміальні активаційні функції в глибоких нейронних мережах.

РЕФЕРАТ

Пояснительная записка: 93 с., 22 рис., 2 прил., 22 источника.

АДАПТИВНАЯ АКТИВАЦИОННАЯ ФУНКЦИЯ, ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ, ДЕЛЬТА-ПРАВИЛО, ОБУЧЕНИЯ С УЧИТЕЛЕМ, СЕМЕЙСТВО ЛИНЕЙНЫХ ВЫПРЯМИТЕЛЕЙ, СИНАПТИЧЕСКИЕ ВЕСЫ, ФОРМАЛЬНЫЙ НЕЙРОН.

Данная работа посвящена исследованию и разработке адаптивной активационной функции и алгоритма её обучения.

Целью работы является создание адаптивной полиномиальной активационной функции – adaptive polynomial activation function (APAF), параметры которой настраиваются в процессе обучения подобно синаптическим весам нейрона, улучшая апроксимационные свойства не только отдельного нейрона, а и нейронной сети в целом.

Методы исследования – математический аппарат вычислительного интеллекта, теория оптимизации, линейная алгебра, методы регуляризации и псевдо обращения.

Объект исследования – процесс создания адаптивной полиномиальной активационной функции и алгоритмов её обучения.

Предмет исследования – адаптивные полиномиальные активационные функции в глубоких нейронных сетей.

ABSTRACT

Explanatory note contains: 93 p., 22 fig., 2 ann., 22 sources.

ADAPTIVE ACTIVATION FUNCTION, CONTROLLED LEARNING, DEEP NEURAL NETWORK, DELTA-RULE, FORMAL NEURON, RECTIFIED LINEAR FAMILY, SYNAPTIC WEIGHTS.

This work is devoted to researching and development of adaptive activation function and its learning algorithm.

The purpose of researching is the creation of an adaptive polynomial activation function APAF whose parameters are tuned in the learning process like the synaptic weights of the neuron, while improving the approximating properties of not only the individual neuron but also the neural network in general.

Methods of researching is mathematical apparatus of computational intelligence, theory of optimization, linear algebra, methods of regularization and pseudo-inversion.

The object of researching is the process of adaptive polynomial activation function creation and its tuning algorithm.

The subject of researching is adaptive polynomial activation function in deep neural networks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШНМ – штучна нейронна мережа;

AdPReLU – Adaptive parametric rectified linear unit – адаптивний параметричний лінійний випрямляч;

APAF – Adaptive polynomial activation function – адаптивна поліноміальна активаційна функція;

DL – Deep Learning – глибинне навчання;

ReLU – rectified linear unit – випрямлена лінійна одиниця.

ВСТУП

Сьогодні нейронні мережі використовують як альтернативу всім існуючим алгоритмам задля рішення різноманітних завдань інтелектуального аналізу даних, таких як розпізнавання образів, прогнозування, задачі ідентифікації та адаптивного управління . Глибоке навчання (Deep learning, DL) – метод машинного навчання, заснований, у першу чергу, на нейронних мережах, хоча можна застосовувати й інші методи. У сучасній реальності практично у всьому, що стосується глибокого навчання, використовують нейронні мережі. Ця популярність пов’язана з різноманітними властивостями нейронних мереж, як, наприклад, здатність до налаштування своєї архітектури та параметрів під час обробки даних. Крім того, штучні нейронні мережі відрізняються високими показниками апроксимації, порівнюючи з іншими методами.

Найпоширенішими є багатошарові персептрони, що складаються з так званих вузлів-нейронів – елементарних персептронів Розенблатта. Найвідомішими активаційними функціями є сигмоїдальна функція, функція гіперболічного тангенсу, поліноміальні активаційні функції та ін.

Впродовж свого розвитку були розроблені нейронні мережі, що використовувалися для машинного перекладу, розпізнавання мови та музики, обробки зображень, визначення об'єктів на фото та відео. Але незважаючи на набутий успіх та популярність, нейронні мережі також мають і недоліки, такі як, наприклад, явища затуючого та вибухаючого градієнту, внаслідок яких з’являються обчислювальні проблеми. Також вони виникають через специфічність форм представників сімейства сигмоїди.

Для вирішення проблем сигмоїдальних активаційних функцій у глибинних нейронних мережах почали використовуватися кусково-лінійні функції сімейства rectified linear unit. Вони вирішують проблему затухаючого градієнту та дають можливість оптимізувати процес навчання

нейронної мережі. Однак суттєвим недоліком даних функцій є те, що їх використання призводить до збільшення прихованих шарів нейронної мережі, ускладнюючі її та суттєво зменшуючи швидкість навчання мережі.

Саме тому у рамках сімейства ReLU була введена адаптивна активаційна функція adaptive rectified linear unit (AdPReLU). Її особливістю стала можливість налаштування свої параметрів подібно до налаштування синаптичних вагів. Дана активаційна функція показала приголомшливі результати по швидкості навчання, порівнюючи з класичними найбільш поширеними активаційними функціями.

Однак було вирішено розробити adaptive polynomial activation function, з додатковими 6 параметрами для налаштування, з оптимізуючим критерієм навчання, що в свою чергу дозволить значно покращити апроксимуючі властивості нейрону, а в майбутньому і нейронної мережі загалом. В рамках даної дослідницької роботи буде порівняна апроксимаційна здатність нейронів з класичними активаційними функціями, такими як сигмоїда, гіперболічний тангенс, ReLU з нейронами з AdReLU та APAF.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

Вузли штучної нейронної мережі, іменовані також штучними нейронами (нейронними клітинами, формальними нейронами) представляють собою елементарні процесори і є спрощеними моделями біологічних нейронів. Це спрощення визначається перш за все тим, що інженерів цікавлять тільки функції нейронів, пов'язані з переробкою інформації. Крім того, не слід забувати про високу складність біологічних систем, які в повній мірі просто не піддаються математичному опису. Тому в теорії ШНМ нейрон − це система відображення з n-мірного простору входів, формованого сигналами з виходів інших нейронів, або зовнішнім середовищем, в одномірний простір (скалярний сигнал) на виході нейронної клітини.

Більшість нейронних мереж утворено однотипними нейронами це гомогенні (однорідні) мережі, хоча відомі гетерогенні мережі, сконструйовані з різних нейронів. Також, слід зауважити, що нейрони бувають аналоговими і бінарними, хоча цей поділ є умовним, оскільки один і той же формальний нейрон може функціонувати як в аналоговому, так і в цифровому режимах.

На сьогоднішній день існує безліч активаційних функцій, але для досягнення поставленої мети необхідно буде вирішити ряд завдань:

– проаналізувати базові поняття проблемної області штучних нейронних мереж;

– розглянути існуючі типи нейронів та активаційних функцій штучних нейронних мереж;

– проаналізувати існуючі алгоритми навчання нейронів;

– розробити адаптивну поліноміальну активаційну функцію (APAF) для оптимізації алгоритму навчання за швидкодією;

– вирішувати за допомогою розробленої активаційної функції проблеми глибинних нейронних мереж.

1.1 Моделі нейронів та методи їх навчання

Відповідно до принципів функціонування біологічних нейронів були створені різні математичні моделі, які реалізують властивості природної нервової клітини. Загальною схемою, що складає основу більшості з таких моделей, є модель Маккхалока-Пітса, що складається з суматора зважених, вхідних сигналів та нелінійний блок відпрацьованих даних вихідного сигналу нейрона, який функціонально залежить від вихідного сигналу суматора. Властивості нелінійної функції, особливо її неперервність, мають вплив на вибір методу навчання нейрона (вибір вагових коефіцієнтів). Іншим важливим фактором є вибір стратегії навчання. Можна виділити два метода: навчання з вчителем та навчання без вчителя.

При навчанні з учителем передбачається, що, крім вхідних сигналів, складових вектору x, відомі також і очікувані вихідні сигнали нейрона, що складають вектор d. У подібній ситуації підбір вагових коефіцієнтів повинен бути організований так, щоб фактичні вихідні сигнали нейрона у брали б значення, як можна ближчі до очікуваних значень d. Ключовим елементом процесу навчання з учителем є знання очікуваних значень вихідного сигналу нейрона.

Якщо такий підхід є неможливим, залишається обрати стратегію навчання без вчителя. Підбір вагових коефіцієнтів в цьому випадку проводиться на основі конкуренції нейронів між собою (стратегія, коли переможець отримує все) або стратегія, коли переможець отримує більше, або ж з урахуванням кореляції навчаючих та вихідних сигналів (навчання по Хеббу). При навчанні без вчителя на етапі адаптації нейрона ми не можемо прогнозувати його вихідні сигнали, тоді як при навчанні з вчителем

результат навчання визначений заздалегідь завдяки апріорі заданим навчальним вибіркам.

1.2 Персептрон

Простий персептрон – це звичайна модель Маккхалока-Пітса з відповідною стратегією навчання [1]. Структурна схема і позначення елементів, i -го персептрона представлені на рисунку 1.1.

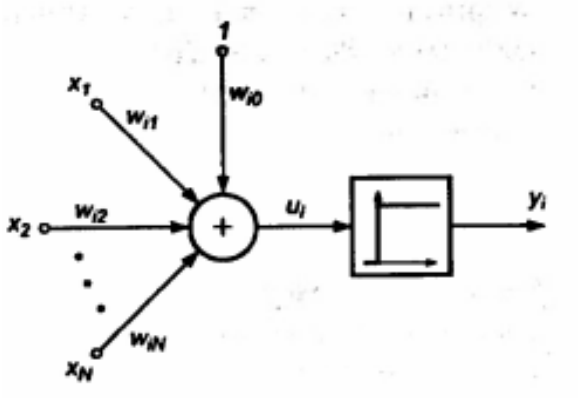

Рисунок 1.1 – Модель Маккхалока-Пітса

Вагові коефіцієнти входів суматора, на які надходять вхідні сигнали нейрона x_j, позначаються w_{ij}, а порогове значення, яке надходить з так званого поляризатора, – w_{i0}.

Нелінійна функція активації персептрона є дискретна функція ступеневого типу, внаслідок чого вихідний сигнал нейрона може приймати тільки два значення – 0 або 1 відповідно до правила:

$$y_i(u_i) = \begin{cases} 1, u \geq 0 \\ 0, u < 0 \end{cases}, \tag{1.1}$$

де u_i – вихідний сигнал суматора

$$u_i = \sum_{j=0}^{N} w_{ij} x_j . \qquad (1.2)$$

У наведеній формулі мається на увазі, що маючи довжину N вектор Вхідних сигналів x доповнений нульовим членом $x_0 = 1$ формує сигнал поляризації, $x = [x_0, x_1, ..., x_N]$. Навчання персептрона вимагає наявності вчителя і полягає в такому підборі ваг w_{ij}, щоб вихідний сигнал y_i був найбільш близький до заданого значення d_i. Це навчання гетероасоціативного типу, при якому кожній навчальній вибірці, представленої вектором x, апріорі поставлено у відповідність очікуване значення d_i на виході i-го нейрона.

Найбільш популярний метод навчання персептрона полягає в застосуванні правила персептрона [1], [2], [3], [4], відповідно до якого підбір ваг здійснюється згідно з наступним алгоритмом:

– випадковим чином обираються значення для ваг w_{ij}, а далі на вхід нейрона подається навчальний вектор x і розраховується значення вихідного сигналу y. За результатами порівняння фактично отриманого значення y_i із заданим значенням d_i уточняються значення ваг;

– якщо значення y_i збігається з очікуваним значенням d_i, то вагові коефіцієнти w_{ij} не змінюються;

– якщо $y_i = 0$, а задане значення $d_i = 1$, то значення ваг w_{ij}, уточняються відповідно до формули:

$$w_{ij}(t+1) = w_{ij}(t) + x_j , \qquad (1.3)$$

де t – це номер попереднього циклу, а $(t+1)$ – номер поточного циклу;

– якщо, а відповідне задане значення $d_i = 0$, то значення ваг уточняються відповідно до формули:

$$w_{ij}(t+1) = w_{ij}(t) - x_j \ , \tag{1.4}$$

де t – це номер попереднього циклу,

$(t+1)$ – номер поточного циклу;

– після завершення уточнення вагових коефіцієнтів представляється ще один вектор для навчання x та пов'язані з ним очікуване значення d_i, та значення ваг уточняється ще раз. Цей процес повторюється на всіх навчаючих вибірках, поки не будуть мінімізована різниця між усіма значеннями y_i та відповідними значеннями d_i.

Слід зазначити, що правило персептрона є окремим випадком запропонованого набагато пізніше правила Уідроу-Хоффа [3], [5].

Відповідно до цього правила підбору вагових коефіцієнтів нейрона, не обов'язково персептроного типу, проводиться по формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij} \, , \tag{1.5}$$

$$\Delta w_{ij} = x_j (d_i - y_i) \, . \tag{1.6}$$

Аналогічні співвідношення використовуються при підборі ваг поляризатора w_{i0}, для якого вхідний сигнал завжди дорівнює 1, в зв'язку з чим:

$$\Delta w_{i0} = (d_i - y_i) \, . \tag{1.7}$$

Легко помітити, що якщо сигнали y_i та d_i приймають тільки такі значення як 0 і 1, то правило Уідроу-Хоффа перетворюється в правило персептрона. Характерна особливість як правила персептрона, так і узагальненого правила Уідроу-Хоффа полягає у використанні для навчання інформації тільки про поточне та очікуване значення вихідного сигналу. У зв'язку з розривністю нелінійної функції активації персептрона неможливо враховувати інформацію про зміну значення y_i (тобто її похідну).

Мінімізація відмінностей між фактичними реакціями нейрона y_i і очікуваними значеннями d_i може бути представлена як мінімізація конкретної функції похибки (цільової функції) E, найчастіше визначається як:

$$E = \sum_{k=1}^{p} (y_i^{(k)} - d_i^{(k)})^2, \tag{1.8}$$

де p означає кількість пропонованих навчальних вибірок.

Така мінімізація при використанні правила персептрона проводиться за методом безградієнтної оптимізації [1]. Ефективність методу при великій кількості навчальних вибірок відносно невелика, а кількість циклів навчання і його тривалість зростають дуже швидко, причому без будь-якої гарантії досягнення мінімуму цільової функції. Усунути ці недоліки можна тільки в разі застосування безперервної функції активації, при якій цільова функція E також стає безперервною, що дає можливість використовувати в процесі навчання інформацію про величину градієнта.

1.3 Сигмоїдальний нейрон

Нейрон сигмоїдального типу (рисунок 1.2) має структуру, подібну моделі Маккхалока-Пітса, але відрізняється тим, що функція активації є

неперервною та може бути виражена у вигляді сигмоїдальної уніполярної чи біполярної функції [3], [6].

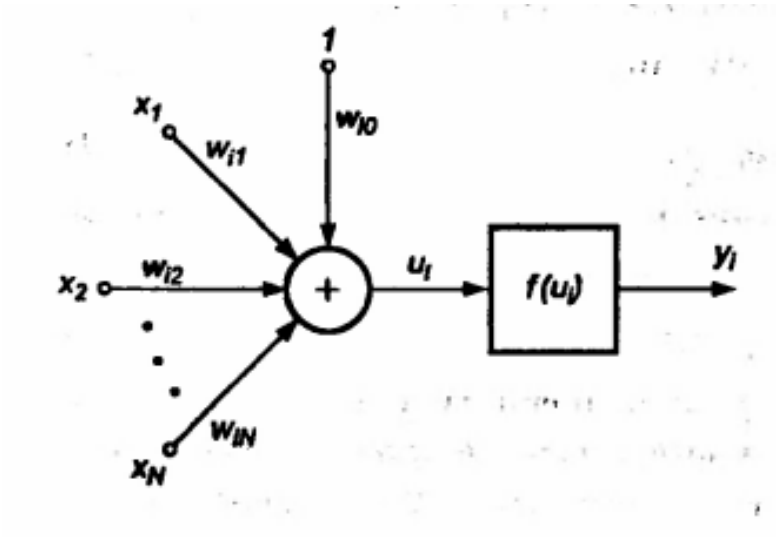

Рисунок 1.2 – Модель сигмоїдального нейрону

Уніполярна функція, як правило, представляється формулою:

$$f(x) = \frac{1}{1+e^{-\beta x}}, \tag{1.9}$$

тоді як біполярна функція задається у вигляді:

$$f(x) = \tanh(\beta x). \tag{1.10}$$

У цих формулах параметр β підбирається користувачем. Його значення має вплив на форму функції активації. На рисунку 1.2 представлені графіки сигмоїдальної функції від змінної x для різних значень β, причому на рисунку 1.3а показана уніполярна, а на рисунку 1.3б – біполярна функція.

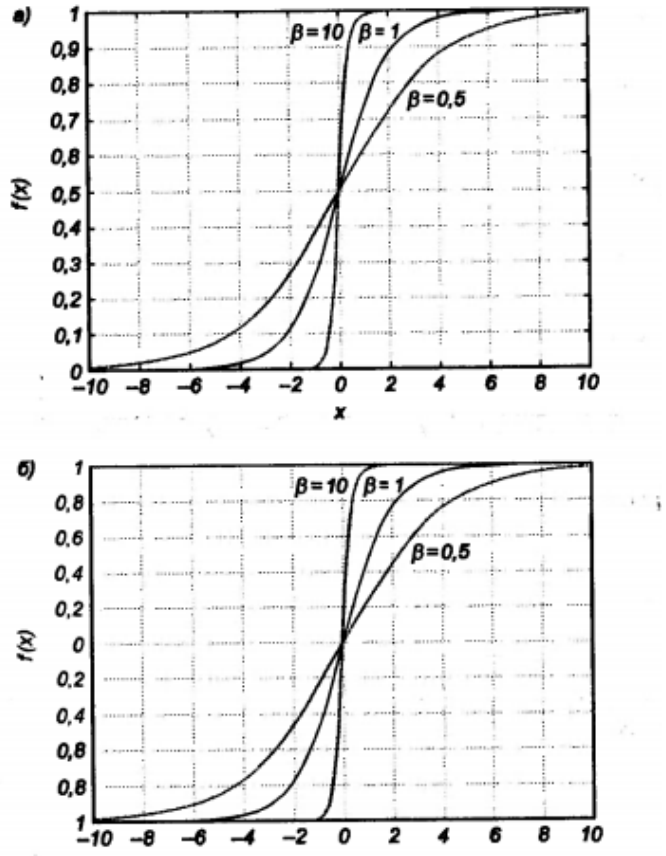

Рисунок 1.3 – Графік сигмоїдальної функції: а) уніполярної; б) біполярної при різних значеннях β

Графіки обох функцій сильно залежать від значення β. При невеликих значеннях β графік функції недостатньо пологий, але по мірі зростання β крутизна графіка збільшується.

При $\beta \to \infty$ сигмоїда перетворюється у функцію ступінчастого типу, ідентичну функції активації персептрона. На практиці частіше всього використовується значення $\beta = 1.$

Важливою властивістю сигмоїдальної функції є її диференційність. Для уніполярної функції маємо:

$$\frac{df(x)}{dx} = \beta f(x)(1-(x)), \tag{1.11}$$

тоді як для біполярної функції:

$$\frac{df(x)}{dx} = \beta f(x)(1 - f^2(x)). \qquad (1.12)$$

Як в першому, так і в другому випадку графік змінення похідної відносно змінної x має дзвіноподібну форму, а його максимум відповідає значенню $x = 0$ (рисунок 1.4).

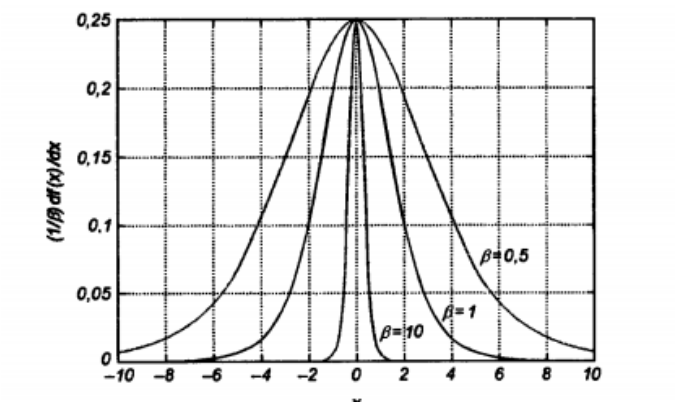

Рисунок 1.4 – Графік похідної від сигмоїдальної функції при різних значеннях коефіцієнта β

Сигмоїдальний нейрон, як правило, навчається зі вчителем по принципу мінімізації цільової функції, яка для одного навчаючого кортежу, а саме $< x, d >$, що визначається у вигляді:

$$E = \frac{1}{2}(y_i - d_i)^2, \qquad (1.13)$$

де

$$y_i = f(u_i) = f\left(\sum_{j=0}^{N} w_{ij} x_j\right). \qquad (1.14)$$

Функція $f(u_i)$ є сигмоїдальною, x – вхідний вектор, що дорівнює значенню виразу $x = \left[x_0, x_j, ..., x_N\right]^T$ з $x_0 = 1$ при наявності поляризації і $x_0 = 0$ при її відсутності, а d_i відповідне йому очікуване значення на виході i-го нейрона. Застосування неперервної функції активації дозволяє використовувати при навчанні градієнтні методи. Найпростіше реалізувати метод найшвидшого спуску, у відповідності з яким уточнення вектору ваг, що виражаються як $w = \left[w_{i0}, w_{i1}, ..., w_{iN}\right]^T$ та проводиться у напрямку негативного градієнту цільової функції. Якщо ця функція визначена виразом (1.1), j-а складова градієнту має вигляд:

$$\nabla_j E = \frac{dE}{dw_{ij}} = e_i x_j \frac{df(u_i)}{du_i}, \qquad (1.15)$$

де $e_i = (y_i - d_i)$ означає різницю між фактичним та очікуваним результатом вихідного сигналу нейрона.

Якщо ввести позначення $\delta_i = e_i \frac{df(u_i)}{u_i}$, то можна отримати вираз, що визначає складову градієнта у вигляді:

$$\nabla_j E = \delta_i x_j. \qquad (1.16)$$

Значення вагових коефіцієнтів можуть уточнювати дискретним способом:

$$w_{ij}(t+1) = w_{ij}(t) - \eta\delta_i x_j, \tag{1.17}$$

де η – це коефіцієнт навчання, значення якого, як правило, обирають емпіричним шляхом з інтервалом (0, 1). чи вирішенням рівняння:

$$\frac{dw_{ij}}{dt} = -\mu\delta_i x_j \ , \tag{1.18}$$

в якому константа μ виступаю в ролі аналогічній значенню η у рівнянні у формулі (1.17).

Два останніх рівняння визначають алгоритм навчання нейрона. На ефективність навчання впливає підбір коефіцієнта навчання. В існуючих додатках його величина може задаватися константою або бути змінною величиною, значення якої змінюється в процесі навчання адаптивним способом або підбирається на кожному кроці за принципом спрямованої мінімізації.

Найбільш ефективним, але одночасно і найбільш трудомістким вважається метод спрямованої мінімізації, згідно до якого коефіцієнт навчання підбирається на кожному кроці шляхом мінімізації цільової функції від однієї змінної в напрямку найшвидшого зменшення значення цієї цільової функції Необхідно підкреслити, що застосування градієнтного методу для навчання нейрона гарантує досягнення тільки локального мінімуму. В разі поліноміальної цільової функції знайдений локальний мінімум може бути досить далекий від глобального мінімуму. Вихід з околиці локального мінімуму при використанні простого алгоритму найшвидшого спуску неможливий. Результативним може виявитися навчання з моментом або розкидом [1], [3]. У цьому методі процес уточнення ваг визначається не тільки інформацією про градієнт функції, а також і фактичним трендом змінення ваг. Подібний спосіб навчання може

бути заданий наступним математичним виразом, що визначає збільшення ваг:

$$\Delta w_{ij}(t+1) = -\eta \delta_i x_j + \alpha \Delta w_{ij}(t)\,, \qquad (1.19)$$

в якому перший член відповідає звичайному методу найшвидшого спуску, тоді як другий член, момент, означає останню зміну ваг та не залежить від фактичного значення градієнту.

Значення коефіцієнта моменту α, як правило, обирається з інтервалу між 0 та 1. Слід звернути увагу на те, що вплив моменту на підбір ваг збільшується зі збільшенням значення α. Подібний вплив суттєво збільшується при наближенні до локального мінімуму, де значення градієнта прагне до 0. В цьому випадку можливі такі зміни у вагових коефіцієнтах, що призводять до збільшенню значення цільової функції та виходу її за пороги області локального мінімуму. Така ситуація стосовно апроксимуючої мережі (яка виконує апроксимацію вхідних даних) ілюструється на рисунку 1.5. Зазначені на графіку точки відповідають значенням цільової функції, що отримуються на кожному кроці навчання. Локальний мінімум P_1 був покинутий завдяки дії моменту. Це дозволило знайти в точці P_2 новий мінімум з меншим значенням цільової функції, який виявився більш підходящим з позицій наближення фактичного значення y_i до очікуваного значення d_i. Слід зазначити, що показник моменту не повинен домінувати в процесі навчання, так як це призведе до нестабільності (розбіжність) алгоритму. Як правило, в процесі навчання відстежується значення похибки e_i з тим, щоб не допустити його зростання понад деякого допустимого рівня, наприклад 5%.

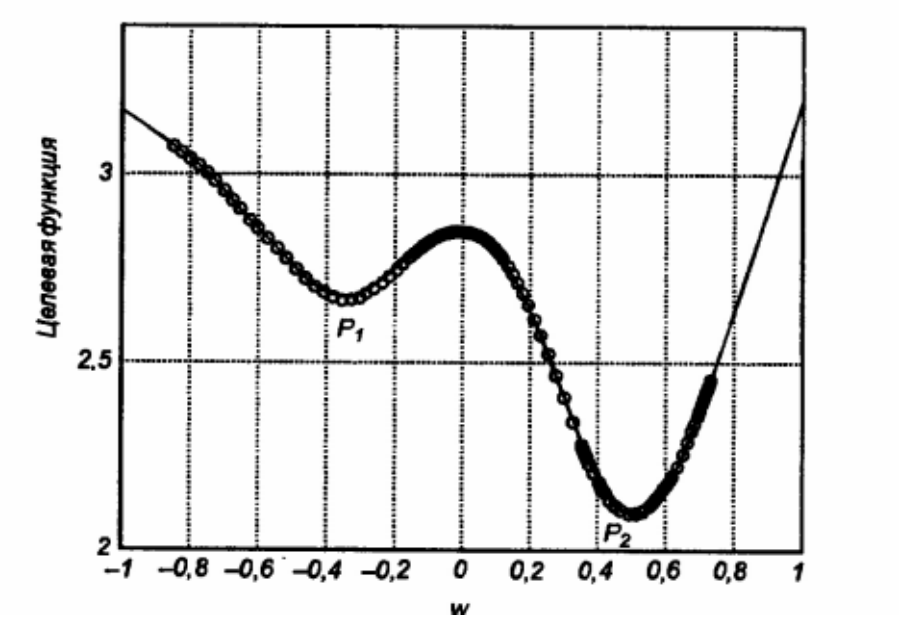

Рисунок 1.5 – Графік впливу моменту на процес навчання нейронної мереж

1.4 Нейрон типу адаліни

Модель нейрона типу адаліни була запропонована Б. Уідроу. Її структурна схема, що демонструє адаптивний спосіб підбора вагових коефіцієнтів, зображена на рисунку 1.6. Згідно методу вагового додавання сигналів нейрон типу адаліни аналогічний до попередньо представлених моделей нейронів.

Функція активації має тип signum, тобто:

$$y_i(u_i) = \begin{cases} 1 & для\ u_i > 0, \\ -1 & для\ u_i \le 0. \end{cases} \qquad (1.20)$$

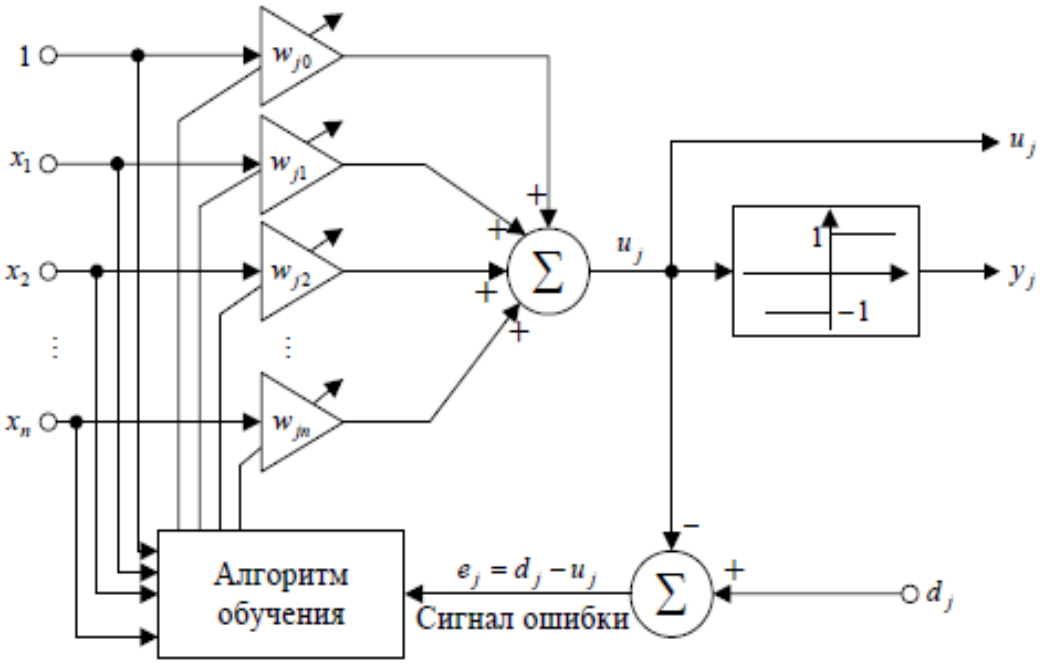

Рисунок 1.6– Структура схема нейрона типу адаліни

Адаптивний підбір вагових коефіцієнтів відбувається в процесі мінімізації квадратичної помилки, що визначається як:

$$E(w) = \frac{1}{2}e_i^2 = \frac{1}{2}\left[d_i - \left(\sum_{j=0}^{N} w_{ij} x_j \right) \right]^2 . \qquad (1.21)$$

Слід звернути увагу, що не дивлячись на нелінійний характер моделі, у цільовій функції присутні тільки лінійні члени, що представляють собою суму зважених вхідних сигналів. У зв'язку з виконанням умов неперервності цільової функції стало можливим застосування алгоритму градієнтного навчання. Як і в ситуації з сигмоїдальним нейроном, у алгоритмі Уідроу для мінімізації цільової функції застосовується метод найшвидшого спуску. Значення вагових коефіцієнтів можуть уточнюватися дискретним шляхом:

$$w_{ij}(t+1) = w_{ij}(t) + \eta e_i x_i \ , \qquad (1.22)$$

чи аналогічним способом – шляхом вирішення рівняння вигляду:

$$\frac{dw_{ij}}{dt} = -\mu \delta_i x_j \ . \qquad (1.23)$$

Не дивлячись на те, що адалайн має на виході нелінійний блок типу signum, он все ж таки вважається лінійним елементом, оскільки у визначенні цільової функції не лінійність відсутня, а підбір ваг відбувається так, ніби ніякої не лінійності не існує. Нейрон типу адаліни має відносно просту практичну реалізацію [7], [8], [9] як у випадку аналогового підходу на основі рівняння (1.23), так і в дискретному варіанті на базі виразу (1.22). Основні компоненти моделі в першому випадку – це обчислювальні елементи інтегратори та суматори, тоді як у другому випадку – це елементи затримки, описувані оператором запізнювання z^{-1}, і також інтегратори та суматори. Обидві адалайн-моделі можуть служити базою для комп'ютерного моделювання нейрона цього тіпа.

Нейрони типу «адалайн» завжди використовуються групами, утворюючи шари, звані мадалайн. Кожен нейрон навчається за правилом адалайн. Вихідні сигнали окремих нейронів можуть формуватися різними способами. Б. Уідроу [9] запропонував три базові типи міжнейронних з'єднань: OR, AND, мажоритарне. На рисунку 1.7 зображені схеми таких з’єднань. Конкретні сигнали y_i сумуються з урахуванням порогового значення, встановленого роздільно для кожного типу зв’язків

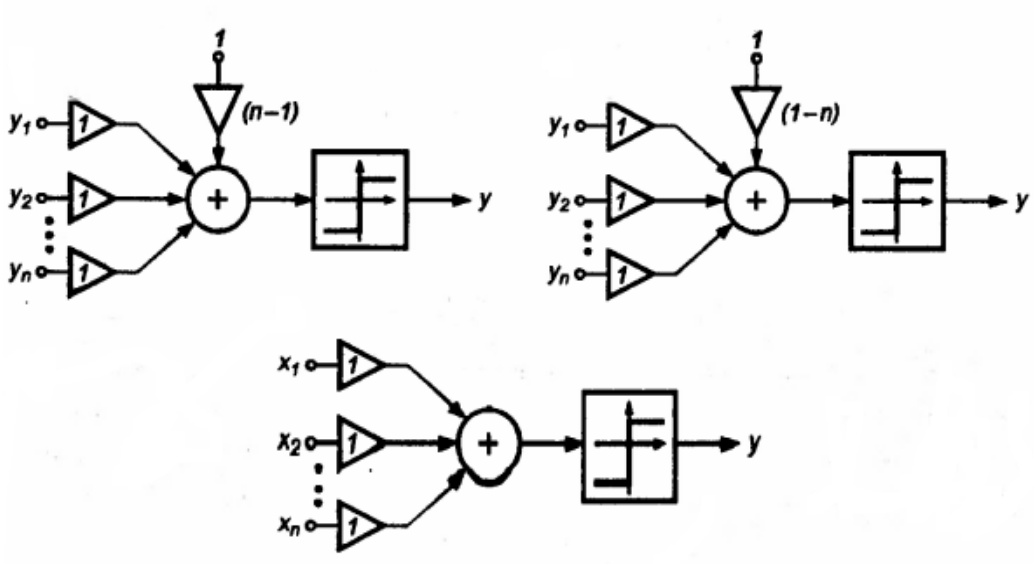

Рисунок 1.7 – Мережа мадалайн з виходами типу OR, AND та мажоритарний

1.5 Модель нейрона Хебба

Д. Хебб в процесі дослідження нервових клітин [1], [10] помітив, що зв'язок між двома клітинами посилюється, якщо обидві клітини пробуджуються (стають активними) в той самий момент часу. Якщо j-а клітина з вихідним сигналом y_i пов'язана з i-тою клітиною, що має вихідний сигнал y_i, зв'язком з вагою w_{ij}, то на силу зв'язків цих клітин впливають значення вихідних сигналів y_i та y_j.

Д. Хебб запропонував формальне правило, в якому відобразилися результати його спостережень. У відповідності з правилом Хебба [10], ваги нейрона змінюється пропорційно до добутку його вхідного та вихідного сигналів:

$$\Delta w_{ij} = \eta y_j y_i \ , \tag{1.24}$$

де η – це коефіцієнт навчання, значення якого обирається в інтервалі (0, 1).

Правило Хебба може застосовуватися для нейронних мереж різних типів с різноманітними функціями активації моделей окремих нейронів.

Структурна схема нейрона Хебба, представлена на рисунку 1.8, відповідає стандартній формі моделі нейрона.

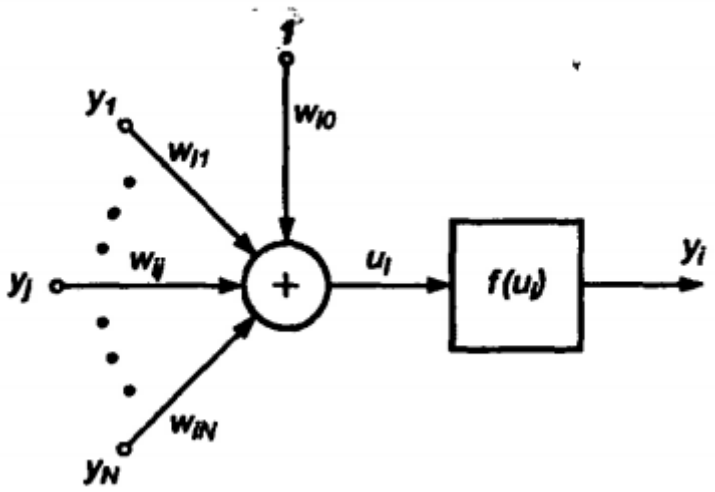

Рисунок 1.8 – Нейрон Хебба

Зв’язок з вагами w_{ij}, спосіб підбору значення якого задається відношенням (1.24), з’єднує вхідний сигнал y_j з суматором i-го нейрона, що виробляэться вихідним сигналом y_i. Навчання нейрону за правилом Хебба може проводитися як з вчителем, так і без нього. У другому випадку у правилі Хебба використовується фактичне значення y_i вихідного сигналу нейрона. При навчанні з вчителем замість значення вихідного сигналу y_i використовується очікувана від цього нейрону реакція d_i. В цьому випадку правило Хебба записується у вигляді:

$$\Delta w_{ij} = \eta y_j d_i \ . \qquad (1.25)$$

Правило Хебба характеризується, що в результаті його використання ваги можуть приймати довільно великі значення, оскільки в кожному циклі навчання відбувається підсумовування поточного значення ваги і його збільшення Δw_{ij}:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \ . \tag{1.26}$$

Один зі способів стабілізації процесу навчання по правилу Хебба полягає в обліку для уточнення ваги останнього значення w_{ij}, зменшеного на коефіцієнт забування γ [1]. При цьому правило Хебба представляється у вигляді:

$$w_{ij}(t+1) = w_{ij}(t)(1-\gamma) + \Delta w_{ij} \ . \tag{1.27}$$

Значення коефіцієнта забування γ обирається, як правило, з інтервалу між 0 та 1 і частіше всього складає деякий процент від коефіцієнта навчання, що позначається η. Застосування великих значень γ призводить до того, що нейрон забуває значну частину того, чого він навчився в минулому. Рекомендовані значення коефіцієнта забування $-\gamma < 0{,}1$, при яких нейрон зберігає більшу частину інформації, накопиченої в процесі навчання, і отримує можливість стабілізувати значення ваг на певному рівні. Навчання Хебба вважається навчанням асоціативного типу.

При навчанні лінійного нейрона по правилу Хебба стабілізація не відбувається навіть при введенні коефіцієнта забування. Вихідний сигнал нейрона визначається виразом:

$$y = \sum_j w_j x_j = w^T x = x^T w \ . \tag{1.28}$$

Згідно до правила Хебба:

$$\Delta w = \eta x y, \tag{1.29}$$

підставити вираз (1.6) до формули (1.7) та обрати для спрощення $\eta = 1$, то отримаємо приріст вектору ваг Δw у вигляді

$$\Delta w = Cw, \tag{1.30}$$

де $C = xx^T$ – це матриця кореляції, яка за визначенням є симетричною та позитивно напів-визначеною, тож має власні натуральні і невід'ємні значення.

При виконанні операцій, що описуються залежністю (1.8) і повторюваних на позитивно напів-визначеної матриці С, процес стає розбіжним, а значення компонентів вектору w прагнуть до нескінченності.

Нестабільність правила Хебба в процесі навчання можна усунути обмеженням вектору ваг за рахунок операції ренормалізації, тобто таким підбором пропорційного коефіцієнта α на кожному кроці навчання, щоб вираз дорівнював $w' = \alpha w$ при $\|w\| = 1$. Цей метод досить складний і вимагає додаткових трудовитрат на етапі навчання.

Е. Ойя [11] модифікував правило Хебба таким чином, що без ренормалізації процесу навчання вектор ваг самостійно наближується до 1. Згідно з правилом Ойі уточнення ваг проводиться згідно до виразу:

$$\Delta w = \eta y (x_i - y w_i) \ . \tag{1.31}$$

Це правило нагадує зворотне поширення, оскільки сигнал x_i змінюється зворотним сигналом, пов’язаним з вихідним сигналом y

нейрона. Для кожного окремого нейрона правило Ойя може вважатися локальним, так як в процесі модифікації x_i приймається до уваги тільки той ваговий коефіцієнт, значення якого обирається в поточний момент часу.

Доказ обмеженості ваг, що уточняються за правилом Ойя, можна отримати, замінюючи скалярний вираз (1.9) векторної формою, яка з урахуванням спрощення $\eta = 1$ у відповідності з (1.9) набуває вигляду:

$$\Delta w = Cw - (w^T Cw)w \ . \tag{1.32}$$

Стабільність процесу навчання достягається, коли при достатньому довготривалому навчанні забезпечується $\|\Delta w\| = 0$, тобто:

$$Cw = (w^T Cw)w \ . \tag{1.33}$$

Якщо власне значення кореляційної матриці C означити λ, а вектор ваг w підбирати як пов'язаний з нею власний вектор, то згідно з визначенням власного значення маємо $Cw = \lambda w$. Підставляючи цей вираз в формулу (1.33), отримуємо:

$$\lambda = w^T Cw = w^T \lambda w = \lambda \left| w^2 \right| \ . \tag{1.34}$$

З (1.34) випливає, що застосування для навчання модифікованого правила Хебба призводить до обмеження модуля вектору одиницею $|w| = 1$, що забезпечує обмеженість значень вагових коефіцієнтів.

1.6 Стохастична модель нейрона

На відміну від всіх детермінованих моделей, в стохастичній моделі [1] вихідний стан нейрона залежить не тільки від зваженої суми вхідних сигналів, але і від деякої випадкової змінної, значення якої обираються при кожній реалізації з інтервалу (0,1).

В стохастичній моделі нейрона вихідний сигнал y_i приймає такі значення як ± 1 з ймовірністю $Prob(y_i = \pm 1) = 1/(1 + \exp(\mp 2\beta u_i))$, де u_i позначена зважена сума вхідних сигналів i-го нейрона, а β – це позитивна константа, та частіше всього дорівнює 1. Процес навчання нейрона в стохастичній моделі складається з наступних етапів:

– розрахунок зваженої суми $u_i = \sum_{j=0}^{N} w_{ij} x_j$ для кожного нейрона мережі;

– розрахунок ймовірності того, що y_i приймає значення ± 1 відповідно:

$$Prob(y_i = \pm 1) = \frac{1}{1 + \exp(\mp 2\beta u_i)}; \qquad (1.35)$$

– генерація значення випадкової змінної $R \in (0,1)$ та формування вихідного сигналу $y_i = \pm 1$, якщо $R < Prob(y_i = \pm 1)$ чи $y_i = \mp 1$, в іншому випадку;

– певний процес здійснюється на випадковій обраній групі нейронів, внаслідок чого їх стан змінюється відповідно до запропонованого правила;

– після фіксації стану відібраних нейронів їх вагові коефіцієнти змінюються за вживаним правилом уточнення ваг. Наприклад, при навчанні з вчителем згідно до правила Уідроу-Хоффа адаптація ваг проводиться по формулі:

$$\Delta w_{ij} = \eta x_j (d_i - y_i) \ . \tag{1.36}$$

Доведено [1], що такий спосіб підбора ваг приводить в результаті до мінімізації цільової функції, визначеної як середньоквадратична похибка:

$$E = \frac{1}{2} \sum_{k=1}^{p} \sum_{i=1}^{n} (d_i^{(k)} - y_i^{(k)})^2 \ , \tag{1.37}$$

розрахована за всіма n нейронами та p навчаючим вибіркам.

2 БАГАТОШАРОВІ НЕЙРОННІ МЕРЕЖІ ТА АЛГОРИТМИ ЇХ НАВЧАННЯ

Об'єднані між собою нейрони утворюють систему, яка в подальшому буде називатися штучної нейронної мережею (скорочено – ШНМ). В залежності від способу об'єднання нейронів вони можуть бути мережами односпрямованими або рекурентними (зі зворотним зв'язком).

Серед різних відомих видів ШНМ найбільший інтерес викликає однонаправлена багатошарова мережа заходів, що складається з нейронів сигмоїдального типу, названа багатошаровим персептроном [4], [6]. Передача сигналів в таких мережах відбувається тільки в одному напрямку від входу до виходу. Їх математичний опис відносно простий і прозорий, а результат може бути виражений у вигляді точної функціональної залежності алгебраїчного типу. Методи навчання подібних мереж також досить прості і мають нескладну практичну реалізацію. Навчання багатошарового персептрона проводиться, як правило, з учителем, а основна ідея навчання полягає в підборі кортежів $< x, d >$ в яких x – вхідний вектор, а d – відповідний йому очікуваний вихідний вектор мережі. Якщо вектори x і d нерівні між собою, мережа називається гетероасоціативною. У разі, коли вектори $x = d$, мережу називається автоасоціативною. У мережах подібного типу використовуються персептрони моделі нейронів або їх узагальнена форма у вигляді сигмоїдальної моделі.

Існують основні методи навчання подібних мереж, в тому числі алгоритм зворотного поширення помилок, методи мінімізації цільової функції, а також різні методи підбору початкових значень вагових коефіцієнтів мережі, прискорюють процес навчання і дозволяють уникнути припинення цього процесу в точках локальних мінімумів. З історичної точки зору першими були створені одношарові мережі і методи їх навчання, і тільки через багато років (з кінця сімдесятих років двадцятого століття) була запропонована ефективна методика навчання багатошарової мережі.

2.1. Одношарова мережа

Одношарову мережу утворюють нейрони, розташовані в одній площині (рисунок 2.1). Кожен i-й нейрон має поляризацію (а також безліч зв'язків з вагами w_{ij}, за якими надходять вхідні сигнали x_j. Значення ваг підбираються в процесі навчання мережі, що складається в наближенні вихідних сигналів y_i до очікуваних значенням d_i. Мірою близькості вважається значення цільової функції, також званої вартісної функцією. При використанні p навчальних векторів $< x, d >$ для навчання мережі, що включає M вихідних нейронів, цільову функцію можна визначити евклідовою метрикою виду

$$E = \frac{1}{2}\sum_{k=1}^{p} \| y^{(k)} - d^{(k)} \|^2 = \frac{1}{2}\sum_{k=1}^{p} \sum_{i=1}^{M} (y_i^{(k)} - d_i^{(k)})^2. \qquad (2.1)$$

Вихідні сигнали нейрона y_i є функціями ваги мережі w_{ij}, значення яких уточняються в процесі навчання за критерієм мінімізації цільової функції.

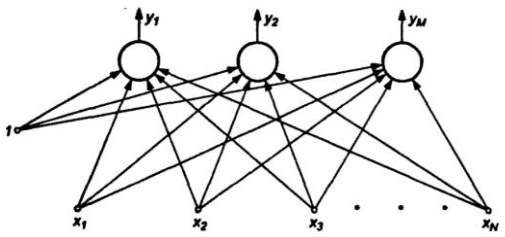

Рисунок 2.1 – Структура одношарової сигмоїдальної нейронної мережі

Розташовані на одному рівні нейрони функціонують незалежно один від одного, тому можливості такої мережі обмежуються властивостями

окремих нейронів. Ваги нейронів утворюють певний простір рішень. Слід враховувати, що кожен нейрон реалізує функціональне відображення, що позначається $y_i = f(\sum_{j=0}^{N} w_{ij} x_j)$. Беручи до уваги, що сигмоїдальна функція являє собою безперервний аналог одноступінчатої порогової функції, можна помітити, що вихідний сигнал нейрона (значення 1 або 0) буде залежати від знаку виразу $\sum_{j=0}^{N} w_{ij} x_j$. Це рівняння лінійно відносно ваг w_{ij}. Вихідний сигнал y_i при фіксованих значеннях ваг залежить від розташування вхідного вектору x, який визначає гіперплоскість, що розділяє багатовимірний простір на два підпростори. Тому завдання класифікації (приписування значення 0 або 1 вихідному сигналу нейрона) може бути вирішена за допомогою єдиного нейрона, якщо вона відноситься до класу задач лінійної сепарації (наприклад, із застосуванням логічних функцій AND або OR). Продемонструємо обмеженість можливостей одношарових мереж на прикладі реалізації логічної функції XOR [12]. Для спрощення будемо використовувати функцію активації у вигляді одноступінчастого порога. Безліч даних для навчання логічної функції XOR представлено на рисунку 2.2.

x_1	0	0	1	1
x_2	0	1	0	1
d	0	1	1	0

Рисунок 2.2 – Сукупність даних для навчання функції XOR

Легко показати, що в цьому випадку неможливо провести єдину лінію, яка розділяє простір даних на два підпростори, з яких одне відповідало б

вхідному сигналу 1, а інше – 0 (на рисунку 2.3 заштрихована область відноситься до одного класу, а незаштриховані – до другого).

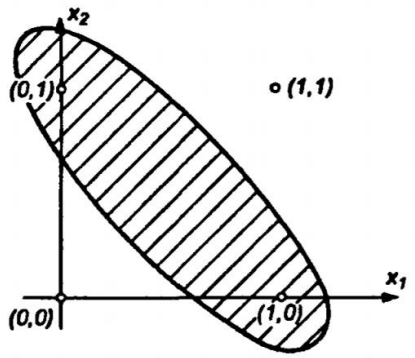

Рисунок 2.3 – Ілюстрація неможливості лінійного розділення навчальних даних

Усередині заштрихованої області вихідний сигнал нейронів повинен дорівнювати 1, а за її межами – 0. Така умова не може бути виконано при використанні для поділу простору єдиною прямою (одного нейрона) незалежно від значень параметрів цієї прямої (ваг w_{10}, w_{11}, w_{12}). Таким чином, одношаровий персептрон не в змозі реалізувати навіть таку нескладну функцію, як XOR. Цю проблему легко вирішити шляхом розширення штучної нейронної мережі. З цією митою додаймо в шар ще один нейрон и підберемо вагу обох нейронів таким чином, щоб вони в залежності від вхідного вектору $x = [x_1, x_2]^T : u_1 = w_{11}x_1 + w_{12}x_2 + w_{10} > 0$ та $u_1 = w_{11}x_1 + w_{12}x_2 + w_{10} < 0$ (перший нейрон) і $u_2 = w_{21}x_1 + w_{22}x_2 + w_{20} > 0$ та $u_2 = w_{21}x_2 + w_{22}x_{20} + w_{20} < 0$ (другий нейрон). Підбір ваг має забезпечити розділення простора, показане на рисунку 2.4. Спільна частина підмножин, які відповідають умовам $u_1 > 0, u_2 > 0$, визначали область, відділену від інших просторів , відповідних умовам $u_1 > 0, u_2 < 0$.

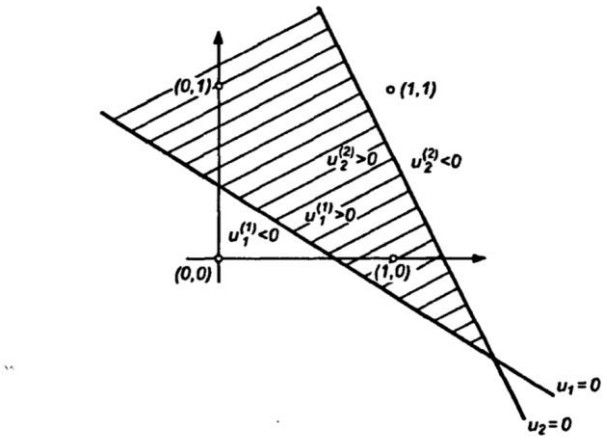

Рисунок 2.4 – Рішення проблеми нелінійного розділення

Додаванням на виході мережі ще одного шару, що складається з єдиного нейрона, можна реалізувати функцію логічного підсумовування, що виділяє загальну частину підмножин $u_1 > 0, u_2 > 0$.Остаточна структура ШНМ, що виконує функцію XOR, представлена на рисунку 2.5. Слід зазначити, що додавання в мережу додаткового шару дозволило вирішити проблему неможливості лінійного поділу даних. Кожен нейрон прихованого шару здійснює додаткове лінійне поділ площині, причому межа такого розділу на області $u_i > 0$ і $u_i < 0$ залежить від значення ваги нейрона. Вихідний шар виконує відповідну лінійну комбінацію(наприклад, логічну суму) під областей, на які сукупність вхідних даних було розділено нейронами скритого шара.

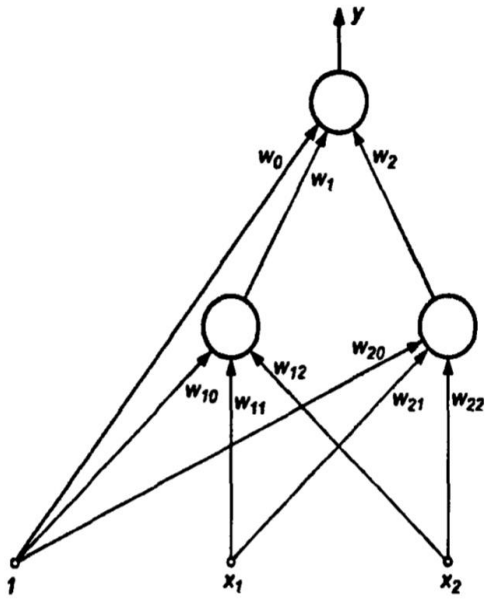

Рисунок 2.5 – Структура ШНМ, яка виконує функцію XOR.

Не дивлячись на те, що одношарова мережа має невелике практичне значення, її продовжують використовувати там, де для рішення поставленої задачі достатньо одного шара нейронів.

Вибір архітектури такої мережі досить простий. Кількість вхідних нейронів визначається розмірністю вхідного векторах, а кількість вихідних нейронів визначається розмірністю вектора d. Навчання мережі проводиться, як правило, з учителем і є точною копією навчання одиночного нейрона.

2.2. Многошаровий персептрон

2.2.1. Структура персептронної мережі

Багатошарова мережа складається з нейронів, розташованих на різних рівнях, причому, крім вхідного і вихідного шарів, є ще, як мінімум, один внутрішній, тобто прихований, шар. Як вже зазначалося в літературі,

присвяченій проблематики нейронних мереж, така нейронна система називається багатошаровим персептроном [4], [6], [13].

На рисунку 2.6 представлена мережа з одним прихованим шаром.

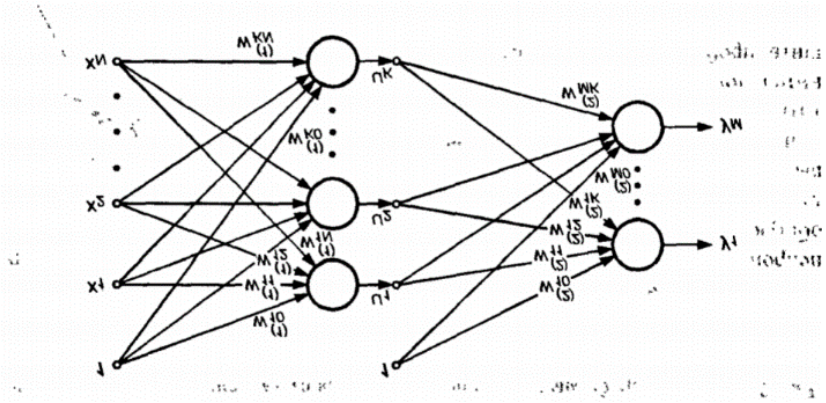

Рисунок 2.6 – Узагальнена структура двошарової сигмоїдальної нейронної мережі (з одним прихованим шаром)

Усі наступні міркування відносяться до мереж саме такого типу. Позначення сигналів і ваг також будуть відповідати цьому малюнку. Ваги нейронів прихованого шару позначимо верхнім індексом (1), а вихідного шару – верхнім індексом (2). Вихідні сигнали нейронів прихованого шару позначимо $v_j(j=1,2,...,K)$ а вихідного шару – $y_j(j=1,2,...,M)$. Приймемо, що функція активації нейронів задана в сигмоїдальній уніполярної або біполярної формі. Для спрощення опису будемо використовувати розширене позначення вхідного вектору мережі у вигляді $x=[x_0,x_1,...,x_N]^T$, де $x_0=1$ відповідає одиничному сигналу поляризації. З вектором x зв'язані два вхідних вектора мережі: вектор фактичних вихідних сигналів $y=[y_0,y_1...,y_M]^T$ і вектор очікуваних вихідних сигналів $d=[d_0,d_1,...,d_M]^T$.

Мета навчання полягає в підборі таких значень ваг $w_y^{(1)}$ та $w_y^{(2)}$ для всіх шарів мережі, щоб при заданому вхідному векторі x отримати на виході значення сигналів y_1, які з необхідною точністю будуть збігатися з

очікуваними значеннями d_1 для $i = 1,2,...,M$. Якщо розглядати одиничний поляризаційний сигнал як один з компонентів вхідного вектору x, то ваги поляризації можна додати в вектори ваг відповідних нейронів обох шарів.

При такому підході вихідний сигнал i-го нейрона прихованого шару вдається описати функцією:

$$v_j = f(\sum_{j=0}^{N} w_y^{(1)} x_j) \ , \tag{2.2}$$

в якій індекс 0 відповідає сигналу та ваг поляризації, причому $v_0 = 1, x_0 = 1$.

У вихідному шарі k-й нейрон виробляє вихідний сигнал, визначається як:

$$y_k = f\left(\sum_{i=0}^{K} w_{k_1}^{(2)} v_1\right) = f\left(\sum_{i=0}^{K} w_{k_1}^{(2)} f\left(\sum_{j=0}^{N} w_y^{(1)} x_j\right)\right). \tag{2.3}$$

З формули (2.1) випливає, що на значення вихідного сигналу впливають ваги обох шарів, тоді як сигнали, що виробляються в прихованому шарі, що не залежать від ваг вихідного шару.

2.2.2. Алгоритм зворотного поширення помилки

Алгоритм зворотного поширення помилки визначає стратегію підбору ваг багатошарової мережі із застосуванням градієнтних методів оптимізації. «Винайдений» кілька разів [6], він в даний час вважається одним з найбільш ефективних алгоритмів навчання багатошарової мережі. Його основу складає цільова функція, що формуються, як правило, у вигляді квадратичної суми різниць між фактичними і очікуваними значеннями вихідних сигналів. У разі одиничної навчальної вибірки (x, d) цільова функція визначається у вигляді:

$$E(w) = \frac{1}{2}\sum_{k=1}^{M}(y_k - d_k)^2. \quad (2.4)$$

При великій кількості навчальних вибірок $j(j = 1,2,...,p)$ цільова функція переходить в суму по всім вибіркам:

$$E(w) = \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{M}(y_k^{(j)} - d_k^{(j)})^2. \quad (2.5)$$

Уточнення ваг може проводитися після пред'явлення кожної навчальної вибірки (так званий режим «онлайн») або одноразово після пред'явлення всіх вибірок, що становлять цикл навчання (режим «офлайн»). В подальшому використовується цільова функція виду (2.3), яка відповідає актуалізації ваг після пред'явлення кожної вибірки.

Для спрощення можна вважати, що мета навчання полягає в такому визначенні значень ваг нейронів кожного шару мережі, щоб при заданому вхідному векторі отримати на виході значення сигналів y_i збігаються з необхідною точністю з очікуваними значеннями d_i при $i = 1,2,...,M$.

Навчання мережі з використанням алгоритму зворотного поширення помилки проводиться в кілька етапів. На першому з них пред'являється навчальна вибірка x і розраховуються значення сигналів відповідних нейронів мережі. При заданому векторі x визначаються спочатку значення вихідних сигналів v_i прихованого шару, а потім значення y_i нейронів вихідного шару. Для розрахунку застосовуються формули (2.1) і (2.2). Після отримання значень вихідних сигналів y_i стає можливим розрахувати фактичне значення цільової функції $E(w)$, заданої виразом (2.3). На другому етапі мінімізується значення цієї функції.

Якщо прийняти, що цільова функція неперервна, то найбільш ефективними способами навчання виявляються градієнтні методи

оптимізації, згідно з якими уточнення вектору ваг (навчання) проводиться за формулою:

$$w(k+1) = w(k) + \Delta w, \tag{2.6}$$

$$\Delta w = \eta p(w), \tag{2.7}$$

де η — коефіцієнт навчання,

$p(w)$ — напрям в багатовимірному просторі w.

Навчання багатошарової мережі із застосуванням градієнтних методів вимагає визначення вектору градієнта щодо ваг всіх верств мережі, що необхідно для правильного вибору напрямку $p(w)$. Ця задача має очевидне рішення тільки для ваг вихідного шару. Для інших верств створена спеціальна стратегія, яка в теорії штучних нейронних мереж називається алгоритмом зворотного поширення помилки [1], [6], які ототожнюються, як правило, з процедурою навчання мережі. Відповідно до цього алгоритмом в кожному циклі навчання виділяються наступні етапи [6]:

– аналіз нейронної мережі в прямому напрямку передачі інформації при генерації вхідних сигналів, що складають черговий вектор x. В результаті такого аналізу розраховуються значення вихідних сигналів нейронів прихованих шарів і вихідного шару, а також відповідні похідні $\frac{df(u_i^{(1)})}{du_i^{(1)}}, \frac{df(u_i^{(2)})}{du_i^{(2)}}, \ldots \frac{df(u_i^{(m)})}{du_i^{(m)}}$ функцій активації кожного шару (m – кількість шарів мережі);

– створення мережі зворотного поширення помилок шляхом зміни напрямків передачі сигналів, заміна функцій активації їх похідними і подача на колишній вихід (а зараз – вхід) мережі збудження у вигляді різниці між фактичним і очікуваним значенням. Для певної таким чином мережі необхідно розрахувати значення необхідних зворотних різниць;

– уточнення ваг (навчання мережі) проводиться за запропонованими вище формулами на основі результатів, отриманих в п. 1 і 2, для оригінальної мережі і для мережі зворотного поширення помилки;

– процес слід повторити для всіх навчальних вибірок, продовжуючи його аж до виконання умови зупинки алгоритму. Дія алгоритму завершується в момент, коли норма градієнта впаде нижче апріорі заданого значення ε, що характеризує точність процесу навчання.

Базові формули і їх модифікації для конкретних типів нейронних мереж вважаються класичними для теорії нейронних мереж. З цієї причини ми розглянемо тільки умови, які стосуються мережі з одним прихованим шаром.

Як і раніше, кількість вхідних вузлів позначимо літерою *N*, кількість нейронів в прихованому шарі *K*, а кількість нейронів у вихідному шарі *M*. Будемо використовувати сигмоїдальну функцію активації цих нейронів. Основу алгоритму становить розрахунок значення цільової функції як квадратичної суми різниць між фактичними і очікуваними значеннями вихідних сигналів мережі. У разі одиничної навчальної вибірки (x,d) цільова функція задається формулою (2.2), а для безлічі навчальних вибірок $j(j=1,2...,p)$ – формулою (2.3). Для спрощення будемо використовувати цільову функцію виду (2.2), яка дозволяє уточнювати ваги після пред'явлення кожної навчальної вибірки.

Ця функція визначається виразом:

$$E=\frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{ki}^{2} v_i\right)-d_k\right]^2=\frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} w_{ki}^{(2)} f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_i\right)\right)-d_k\right]^2. \qquad (2.8)$$

Конкретні компоненти градієнта розраховуються диференціюванням залежності (2.5). В першу чергу підбираються ваги нейронів вихідного шару. Для вихідних ваг отримуємо:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}} v_j, \tag{2.9}$$

де $u_i^{(2)} = \sum_{j=0}^{K} w_{ij}^{(2)} v_j$.

Якщо ввести позначення $\delta_i^{(2)} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}}$, то відповідний компонент градієнта щодо ваг нейронів вихідного шару можна представити у вигляді:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \partial_i^{(2)} v_j. \tag{2.10}$$

Компоненти градієнта щодо нейронів прихованого шару визначаються за тим же принципом, проте вони описуються інший, більш складної залежністю, наступної з існування функції, заданій у вигляді:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k)\frac{dy_k}{dv_i}\frac{dv_k}{dw_{ij}^{(1)}}. \tag{2.11}$$

Після конкретизації окремих складових цього виразу отримуємо:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k)\frac{df(u_k^{(2)})}{du_k^{(2)}} w_{ki}^{(2)} \frac{df(u_k^{(1)})}{du_i^{(1)}} x_j. \tag{2.12}$$

Якщо ввести позначення:

$$\partial_i^{(1)} = \sum_{k=1}^{M} (y_k - d_k)\frac{df(u_k^{(2)})}{du_k^{(2)}} w_{ki}^{(2)} \frac{df(u_k^{(1)})}{du_i^{(1)}} x_j, \tag{2.13}$$

то отримаємо вираз, що визначає компоненти градієнта щодо ваг нейронів прихованого шару у вигляді:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \partial_i^{(1)} x_j . \tag{2.14}$$

В обох випадках (формули (2.6) і (2.7)) опис градієнта має аналогічну структуру і представляється добутком двох сигналів: перший відповідає початковому вузлу даної зваженої зв'язку, а другий – величиною похибки, перенесеної на вузол, з яким цей зв'язок встановлено. Визначення вектору градієнта дуже важливо для подальшого процесу уточнення ваг. У класичному алгоритмі зворотного поширення помилки фактор $p(w)$ врахований у вираженні (2.2), задає напрям негативного градієнта, тому:

$$\Delta w = -\eta \nabla E(w), \tag{2.15}$$

2.3. Потокові графи і їх застосування для генерації градієнта

На основі методу потокових графів вдається побудувати дуже прості правила формування компонентів градієнта, які мають постійну структуру, не залежну від складності мережі. При цьому базу таких правил складають співвідношення, отримані в результаті аналізу чутливості мережі методом сполучених елементів. У теорії систем [3] під повною чутливістю об'єкта розуміється похідна будь-якого циркулюючого в ньому сигналу щодо значень ваг, яка може бути розрахована на підставі знань про сигнали, що поширюються по звичайному графу (позначається *G*) і парному з ним графу, що позначається $\check{G}$. Граф $\check{G}$ визначається як вихідний граф G, в якому спрямованість всіх дуг змінена на протилежну. Лінійна дуга графа G і відповідна їй дуга сполученого графа $\check{G}$ мають ідентичні опису. У разі нелінійної зв'язку $f(x,k)$, де x – вхідний сигнал, а k – параметр, відповідна їй дуга графа $\check{G}$ лінеаризується з коефіцієнтом $\beta = \dfrac{\partial f(x,k)}{\partial x}$, розрахованим для фактичного вхідного сигналу x графа *G*.

Як показано в роботах [3], [8], [14], метод розрахунку чутливості нейронної мережі з використанням потокових графів заснований на аналізі вихідного графа *G* і сполученого з ним графа $\check{G}$ при порушенні останнього одиничним сигналом, що подається на вхід $\check{G}$ (відповідний виходу *G*). Чутливість графа *G* щодо параметрів дуг цього графа до довільного вхідного сигналу v_0 можна виразити таким чином:

– для лінійної дуги w_y графу *G*

$$\frac{dv_0}{dw_y} = v_j v_i, \tag{2.16}$$

де w_y – це коефіцієнт посилення лінійної дуги, спрямованої від j-го вузла до i-му;

v_j позначає сигнал j-го вузла графа G,

v_i – сигнал,

i-го вузла сполученого графа $\check{G}$, для якого в якості вхідного сигналу задається значення $v_0 = 1$;

– для нелінійної дуги графа G, об'єднуючою i-й и k-й вузли та описуваної функцією $v_k = f_{ki}(v_i, K)$, чутливість щодо параметра K визначається виразом:

$$\frac{dv_0}{dK} = v_k \frac{\partial f_{kl}(v_l, K)}{\partial K}, \tag{2.17}$$

де $\frac{\partial f_{kl}(v_l, K)}{\partial K}$ розраховується для сигналу v_l l-го вузла графа G.

Позначимо w вектор оптимізованих параметрів (ваг w_i) системи, представленої графом G, $w = [w_1, w_2 ..., w_n]^T$, а $E(w)$ – цільову функцію. Тоді градієнт $\nabla E(w)$, скорочено позначається як $g(w) = \nabla E(w)$, можна визначити у вигляді:

$$g(w) = \left[\frac{\partial E(w)}{\partial w_1}, \frac{\partial E(w)}{\partial w_2}, ..., \frac{\partial E(w)}{\partial w_n} \right]^T. \tag{2.18}$$

В цьому виразі для позначення елементів вектору w використаний один індекс ($i = 1, 2, ..., n$). Якщо уявити цільову функцію у формі, яка враховує тільки одну навчальну вибірку:

$$E(w) = \frac{1}{2}\sum_{i=1}^{M}(y_i - d_i)^2, \qquad (2.19)$$

де d_i; позначено очікуване значення i-го вихідного нейрона, $i = 1,2,...,M$, то градієнт цільової функції набирає вигляду:

$$g(w) = \left[g_1(w), g_2(w), ..., g_n(w)\right]^T, \qquad (2.20)$$

в якій

$$g_k(w) = \sum_{i=1}^{M}(y_i - d_i)\frac{\partial y_i}{\partial w_k}. \qquad (2.21)$$

Для завдання вектору градієнта також необхідні похідні вихідних сигналів y_i графа щодо ваг w_k (показники їх чутливості), помножені на величину похибки ($y_i - d_i$). Завдяки використання методів теорії графів всі ці операції (в тому числі і підсумовування) можна виконати за один крок за допомогою вихідного графа *G* і сполученого з ним графа $\check{G}$ при дотриманні відповідних умов збудження графа $\check{G}$. Як показано в [14], внаслідок лінійності сполученого графа всі ці операції можуть бути реалізовані автоматично в разі, коли в зв'язаному графі замість одиничних збуджень генеруються сигнали у вигляді різниць між фактичними y_i і очікуваними d_i значеннями вихідних сигналів. спосіб формування сполученого графа $\check{G}$ і методика його порушення для автоматичного розрахунку вектору градієнта на основі аналізу лише двох графів С і $\check{G}$ представлені на рисунку 2.7.

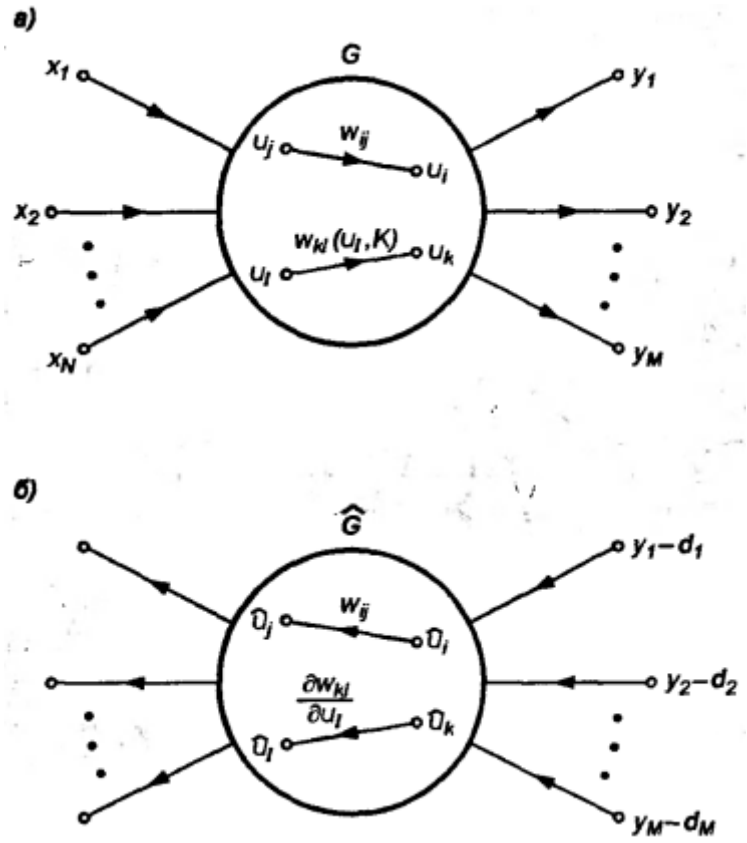

Рисунок 2.7 – Ілюстрація застосування способу формування і збудження сполученого графа: а) вихідний граф *G*; б) споряджений граф Ğ

При заміні всіх одиничних збуджень в $\check{G}$ на $(y_i - d_i)$ будь-який компонент вектору градієнта $g_k(w)$ може бути розрахований за відповідними сигналами вихідного графа *G* і сполученого з ним графа $\check{G}$ точно так же, як і при визначенні звичайної чутливості. Для лінійної дуги графа *G*, описуваної вагою w_{ij} формула має вигляд:

$$\frac{\partial E(w)}{\partial w_{ij}} = v_j v_i . \qquad (2.22)$$

Для нелінійної дуги графа G, яка описана функцією $w_{ki}(v_l, K)$ отримуємо:

$$\frac{\partial E(w)}{\partial K} = v_k \frac{\partial w_{ki}(v_l, K)}{\partial K}. \qquad (2.23)$$

Представлені вираження застосовні для будь-яких систем (лінійних, нелінійних, рекурентних та інших). Вони практично застосовуються для аналізу односпрямованих багатошарових нейронних мереж, описуваних потоковим графом проходження сигналів.

Розглянемо зображену на рисунку 2.8а типову багатошарову мережу, що складається з m шарів з довільною безперервною функцією активації нейронів. Кількість нейронів у кожному шарі будемо позначати $K_j (j = 1, 2, ..., m)$ причому останній шар є вихідним шаром мережі, що містить $K_m = M$ нейронів. Вихідні сигнали нейронів в конкретних шарах позначимо $v_j^{(k)}$, причому для останнього шару $v_j^{(m)} = y_j$.

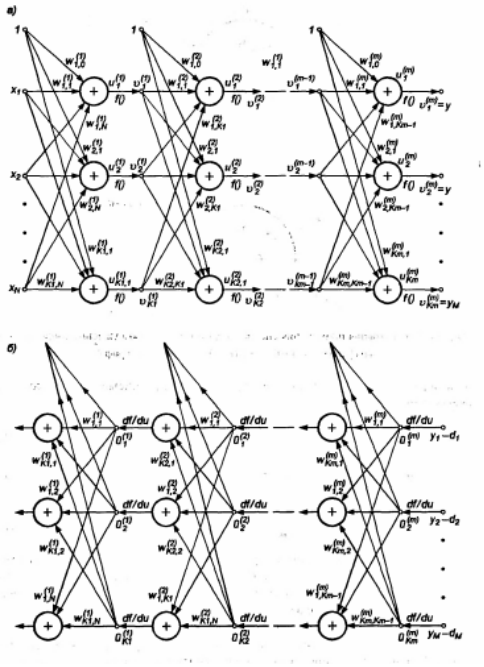

Рисунок 2.8 Ілюстрація застосування методу сполучених графів для генерації вектору градієнта однонаправленной багатошарової мережі: а) вихідний граф мережі; б) пов'язаний граф

Для визначення компонентів градієнта щодо ваг конкретних шарів мережі будемо застосовувати формулювання, пов'язані з парним графу. На рисунку 2.8б представлений пов'язаний граф мережі, яку можна детальніше розглянути на рисунку 2.8а,. Сполучений граф збуджується різницями між фактичними y_i і очікуваними d_i значеннями вихідних сигналів. Нелінійні дуги графа *G* замінюються в зв'язаному графі Ğ похідними $\frac{df(x)}{dx}$, значення яких розраховуються роздільно для кожного шару в точках $x = u_i$. Якщо, наприклад, функція активації нейронів має сигмоїдальну уніполярного форму $f(x) = \frac{1}{1+\exp(-x)}$ то $\frac{df(x)}{dx} = f(x)(1 - f(x))$ розраховується безпосередньо на основі відомого значення сигмоїдальної функції в точці x і не вимагає ніяких додаткових обчислень.

Спираючись на запропонований алгоритм визначення градієнта методами теорії графів, можна розрахувати конкретні компоненти вектору градієнта $d(w)$ для будь-якого шару нейронів: для вихідного шару , для k-го прихованого шару, для першого прихованого шару:

$$\frac{\partial E(w)}{\partial w_{ij}^{(m)}} = v_j^{(m-1)} u_i^{(m)}, \tag{2.24}$$

$$\frac{\partial E(w)}{\partial w_{ij}^{(k)}} = v_j^{(k-1)} u_i^{(k)}, \tag{2.25}$$

$$\frac{\partial E(w)}{\partial w_{ij}^{(m)}} = v_j^{(m-1)} u_i^{(m)}. \tag{2.26}$$

З наведених формул видно, що їх структури (при використанні відповідних позначень сигналів) абсолютно ідентичні незалежно від того, в якому шарі нейронів знаходиться враховується вага. Сформульоване

правило є надзвичайно простим з прикладної точки зору, оскільки для розрахунку будь-якого компонента градієнта необхідно знати тільки два сигнали: від вузла, з якого виходить зважена дуга в оригінальному графі, і від вузла, з якого виходить зважена дуга в зв'язаному графі. У цьому сенсі правило розрахунку може вважатися локальним.

Ще однією важливою перевагою графічного методу, крім значного спрощення обчислювальних процедур, вважається можливість обліку рівності значень різних ваг мережі [3]. Якщо, наприклад, вага зі значенням ваг w відноситься до дуги w_{kl}, що з'єднує i-й і j-й вузли в напрямку від j до i), і до дуги w_{kl}, що з'єднує k-й і l-й вузли (в напрямку від l до k), то легко помітити, що вага w буде присутній в двох різних позиціях виразу, що визначає цільову функцію. Згідно з правилом диференціювання складовою функції її похідна представляється у вигляді суми похідних відносно w_{ij} та w_{kl}. Отже,

$$\frac{\partial E(w)}{\partial w} = \frac{\partial E}{\partial w_{ij}} + \frac{\partial E}{\partial w_{kl}}. \qquad (2.27)$$

З урахуванням міркувань щодо сполученого графа при введенні уніфікованих позначень сигналів для будь-яких вузлів у вигляді v ($\hat{v}$) з відповідними індексами отримуємо остаточний вираз:

$$\frac{\partial E(w)}{\partial w} = v_j u_i + v_l u_k. \qquad (2.28)$$

Як випливає з формули (2.11), облік рівності окремих ваг ($w_{ij} = w_{kl}$) не тільки не ускладнює загальну розрахункову формулу, але, навпаки, спрощує її за рахунок зменшення кількості змінних. Необхідно відмітити, що збігаються ваги можуть лежати як в одному і тому ж, так і в абсолютно

різних шарах. Сутність формули (2.11) при цьому абсолютно не змінюється. У цьому полягає найважливіше відмінний метод генерації градієнта, заснованого на потокових графах поширення сигналів, від класичного підходу, широко представленого в світовій літературі [1], [6].

2.4. Градієнтні алгоритми навчання мережі

2.4.1. Основні положення навчання мереж

Завдання навчання нейронної мережі будемо розглядати на даному етапі як вимога мінімізувати апріорі певну цільову функцію $E(w)$. При такому підході можна застосовувати для навчання алгоритми, які в теорії оптимізації вважаються найбільш ефективними. До них, без сумніву, відносяться градієнтні методи, чию основу складає виявлення градієнта цільової функції. Вони пов'язані з розкладанням цільової функції $E(w)$ в ряд Тейлора в найближчій околиці точки наявного рішення w. У разі цільової функції від багатьох змінних $(w=[w_1, w_2, ..., w_n]^T)$ таке подання зв'язується з околицею раніше певної точки (зокрема, при старті алгоритму це вихідна точка w_0) в напрямку p. Подібне розкладання описується універсальною формулою виду [15], [16]:

$$E(w+p) = E(w) + \left[g(w)\right]^T p + \frac{1}{2} p^T H(w) p + \ldots, \qquad (2.29)$$

де $g(w) = \nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots \frac{\partial E}{\partial w_n}\right]^T$ – це вектор градієнта, а симетрична квадратна матриця є матрицею похідних другого порядку, званої гесіаном.

$$H(w)=\begin{bmatrix} \dfrac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \dfrac{\partial^2 E}{\partial w_1 \partial w_n} \\ \dfrac{\partial^2 E}{\partial w_n \partial w_1} & & \dfrac{\partial^2 E}{\partial w_n \partial w_n} \end{bmatrix}. \qquad (2.30)$$

У вираженні (2.12) p грає роль направляючого вектору, що залежить від фактичних значень вектору w. На практиці найчастіше розраховуються три перших члена ряду (2.12), а наступні просто ігноруються. При цьому залежність (2.12) може вважатися квадратичним наближенням цільової функції $E(w)$ в найближчій околиці знайденої точки w з точністю, рівній локальної похибки відсіченою частини $O(h^3)$, де $h = \| p \|$. Для спрощення опису значення змінних, отримані в k-м циклі, будемо записувати з нижнім індексом k. Точкою рішення $w = w_k$ будемо вважати точку, в якій досягається мінімум цільової функції $E(w)$ та $g(w_k) = 0$, а гесіан $H(w_k)$ є позитивно певним [15], [16]. При виконанні цих умов функція в будь-якій точці, що лежить в околиці їхні, має більше значення, ніж в точці w_k, тому точка w_k є рішенням, відповідним критерієм мінімізації цільової функції.

В процесі пошуку мінімального значення цільової функції напрямок пошуку p і крок h підбираються таким чином, щоб для кожної черговий точки $w_k + 1 = \eta_k p_k$ виконувалася умова $E(w_{k+1}) < E(w_k)$. Пошук мінімуму триває, поки норма градієнта не впаде нижче апріорі заданого значення допустимої похибки або поки не буде перевищено максимальний час обчислень (кількість ітерацій).

Універсальний оптимізаційний алгоритм навчання нейронної мережі можна представити в наступному вигляді (будемо вважати, що початкове значення вектору відомо і становить $w_k = w_0$):

– перевірка збіжності і оптимальності поточного рішення w_k. Якщо точка w_k відповідає градієнтним умов зупинки процесу – завершення обчислень. В іншому випадку перейти до п.2;

– визначення вектору напрямку оптимізації p_k для точки w_k;

– вибір величини кроку η_k в напрямку p_k, при якому виконується умова $E(w_k + \eta_k p_k) < E(w_k)$;

– визначення нового рішення $w_{k+1} = w_k + \eta_k p_k$, а також відповідних йому значень $E(w)$ та $g(w_k)$, а якщо потрібно – то і $H(w_k)$.

2.4.2. Алгоритм найшвидшого спуску

Якщо при розкладанні цільової функції $E(w)$ в ряд Тейлора обмежитися її лінійним наближенням, то ми отримаємо алгоритм найшвидшого спуску. Для виконання співвідношення $E(w_{k+1}) < E(w_k)$ досить підібрати $g(w_k)^T p < 0$. Умові зменшення значення цільової функції відповідає вибір вектору напрямку:

$$p_k = -g(w_k). \qquad (2.31)$$

Саме виразом (2.13) визначається вектор напрямку p в методі найшвидшого спуску.

Обмеження складовою першого порядку при розкладанні функції в ряд Тейлора не дозволяє використовувати інформацію про її кривизни. Це обумовлює повільну збіжність методу (вона залишається лінійної). Зазначений недолік, а також різке уповільнення мінімізації в найближчій околиці точки оптимального рішення, коли градієнт приймає дуже малі значення, роблять алгоритм найшвидшого спуску неефективним. Проте з урахуванням його простоти, невисоких вимог до обсягу пам'яті і щодо невеликої обчислювальної складності саме цей метод протягом багатьох років був і залишається в даний час основним способом навчання багатошарових мереж. Підвищити його ефективність вдається шляхом

модифікації (як правило, евристичної) виразу, що визначає напрямок. Хороші результати приносить застосування методу навчання з так званим моментом. При цьому підході уточнення ваг мережі ($w_{k+1} = w_k + \nabla w_k$) проводиться з урахуванням модифікованої формули визначення значення ваг Δw_k.

$$\Delta w_k = \eta_k p_k + a(w_k - w_{k-1}), \tag{2.32}$$

де a – це коефіцієнт моменту, який приймає значення в інтервалі [0, 1]. Перший доданок цього виразу відповідає звичайному навчання за методом найшвидшого спуску, тоді як друге враховує останнім зміна ваг і не залежить від фактичного значення градієнта. Чим більше значення коефіцієнта a, тим більше значення надає показник моменту на підбір ваг. Це вплив істотно зростає на плоских ділянках цільової функції, а також поблизу локального мінімуму, де значення градієнта близько до нуля.

На плоских ділянках цільової функції збільшення ваг (при постійному значенні коефіцієнта навчання $\eta_k = \eta$) залишається приблизно незмінним. Це означає, що $\Delta w_k = \eta p_k + a\Delta w_k$, тому ефективний приріст значень ваг можна описати відношенням:

$$\Delta w_k = \frac{\eta}{1-a} p_k . \tag{2.33}$$

При значенні a = 0,9 це відповідає 10-кратному збільшенню ефективного значення коефіцієнта навчання і, отже, також 10-кратному прискоренню процесу навчання.

Поблизу локального мінімуму показник моменту, не пов'язаний з градієнтом, може викликати дуже велика зміна ваг, що приводить до збільшення значення цільової функції і до виходу з «зони тяжіння» цього

мінімуму. При малих значеннях градієнта показник моменту починає домінувати у натуральному вираженні (2.14), що призводить до такого збільшенню ваги Δw_k, яке відповідає збільшенню значення цільової функції, що дозволяє вийти із зони локального мінімуму. Однак показник моменту не повинен повністю домінувати протягом усього процесу навчання, оскільки це призвело б до нестабільності алгоритму. Для запобігання такого надлишкового домінування значення цільової функції E контролюється так, щоб допускати його збільшення лише в обмежених межах, наприклад не більше 4%. При такому підході, якщо на чергових кроках ітерації виконується умова $E(k+1)<1.04E(k)$, то зміни ігноруються і вважається, що $(w_k - w_{k-1}) = 0$. При цьому показник градієнта починає домінувати над показником моменту і процес розвивається в напрямку мінімізації, заданому вектором градієнта. Слід підкреслити, що підбір величини коефіцієнта моменту є непростою справою і вимагає проведення великої кількості експериментів, що мають на меті вибрати таке значення, яке найкращим чином відображало б специфіку вирішуваної проблеми.

2.4.3. Алгоритм змінної метрики

У методі змінної метрики використовується квадратичне наближення функції $E(w)$ в околиці отриманого рішення w_k. Якщо у формулі (2.12) обмежитися трьома першими складовими, то отримаємо:

$$E(w_k + p_k) = E(w_k) + g(w_k)^T p_k + \frac{1}{2} p_k^T H(w_k) p_k + 0(h^3). \qquad (2.34)$$

Для досягнення мінімуму функції (2.15) потрібно, щоб $\frac{\partial E(w_k + p_k)}{\partial p_k} = 0$. При виконанні відповідного диференціювання можна отримати умову оптимальності у вигляді:

$$g(w_k) + H(w_k)\,p_k = 0. \qquad (2.35)$$

Елементарне перетворення цього виразу дає очевидне рішення:

$$p_k = -\left[H(w_k)\right]^{-1} g(w_k). \qquad (2.36)$$

Формула (3.16) однозначно вказує напрямок p_k, яке гарантує досягнення мінімального для даного кроку значення цільової функції. З нього випливає, що для визначення цього напрямку необхідно в кожному циклі обчислювати значення градієнта g і гесіан H в точці відомого останнього рішення w_k.

Формула (3.36), що представляє собою основу алгоритму оптимізації, є чисто теоретичним виразом, оскільки її застосування вимагає позитивної визначеності гесіан на кожному кроці, що в загальному випадку практично нездійсненно. З цієї причини в наявних реалізаціях алгоритму, як правило, замість точно певного гесіан $H(w_k)$ використовується його наближення $G(w_k)$. Одним з найбільш популярних вважається метод змінної метрики, що згадується в [15], [16]. Відповідно до цього методу на кожному кроці гесіан або зворотна йому величина, отримана на попередньому кроці, модифікується на величину деякої поправки. Якщо приріст вектору w_k і градієнта g на двох послідовних кроках ітерації позначити відповідно s_k і r_k тобто $s_k = w_k - w_{k-1}$ та $r_k = g(w_k) - g(w_{k-1})$, а матрицю, зворотну наближенню гесіан $V_k = \left[G(w_k)\right]^{-1}, V_{k-1} = \left[G(w_{k-1})\right]^{-1}$, позначити V, то відповідно до дуже ефективною формулою Бройде-Флетчера-Гольдфарба-Шенно процес уточнення значення матриць V можна описати рекуррентной залежністю:

$$V_k = V_{k-1} + \left[1 + \frac{r_k^T V_{k-1} r_k}{s_k^T r_k}\right] \frac{s_k s_k^T}{s_k^T r_k} - \frac{s_k r_k^T V_{k-1} r_k s_k^T}{s_k^T r_k}. \quad (2.37)$$

В іншому відомому алгоритмі Девідона-Флетчера-Пауелла значення гесіан уточняється відповідно до виразу [15]:

$$V_k = V_{k-1} + \frac{s_k s_k^T}{s_k^T r_k} - \frac{V_{k-1} r_k r_k^T V_{k-1}}{r_k^T V_{k-1} r_k}. \quad (2.38)$$

В якості початкового значення зазвичай приймається $V_0 = 1$, а перша ітерація проводиться відповідно до алгоритму найшвидшого спуску. Як показано в [15], [16], при початковому значенні $V_0 = 1$ і при використанні спрямованої мінімізації на кожному кроці оптимізації можна забезпечити позитивну визначеність апроксимованої матриці гесіан. Спрямована мінімізація необхідна при реалізації як стратегії BFGS, так і DFP, причому відповідно до проведених тестами метод BFGS менш чутливий до різних погрішностей обчислювального процесу. З цієї причини, незважаючи на трохи більшу обчислювальну складність, метод BFGS застосовується частіше, ніж DFP.

Метод змінної метрики характеризується більш швидкої збіжністю, ніж метод найшвидшого спуску. Крім того, факт позитивної визначеності гесіан на кожному кроці ітерації надає впевненість в тому, що виконання умови $g(w_k) = 0$ дійсно гарантує рішення проблеми оптимізації. Саме цей метод вважається в даний час одним з найбільш ефективних способів оптимізації функції декількох змінних. Його недолік полягає у відносно великої обчислювальної складності (пов'язаної з необхідністю розрахунку в кожному циклі n^2 елементів гесіан), а також у використанні значних обсягів пам'яті для зберігання елементів гесіан, що в разі оптимізації функції з великою кількістю змінних може стати серйозною проблемою. З цієї

причини метод змінної метрики застосовується для не дуже великих мереж. Зокрема, з використанням персонального комп'ютера була доказана його ефективність для мережі, що містить не більше тисячі зважених зв'язків.

2.4.4. Алгоритм Левенберга -Марквардта

Іншим додатком ньютонівської стратегії оптимізації є алгоритм Левенберга-Марквардта [15]. При його використанні точне значення гесіан $H(w)$ у формулі (2.15) замінюється значенням $G(w)$, яке розраховується на основі міститься в градієнті інформації з урахуванням деякого регулярізаціонного фактора. Для опису цього методу представимо цільову функцію у вигляді, яке відповідає існуванню єдиної навчальної вибірки,

$$E(w) = \frac{1}{2}\sum_{i=1}^{M}\left[e_i(w)\right]^2, \tag{2.39}$$

де $e_i = \left[y_i(w) - d_i\right]$.

При використанні позначень:

$$e(w) = \begin{bmatrix} e_1(w) \\ e_2(w) \\ \dots \\ e_M(w) \end{bmatrix}, J(w) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1}\frac{\partial e_1}{\partial w_2}\dots\frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial w_1}\frac{\partial e_2}{\partial w_2}\dots\frac{\partial e_2}{\partial w_n} \\ \dots \\ \frac{\partial e_M}{\partial w_1}\frac{\partial e_M}{\partial w_2}\dots\frac{\partial e_M}{\partial w_n} \end{bmatrix}, \tag{2.40}$$

вектор градієнта і аппроксимірована матриця гесіан, відповідні цільової функції (2.16), визначаються у вигляді:

$$g(w) = \left[J(w)\right]^T e(w), \tag{2.41}$$

$$G(w) = \left[J(w)\right]^T J(w) + R(w), \tag{2.42}$$

де $R(w)$ позначені компоненти гесіан $H(w)$, що містять вищі 5^{-2162} похідні щодо w. Сутність підходу Левенберга-Марквардта складається в апроксимації $R(w)$ за допомогою регулярізаціонного фактора *v1*, в якому змінна v, звана параметром Левенберга-Марквардта, є скалярною величиною, що змінюється в процесі оптимізації. Таким чином, аппроксимирована матриця гесіан на k-м кроці алгоритму набуває вигляду:

$$G(w_k) = \left[J(w_k)\right]^T J(w_k) + v_k 1. \tag{2.43}$$

На початку процесу навчання, коли фактичне значення w_k ще далеко від шуканого рішення (велике значення вектору похибки e), використовується значення параметра v_k, набагато перевищує власне значення матриці $\left[J(w_k)\right]^T J(w_k)$. В такому випадку гесіан фактично підміняється регуляризаційним фактором:

$$G(w_k) = v_k 1, \tag{2.44}$$

а напрямок мінімізації вибирається за методом найшвидшого спуску:

$$p_k = -\frac{g(w_k)}{v_k}. \tag{2.45}$$

По міру зменшення похибки і наближення до шуканого рішення величина параметра v_k знижується і перший доданок у формулі (2.18) починає грати все більш важливу роль.

На ефективність алгоритму впливає грамотний підбір величини v_k. Занадто велике початкове значення v_k по мірі прогресу оптимізації повинно зменшуватися аж до нуля при досягненні фактичного рішення, близького до шуканого. Відомі різні способи підбору цього значення, але ми обмежимося описом тільки однієї оригінальної методики Д. Марквардта [17]. Нехай значення цільової функції на *k*-м і *(k-1)*-м кроках ітерації позначаються відповідно E_k і E_{k-1}, а значення параметра *v* на цих же кроках - v_k та v_{k-1}. Коефіцієнт зменшення значення *v* позначимо *r*, причому *r>1*. Відповідно до класичного алгоритму Левенберга-Марквардта значення *v* змінюється за наступною схемою:

– якщо $E(\frac{v_{k-1}}{r}) \le E_k$, то прийняти $v_k = \frac{v_k - 1}{r}$;

– якщо $E(\frac{v_{k-1}}{r}) > E_k$ та $E(v_{k-1}) < E_k$, то прийняти $v_k = v_{k-1}$;

– якщо $E(\frac{v_{k-1}}{r}) > E_k$ та $E(v_{k-1}) > E_k$, то збільшити послідовно *m* раз значення *v* до досягнення $E(v_{k-1}r^m) \le E_k$, одночасно приймаючи $v_k = v_{k-1}r^m$.

Така процедура зміни значення *v* виконується до моменту, в якому так званий коефіцієнт вірності відображення *q*, що розраховується по формулою:

$$q = \frac{E_k - E_{k-1}}{\left[\nabla w_k\right]^T g_k + 0.5\left[\Delta w_k\right]^T G_k \Delta w_k}, \qquad (2.46)$$

досягне значення, близького до одиниці.

При цьому квадратична апроксимація цільової функції має високу ступінь збігу з істинними значеннями, що свідчить про близькість

оптимального рішення. В такій ситуації регуляризаційний фактор $v_k 1$ у формулі (2.17) може бути опущений $(v_k = 0)$, процес визначення гесіан зводиться до безпосередньої апроксимації першого порядку, а алгоритм Левенберга-Марквардта перетворюється в алгоритм Гаусса-Ньютона, який характеризується квадратичною збіжністю до оптимального рішення.

2.4.5. Алгоритм сполучених градієнтів

У цьому методі при виборі напрямку мінімізації не використовується інформація про гесіане. Напрямок пошуку p_k вибирається таким чином, щоб воно було ортогональним і зв'язаних до всіх попередніх напрямками $p_0, p_1..., p_{k-1}$. Безліч векторів $p_i, i = 0,1,...,k$ буде взаємно зв'язаних щодо матриці *G*, якщо:

$$p_i^T G p_j = 0, i \neq j . \tag{2.47}$$

Як показано в [9], [15], вектор p_k задовольняє заданим вище умовами, має вигляд:

$$p_k = -g_k + \beta_{k-1} p_{k-1} , \tag{2.48}$$

де $g_k = g(w_k)$ позначає фактичне значення вектору градієнта. З (2.19) випливає, що новий напрямок мінімізації залежить тільки від значення градієнта в точці рішення і w_k від попереднього напрямку пошуку p_{k-1}, помноженого на коефіцієнт сполучення β_{k-1}. Цей коефіцієнт відіграє дуже важливу роль, акумулюючи в собі інформацію про попередні напрямках пошуку. Існують різні правила розрахунку його значення. Найбільш відомі серед них [15], [16]:

$$\beta_{k-1} = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}} \ , \tag{2.49}$$

$$\beta_{k-1} = \frac{g_k^T (g_k - g_{k-1})}{-g_{k-1}^T g_{k-1}}, \tag{2.50}$$

З огляду на накопичення похибок округлення в послідовних циклах обчислень практичне застосування методу сполучених градієнтів пов'язано з поступовою втратою властивості ортогональності між векторами напрямків мінімізації. З цієї причини після виконання і ітерацій (значення п розраховується як функція від кількості змінних, що підлягають оптимізації) проводиться рестарт процедури, на першому кроці якої напрямок мінімізації з точки отриманого рішення вибирається по алгоритму найшвидшого спуску. Метод сполучених градієнтів має збіжність, близьку до лінійної, і він менш ефективний, ніж метод змінної метрики, однак помітно швидше методу найшвидшого спуску. він широко застосовується як єдино ефективний алгоритм оптимізації при вельми значній кількості змінних, яке може досягати декількох десятків тисяч. Завдяки невисоким вимогам до пам'яті і щодо низької обчислювальної складності методу сполучених градієнтів дозволяє успішно вирішувати дуже серйозні оптимізаційні задачі.

2.5. Підбір коефіцієнта навчання

Алгоритми, представлені в попередньому підрозділі, дозволяють визначити тільки напрямок, в якому зменшується цільова функція, але не говорять нічого про величину шагу, при якому ця функція може отримати мінімальне значення. Після вибору правильного напряму p_k слід визначити на ньому нову точку рішення w_{k+1}, в якій буде виконуватися умова

$E(w_{k+1}) < E(w_k)$. Необхідно підібрати таке значення η_k, щоб нове рішення $w_{k+1} = w_k + \eta_k p_k$ лежало якомога ближче до мінімуму функції $E(w)$ в напрямку p_k. Грамотний підбір коефіцієнта η_k має великий вплив на збіжність алгоритму оптимізації до мінімуму цільової функції. Чим сильніше величина η_k відрізняється від значення, при якому $E(w)$ досягає мінімуму в обраному напрямку p_k тим більша кількість ітерацій потрібно для пошуку оптимального рішення. Занадто мале значення η не дозволяє мінімізувати цільову функцію за один шаг і викликає необхідність повторно рухатися в тому ж напрямку. Занадто великий крок призводить до «перестрибування» через мінімум функції і фактично змушує повертатися до нього.

Існують різні способи підбору значення η, званого в теорії нейронних мереж коефіцієнтом навчання. Найпростіший з них (відносно рідко застосовуються в даний час, головним чином для навчання в режимі «Онлайн») заснований на фіксації постійного значення η на весь період оптимізації. Цей спосіб практично використовується тільки спільно з методом найшвидшого спуску. Він має низьку ефективність, оскільки значення коефіцієнта навчання ніяк не залежить від вектору фактичного градієнта і, отже, від напрямку p на даній ітерації. Величина η підбирається, як правило, окремо для кожного шару мережі з використанням різних емпіричних залежностей. Один з підходів полягає у визначенні мінімального значення коефіцієнта η для кожного шару за формулою [18]:

$$\eta \le \min\left(\frac{1}{n_i}\right), \tag{2.51}$$

де n_i позначає кількість входів i -го нейрона в шарі.

Інший більш ефективний метод заснований на адаптивному підборі коефіцієнта η з урахуванням фактичної динаміки величини цільової функції в результаті навчання. Відповідно до цього методу стратегія зміни значення η визначається шляхом порівняння сумарної похибки ε на *i*-й ітерації з її попереднім значенням, причому ε розраховується за формулою:

$$\varepsilon = \sqrt{\sum_{j=1}^{M}(y_j - d_j)^2}\,. \qquad (2.52)$$

Для прискорення процесу навчання слід прагнути до безперервного збільшення η при одночасному контролі приросту похибки ε по порівняно з її значенням на попередньому кроці. Незначне зростання цієї похибки вважається допустимим. Якщо похибка на *(i–1)* і *i*-й ітераціях позначити відповідно ε_{i-1} та ε_i, а коефіцієнти навчання на цих же ітераціях η_{i-1} та η_i то в разі $\varepsilon_i > k_w\varepsilon_{i-1}$ – коефіцієнт допустимого приросту похибки значення η має зменшуватися відповідно до формули:

$$\eta_{i+1} = \eta_i\rho_d\,, \qquad (2.53)$$

де p_d – коефіцієнт зменшення η. В іншому випадку, коли $\varepsilon_i \le k_w\varepsilon_{i-1}$, приймається:

$$\eta_{i+1} = \eta_i\rho_i\,, \qquad (2.54)$$

де p_i – коефіцієнт збільшення η. Незважаючи на деяке зростання обсягу обчислень (необхідних для додаткового розрахунку значень ε), можливо істотне прискорення процесу навчання. Наприклад, реалізація представленої стратегії в програмі *MATLAB* [19] зі значеннями $k_w = 1.41$,

$p_d = 0.7$, $p_i = 1.05$ дозволила в кілька разів прискорити навчання при вирішенні проблеми апроксимації нелінійних функцій.

Цікаво простежити характер зміни коефіцієнта η в процесі навчання. Як правило, на початкових етапах домінує тенденція до його збільшення, однак при досягненні деякого квазістаціонарного стану величина η поступово зменшується, але не монотонно, а циклічно зростаючи і знижуючи в наступних один за одним циклах.

Однак необхідно підкреслити, що адаптивний метод підбору η сильно залежить від виду цільової функції і значень коефіцієнтів k_w, p_d, p_i. Значення, оптимальні для функції одного виду, можуть уповільнювати процес навчання при використанні іншої функції. Тому при практичній реалізації цього методу слід звертати увагу на механізми контролю і управління значеннями коефіцієнтів, підбираючи їх відповідно до специфікою розв'язуваної задачі.

Найбільш ефективний, хоча і найбільш складний, метод підбору коефіцієнта навчання пов'язаний з спрямованої мінімізацією цільової функції в обраному заздалегідь напрямку p_k. Необхідно так підібрати скалярний значення η_k, щоб нове рішення $w_{k+1} = w_k + \eta_k p_k$ відповідало мінімуму цільової функції в даному напрямку p_k. Насправді отримується рішення w_{k+1} тільки з певним наближенням можна вважати справжнім мінімумом. Це результат компромісу між обсягом обчислень і впливом величини η_k на збіжність алгоритму.

Серед найбільш популярних способів спрямованої мінімізації можна виділити без градієнтні і градієнтні методи. У без градієнтних методах використовується тільки інформація про значеннях цільової функції, а її мінімум досягається в процесі послідовного зменшення діапазону значень вектору *w*. Прикладами можуть служити методи розподілу навпіл, золотого перетину або метод Фібоначчі [15], [16], що розрізняються способом декомпозиції одержуваних під діапазонів.

Заслуговує на увагу метод апроксимації цільової функції *E(w)* в попередньо вибраному напрямку p_k з подальшим розрахунком мінімуму, одержуваного таким чином, функції однієї змінної η. Оберемо для апроксимації многочлен другого порядку виду:

$$E(w) \to P_2(\eta) = a_2\eta^2 + a_1\eta + a_0, \tag{2.55}$$

де a_2, a_1, a_0 позначені коефіцієнти, які визначаються в кожному циклі оптимізації. Якщо для розрахунку входять до P_2 коефіцієнтів використовуються три довільні точки w_1, w_2, w_3, що лежать в напрямку p_k тобто $w_1 = w + \eta_1 p_k$, $w_2 = w + \eta_2 p_k$, $w_3 = w + \eta_3 p_k$ (в цьому виразі в позначено попереднє рішення), а відповідні цим точкам значення цільової функції *E(w)* позначені $E_1 = E(w_1), E_2 = E(w_2), E_3 = (w_3)$, то:

$$P_2(\eta_1) = E_1, P_2(\eta_2) = E_2, P_2(\eta_3) = E_3. \tag{2.56}$$

Коефіцієнти a_2, a_1, a_0 многочлена P_2 розраховуються відповідно з системою лінійних рівнянь, що описуються в (2.21). Для мінімуму цього многочлена його похідна $\frac{\partial P_2}{\partial \eta} = 2a_2\eta + a_1$ прирівнюється до нулю, що дозволяє отримати значення η у вигляді $\eta_{\min} = \frac{-a_1}{2a_2}$. Після підстановки виразів для E_1, E_2, E_3 в формулу розрахунку $\eta_{\min}$ отримуємо:

$$\eta_{\min} = \eta_2 - \frac{1}{2}\frac{(\eta_2 - \eta_1)^2(E_2 - E_3) - (\eta_2 - \eta_3)^2(E_2 - E_1)}{(\eta_2 - \eta_1)(E_2 - E_3) - (\eta_2 - \eta_3)(E_2 - E_1)}. \tag{2.57}$$

Однак кращим рішенням вважається застосування градієнтних методів, в яких, крім значення функції, враховується також і її похідна вздовж направляючого вектору p_k. Вони дозволяють значно прискорити досягнення мінімуму, оскільки використовують інформацію про напрямок зменшення величини цільової функції. У такій ситуації застосовується, як правило, апроксимуючий многочлен третього порядку:

$$P_2(\eta) = a_3\eta^3 + a_2 n^2 + a_1\eta + a_0 . \qquad (2.58)$$

Значення чотирьох коефіцієнтів a_i цього многочлена можна отримати виходячи з інформації про величину функції і її похідної всього лише в двох точках. Якщо прирівняти до нуля похідну многочлена щодо η, то можна отримати формулу для розрахунку $\eta_{\min}$ у вигляді:

$$\eta_{\min} = \frac{-a_2 + \sqrt{a_2^2 - 3a_2 a_1}}{3a_3} . \qquad (2.59)$$

Для вибору напрямку мінімізації в ньому застосовується метод змінної метрики або класичний метод сполучених градієнтів, а для розрахунку оптимального значення коефіцієнта навчання – апроксимація многочленом третього порядку. Програма дозволяє працювати з будь-якою кількістю шарів, причому функція активації нейронів кожного шару задається індивідуально. На вибір пропонуються функції: сигмоїдальна уніполярна (Sigm), сигмоїдальна біполярна (Bip), а також лінійна (Lin). Градієнт розраховується з застосуванням методу потокових графів.

2.6. Методи ініціалізації ваг

Навчання нейронних мереж, навіть при використанні найефективніших алгоритмів, являє собою трудомісткий процес, далеко не завжди дає очікувані результати. Проблеми виникають через нелінійних функцій активації, що утворюють численні локальні мінімуми, кохалась може зводитися процес навчання. Звичайно, застосування продуманої стратегії поведінки (наприклад, імітації відпалу, методу мультістара, генетичних алгоритмів) зменшує ймовірність зупинки процесу в точці локального мінімуму, однак платою за це стає різке збільшення трудомісткості і тривалості навчання. Крім того, для застосування названих методів необхідний великий досвід в області вирішення складних проблем глобальної оптимізації, особливо для правильного підбору керуючих параметрів. На результати навчання величезний вплив справляє підбір початкових значень ваг мережі. Ідеальними вважаються початкові значення, досить близькі до оптимальних. При цьому вдається не тільки усунути затримки в точках локальних мінімумів, а й значно прискорити процес навчання. Нажаль, не існує універсального методу підбору ваг, який би гарантував знаходження найкращої початкової точки для будь-якої розв'язуваної завдання. З цієї причини в більшості практичних реалізацій найчастіше застосовується випадковий підбір ваг з рівномірним розподілом значень в заданому інтервалі.

Неправильний вибір діапазону випадкових значень Терезів може викликати занадто раннє насичення нейронів, в результаті якого, незважаючи на триваюче навчання, середньоквадратична похибка буде залишатися практично постійною.

Явище цього типу не означає попадання в точку локального мінімуму, а свідчить про досягнення сідловій зони цільової функції внаслідок занадто великих початкових значень ваг. При визначених навчальних сигналах в

вузлах підсумкових нейронів генеруються сигнали $u_i = \sum_j w_{ij} x_j$ зі значеннями, відповідними глибокого насичення сигмоїдальної функції активації. При цьому поляризація насичення обернена очікуваної. Значення поворотного сигналу, генерується в методі зворотного поширення, пропорційне величині похідною від функції активації $\frac{\partial f}{\partial x}$, в точці насичення близько нуля. Тому зміни значень ваг, які виведуть нейрон зі стану насичення, відбуваються дуже повільно. Процес навчання надовго застряє в сідловій зоні. Процес навчання надовго застряє в сідловій зоні.

Слід звернути увагу, що в стані насичення може перебувати одна частина нейронів, тоді як інша частина залишається в лінійному діапазоні, з для них зворотний навчальний сигнал приймає нормальний вигляд. Це означає, що пов'язані з такими нейронами ваги уточняються нормальним чином, і процес їх навчання веде до швидкого зменшення похибки.

Як наслідок, нейрон, що залишається в стані насичення, не бере участі в показі даних, скорочуючи таким чином ефективна кількість нейронів в мережі. В результаті процес навчання надзвичайно сповільнюється, тому стан насичення окремих нейронів може тривати практично безперервно аж до вичерпання ліміту ітерацій.

Випадкова ініціалізація, що вважається єдиним універсальним способом приписування початкових значень ваг мережі, повинна забезпечити таку стартову точку активації нейронів, яка лежала б досить далеко від зони насичення. Це досягається шляхом обмеження діапазону допустимих значень. Оцінки нижньої і верхньої меж такого діапазону, пропоновані різними дослідниками на підставі численних комп'ютерних експериментів, відрізняються в деталях, проте практично всі лежать в межах нулю та одиниці.

В роботі [20] запропоновано рівномірний розподіл ваг, нормалізоване для кожного нейрона по амплітуді $\frac{2}{\sqrt{n_{in}}}$, де n_{in} означає кількість входів нейрона. Значення ваг поляризації для нейронів прихованих шарів повинні приймати випадкові значення з інтервалу $\left[-\frac{\sqrt{n_{in}}}{2}, \frac{\sqrt{n_{in}}}{2}\right]$, а для вихідних нейронів – нульові значення.

Д. Нгуен і Б. Уідроу в своїх міркуваннях на тему оптимальних значень початкових ваг використовують кусково-лінійну апроксимацію сигмоїдальної функції активації. На цій основі вони визначили оптимальну довжину випадкового вектору ваг нейронів прихованих шарів рівної $\sqrt[n_{in}]{N_h}$, де N_h означає кількість нейронів в прихованому шарі, а n_{in} – кількість входів даного нейрона.

Оптимальний діапазон ваг поляризації для нейронів прихованого шару визначений в межах $\left[-\sqrt[n_{in}]{N_h}, \sqrt[n_{in}]{N_h}\right]$. Початкові ваги вихідних нейронів, по думку згаданих авторів, не повинні залежати від топології області допустимих значень і можуть вибиратися випадковим чином з інтервалу [– 0,5, 0,5].

Рішення представлених проблем випадкової ініціалізації ваг мережі спирається або на інтуїцію дослідника, або на результати великої кількості чисельних експериментів. Більш детальний аналіз подій, що відбуваються в процесі навчання, дозволить точніше виявити причини уповільнення навчання персептрона мережі, затримок в сідлових зонах, а також занадто раннього завершення навчання в точках локальних мінімумів, далеких від оптимального рішення.

Результатом такого аналізу повинні стати заходи попередження цих небажаних явищ за рахунок застосування відповідних процедур попередньої обробки навчальних даних для необхідної ініціації як структури мережі, так і значень ваг.

Ці процедури базуються або на аналізі даних з використанням конкуренції [21], подібно до того, як це відбувається в мережах, що самоорганізуються на основі конкуренції, або на використанні інформації про кореляційні залежності навчальних даних [22].

3 АДАПТИВНА ПОЛІНОМІАЛЬНА АКТИВАЦІЙНА ФУНКЦІЯ В ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖАХ

3.1 Архітектура нейрону з адаптивною активаційною функцією

Елементарний персептрон Розенблатта як вузол будь-якої нейронної мережі реалізує нелінійне відображення виду:

$$\begin{aligned}\hat{y}_j(k) &= \psi_j\left(\theta_{j0} + \sum_{i=1}^{n} w_{ji} x_i(k)\right) = \psi_j\left(\sum_{i=1}^{n} w_{ji} x_i(k)\right) = \\ &= \psi_j\left(w_j^T x(k)\right) = \psi_j\left(u_j(k)\right),\end{aligned} \tag{3.1}$$

де $\hat{y}_j(k)$ – вихідний сигнал j-го нейрону мережі у момент часу k=1,2,…;

$x(k) = \left(1,\ x_1(k),\ldots,x_i(k),\ldots,x_n(k)\right)^T \in R^{(n+1)}$ – вхідний векторний сигнал;

$\theta_{j0} \equiv w_{j0}$ – сигнал зміщення;

$w_j - \left(w_{j0}, w_{j1},\ldots,w_{ji},\ldots,w_{jn}\right)^T \in R^{(n+1)}$ – вектор синаптичних вагів, що мають бути налаштовані в процесі навчання;

$u_j(k)$ – сигнал внутрішньої активації;

$\psi_j(\bullet)$ – активаційна функція j-го нейрону, що вибирається, як правило, з емпіричних міркувань в процесі навчання та функціонування нейронної мережі.

Таким чином, у теоремі Цибенко використовується σ-функція:

$$\hat{y}_j(k) = \psi_j(u_j(k)) = \frac{1}{1+\exp\left(-\gamma_j(u_j(k))\right)}, \tag{3.2}$$

де γ_j – це так званий параметр посилення (gain parameter), що визначає форму цієї функції.

Зауважимо, що похідна сигмоїдальної функції має вигляд:

$$\psi_j'\left(u_j(k)\right)=\gamma_j\hat{y}_j(k)\left(1-\hat{y}_j(k)\right), \tag{3.3}$$

тобто, вона є дзвонкуватою функцією, при цьому чим ближче значення $\hat{y}_j(k)$ до 0 чи 1, тим ближче значення цієї похідної до 0, що породжує явище «затухаючого градієнту». Сімейство функцій ReLU у загальному вигляді може бути записано у формі

$$\psi_j(u_j(k))=\begin{cases}u_j(k)\text{, якщо } u_j>0,\\ a_j u_j(k) \text{ в іншому випадку},\end{cases} \tag{3.4}$$

де параметр a_j обирається з тих чи інших емпіричних міркувань та залишається незмінним в процесі навчання, при цьому в стандартному вигляді ReLU параметр $a_j=0$, тобто

$$\psi_j(u_j(k))=0, \text{ якщо } u_j(k)<0. \tag{3.5}$$

Це призводить до того, що при відмінних значеннях сигнала внутрішньої активації процес навчання зупиняється.

Узагальненням активаційної функції (3.1) є вираз:

$$\psi_j(u_j(k))=\begin{cases}a_j^R u_j(k)\text{, якщо } u_j>0,\\ a_j^L u_j(k) \text{ в іншому випадку}.\end{cases} \tag{3.6}$$

Однак при цьому виникає питання як вибрати параметри a_j^R та a_j^L.

Відповіддю на це запитання є введення в процес навчання нейрона додаткової процедури налаштування a_j^R та a_j^L. Авжеж, це ускладнює процес навчання та призводить до того, що замість $n+1$ налаштованих параметрів, що входять до вектору w_j, треба уточнювати $n+3$ параметра. При цьому, однак, забезпечуються покращені апроксимаційні властивості, так як (3.1) може приймати різні форми, наприклад,

$$\psi_j(u_j(k)) = \left|u_j(k)\right|. \tag{3.7}$$

На рисунку 3.1 зображена схема нейрона з адаптивною активаційною функцією AdPReLU, де в процесі навчання налаштовуються $n+3$ параметра: w_j, a_j^R та a_j^L.

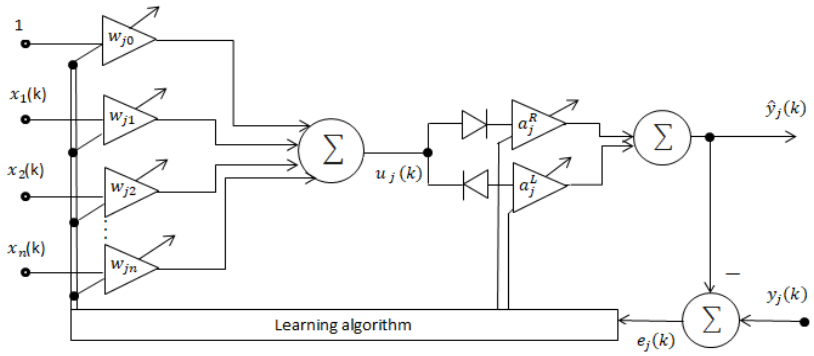

Рисунок 3.1 – Схема нейрона з адаптивною активаційною функцією AdPReLU

де $y_j(k)$ – зовнішній навчальний сигнал;

$e_j(k) = y_j(k) - \hat{y}_j(k) = y_j(k) - \psi_j\left(u_j(k)\right)$ – помилка навчання.

Але, вважаємо, що нам недостатньо того, що розроблений нейрон з активаційною функцією AdPReLU, що має n+3 параметрами для налаштування, показав покращені апроксимаційні властивості та

підвищення швидкості процесу навчання. Тому було вирішено розробити адаптивну поліноміальну активаційну функція (adaptive polynomial activation function) APAF, що буде мати n+7 параметрів для налаштування.

Узагальненням активаційної функції (3.1) є вираз:

$$\psi_j(u_j(k)) = \begin{cases} a_{j_1}^R u_j(k) + a_{j_2}^R u_j^2(k) + a_{j_3}^R u_j^3(k), \textit{ якщо } u_j > 0, \\ a_{j_1}^L u_j(k) + a_{j_2}^L u_j^2(k) + a_{j_3}^L u_j^3(k) \textit{ в іншому випадку.} \end{cases} \quad (3.8)$$

При цьому якщо $a_{j_2}^R = a_{j_3}^R = a_{j_2}^L = a_{j_3}^L = 0$, то цей вираз перетвориться на узагальнення функції активації AdPReLU.

Тоді як $a_{j_1}^L = 0$, $a_{j_1}^R = 1, a_{j_1}^R = a_{j_1}^L = 1{,}5$, $a_{j_2}^R = a_{j_2}^L = 0$, $a_{j_3}^R = a_{j_3}^L = -0{,}5$, розглядається як squashing function (рисунок 3.2):

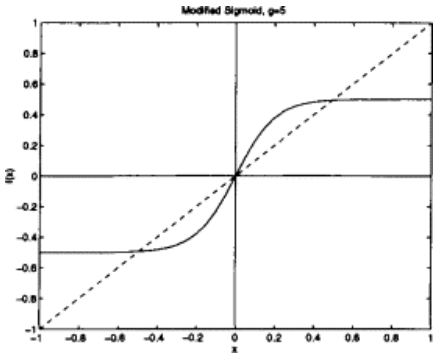

Рисунок 3.2 – Squashing function

На рисунку 3.3 зображена схема нейрона з адаптивною активаційною функцією APAF, де в процесі навчання налаштовуються $n+7$ параметра.

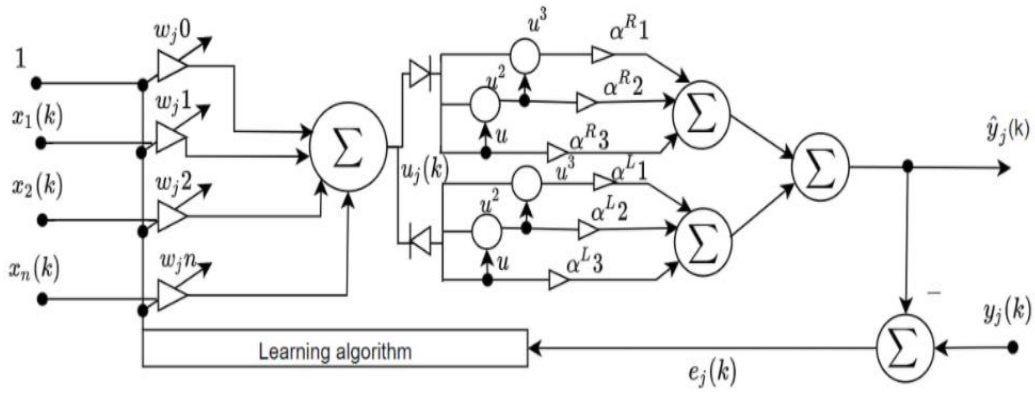

Рисунок 3.3 – Схема нейрона з адаптивною активаційною функцією APAF

де $y_j(k)$ – зовнішній навчальний сигнал;

$e_j(k) = y_j(k) - \hat{y}_j(k) = y_j(k) - \psi_j\left(u_j(k)\right)$ – помилка навчання.

3.2 Процедура навчання нейрону з адаптивною активаційною функцією

Як було вказано вище, необхідно налаштувати $n+1+6=n+7$ параметрів: w_j, $a_{j_1}^R$, $a_{j_2}^R$, , $a_{j_3}^R$, та $a_{j_1}^L$, $a_{j_2}^L$, $a_{j_3}^L$. Далі індекси R та L тимчасово не використовуємо.

Розглянемо вектори:

$$\tilde{a}_j = \left(a_{j_1}, a_{j_2}, a_{j_3}\right)^T, \tag{3.9}$$

$$\tilde{u}_j = \left(u_j, u_j^2, u_j^3\right)^T. \tag{3.10}$$

Тоді, використовуючи ці вирази:

$$\psi_j\left(u_j(k)\right)=\tilde{a}_j^T\tilde{u}_j(k). \tag{3.11}$$

Похідна виразу:

$$\psi_j'\left(u_j(k)\right)=a_{j_1}+2a_{j_2}u_j(k)+3a_{j_3}u_j^2(k). \tag{3.12}$$

В якості критерія навчання використовується стандартна квадратична функція:

$$\begin{aligned}E_j(k)&=\frac{1}{2}e_j^2(k)=\frac{1}{2}\left(y_i(k)-\psi_j\left(u_j(k)\right)\right)^2=\frac{1}{2}\left(y_i(k)-\tilde{a}_j^T\tilde{u}_j(k)\right)^2=\\&=\frac{1}{2}\left(y_j(k)-a_{j_1}u_j(k)-a_{j_2}u_j^2(k)-a_{j_3}u_j^3(k)\right)^2,\end{aligned} \tag{3.13}$$

де $u_j(k)=w_j^Tx(k)$ мінімізація якої за допомогою градієнтної процедури призводить до алгоритму налаштування синаптичних вагів виду

$$\begin{aligned}w_{ji}(k)&=w_{ji}(k-1)-\eta(k)\frac{\partial E_j(k)}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial w_{ji}}=\\&=w_{ji}(k)-\eta(k)e_j(k)\frac{\partial e_j(k)}{\partial w_{ji}}=\\&=w_{ji}(k-1)-\eta(k)e_j(k)\frac{\partial e_j(k)}{\partial u_j(k)}\frac{\partial u_j(k)}{\partial w_{ji}}=\\&=w_{ji}(k-1)+\eta(k)e_j(k)\psi_j'(u_j(k))x_i(k)=\\&=w_{ji}(k-1)+\eta(k)\delta_j(k)x_i(k),\end{aligned} \tag{3.14}$$

$$\begin{aligned}w_j(k)&=w_j(k-1)+\eta(k)e_j(k)\psi'\left(u_j(k)\right)x(k)=\\&=w_j(k-1)+\eta(k)\delta_j(k)x(k),\end{aligned} \tag{3.15}$$

чи у векторній формі:

$$w_j(k) = w_j(k-1) + \eta(k)\delta_j(k)x_i(k), \tag{3.16}$$

де $\eta(k)$ – це параметр навчання;

$\delta_j(k) = e_j(k)\psi'_j\left(u_j(k)\right)$ – це δ-помилка.

Для стандартної функції гіперболічного тангенсу можна записати:

$$\begin{aligned}&\frac{\partial\psi(u_j)}{\partial u_j} = \gamma_j\left(1-\left(\tanh\gamma_j u_j\right)^2\right) = \gamma_j\left(\operatorname{sech}\gamma_j u_j\right) = \gamma_j(1-\hat{y}_j{}^2)w_j(k) = \\ &= w_j(k-1) + \eta(k)e_j(k)\gamma_j\left(1-\hat{y}_j{}^2(k)\right)x(k),\end{aligned} \tag{3.17}$$

звідки видно, що при $\hat{y}_j(k) \to \pm 1$ виникає явище «затухаючого градієнту».

Процес навчання нейрона, що був наведений на рисунку 3.1, згідно з методом зворотного поширення, починається з налаштування $\tilde{a}_j$, при цьому для спрощення перетворень індекси R та L тимчасово опускаємо. Тоді

$$\tilde{a}_j(k) = \tilde{a}_j(k-1) - \eta_a(k)\nabla_{\tilde{a}_j}E_j(k) = a_j(k-1) + \eta_a(k)e_j(k)\tilde{u}_j(k), \tag{3.18}$$

де $e_j(k) = y_j(k) - \tilde{a}_j^T(k-1)\tilde{u}_j(k)$.

Оскільки $e_j(k)$ лінійно залежить від $\tilde{a}_j$, то можна скористуватися алгоритмом навчання Качмажа-Уідроу-Хоффа:

$$\tilde{a}_j(k) = \tilde{a}_j(k-1) + \frac{y_j(k) - \tilde{a}_j^T(k-1)\tilde{u}_j(k)}{\left\|\tilde{u}_j(k)\right\|^2}\tilde{u}_j(k), \tag{3.19}$$

а також його регуляризовану версію:

$$\tilde{a}_j(k) = \tilde{a}_j(k-1) + \frac{y_j(k) - \tilde{a}_j^T(k-1)\tilde{u}_j(k)}{\gamma_a + \left\|\tilde{u}_j(k)\right\|^2}\tilde{u}_j(k), \tag{3.20}$$

чи скористуватися операцією псевдообернення:

$$\tilde{a}_j(k) = \tilde{a}_j(k-1) + \left(y_j(k) - \tilde{a}_j^T(k-1)\tilde{u}_j(k)\right)\tilde{u}_j^T(k), \tag{3.21}$$

Після такту налаштування параметрів $\tilde{a}_j$ можна перейти до такту навчання синаптичних ваг $w_j(k)$, в цьому випадку критерій навчання базується на помилці $\hat{e}_j(k)$.

Вводимо критерій навчання з урахуванням того, що $\tilde{a}_j$ вже попередньо налаштовані:

$$\tilde{E}_j(k) = \frac{1}{2}\tilde{e}^2(k) = \frac{1}{2}(y_j(k) - \tilde{\alpha}^T(k)\tilde{u}_j(k))^2, \tag{3.22}$$

де $\tilde{u}_j(k) = \left(w_j^T(k-1)x(k), \left(w_j^T(k-1)x(k)\right)^2, \left(w_j^T(k-1)x(k)\right)^3\right)^T$,

$$\begin{aligned} w_j(k) &= w_j(k-1) + \eta_w(k)\tilde{e}_j(k)(a_{j1}(k) + \\ &+ 2a_{j2}(k)u_j(k) + 3a_{j3}(k)u_j^2(k))x(k), \end{aligned} \tag{3.23}$$

$$\begin{aligned} w_j(k) &= w_j(k-1) + \eta_w(k)\tilde{e}_j(k)(a_{j1}(k) + \\ &+ 2a_{j2}(k)u_j(k) + 3a_{j3}(k)u_j^2(k))x(k), \end{aligned} \tag{3.24}$$

або

$$w_j(k) = w_j(k-1) + \eta_w(k)\tilde{e}_j(k)J_j(k). \tag{3.25}$$

Можна скористуватися однокроковою версією алгоритму Левенберга-Марквардта та записати формули у наступному вигляді:

$$w_j(k) = w_j(k-1) + \left(J_j(k)J_j^T(k) + \gamma_w I\right)^{-1}\tilde{e}_j(k)J_j(k), \tag{3.26}$$

де γ_w – параметр регуляризації синаптичних вагів.

Далі використовуємо лему обернення матриць:

$$\begin{aligned}
&\left(J_j(k)J_j^T(k) + \gamma_w I\right)^{-1} = \gamma_w^{-1}I - \frac{\gamma_w^{-1}IJ_j(k)J_j^T(k)\gamma_w^{-1}I}{1 + J_j^T(k)\gamma_w^{-1}IJ_j(k)} = \\
&= \gamma_w^{-1}I - \frac{\gamma_w^{-2}IJ_j(k)J_j^T(k)}{1 + \gamma_w^{-1}IJ_j(k)J_j^T(k)} = \\
&= \frac{\gamma_w^{-1}I + \gamma_w^{-2}IJ_j^T(k)J_j(k) - \gamma_w^2 J_j^T(k)J_j(k)}{1 + \gamma_w^{-1}J_j^T(k)J_j(k)}.
\end{aligned} \tag{3.27}$$

Алгоритм навчання синаптичних ваг можна переписати таким чином:

$$\begin{aligned}
&w_j(k) = w_j(k-1) + \\
&+ \frac{\gamma_w^{-1}I + \gamma_w^{-2}IJ_j^T(k)J_j(k) - \gamma_w^2 J_j^T(k)J_j(k)}{1 + \gamma_w^{-1}J_j^T(k)J_j(k)}J_j(k)\tilde{e}_j(k) = \\
&= w_j(k-1) + \frac{\gamma_w^{-1}J_j(k)\tilde{e}_j(k)}{1 + \gamma_w^{-1}J_j^T(k)J_j(k)} = w_j(k-1) + \frac{\tilde{e}_j(k)J_j(k)}{\gamma_w + \left\|J_j(k)\right\|^2}.
\end{aligned} \tag{3.28}$$

4 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

Для демонстрації ефективності запропонованого нейрону та його процедури навчання, було реалізовано моделювання тесту, що базується на апроксимації вихідного сигналу $y_j(k)$, що визначається виразом:

$$y_j(k) = \tanh\left(0{,}1x_1(k) + 0{,}2x_2(k) + 0{,}3x_3(k) + 0{,}4x_4(k)\right) = \\ = \tanh\left(u_j(k)\right), \tag{4.1}$$

де $x_i(k)$ – рівномірно розподілена величина в інтервалі $-1 \le x_i(k) \le 1$.

Результати запропонованого підходу порівнювалися з результатами отриманими за допомогою нейрона-адаліни, нейрона з активаційною функцією ReLU та нейрона з класичною активаційною функцією $\tanh u_j(k)$.

В цьому експерименті найкращі результати показав нейрон з активаційною функцією (AdPReLU), тобто нейрон перевершує Адаліну, нейрон з ReLU та нейрон з tanhu_j (k).

На рисунку 4.1–4.3 показано як змінюється середня квадратична помилка оцінювання.

$$\overline{e_j}^2(N) = \frac{1}{N}\sum_{k=1}^{n} e_j{}^2(N-1) + \frac{1}{N}\left(e_j{}^2(N) - \overline{e_j}^2(N-1)\right). \tag{4.2}$$

В якості вихідного сигналу були вибрані вирази:

$$y_j(k) = \sin\left(0{,}5\pi u_j(k)\right), \tag{4.3}$$

$$y_j(k) = \tanh u_j(k). \tag{4.4}$$

$$y_j(k)=\begin{cases}\tanh u_j(k), якщо\, u_j(k)>0,\\ u_j{}^3(k) в\, іншому\, випадку.\end{cases} \quad (4.5)$$

Приведемо результати першого експерименту на рисунку 4.1. Порівнюючи APAF з класичними активаційними функціями цікаво також подивитися чи зможе вона покращити результати отримані з використанням функції AdPreLU.

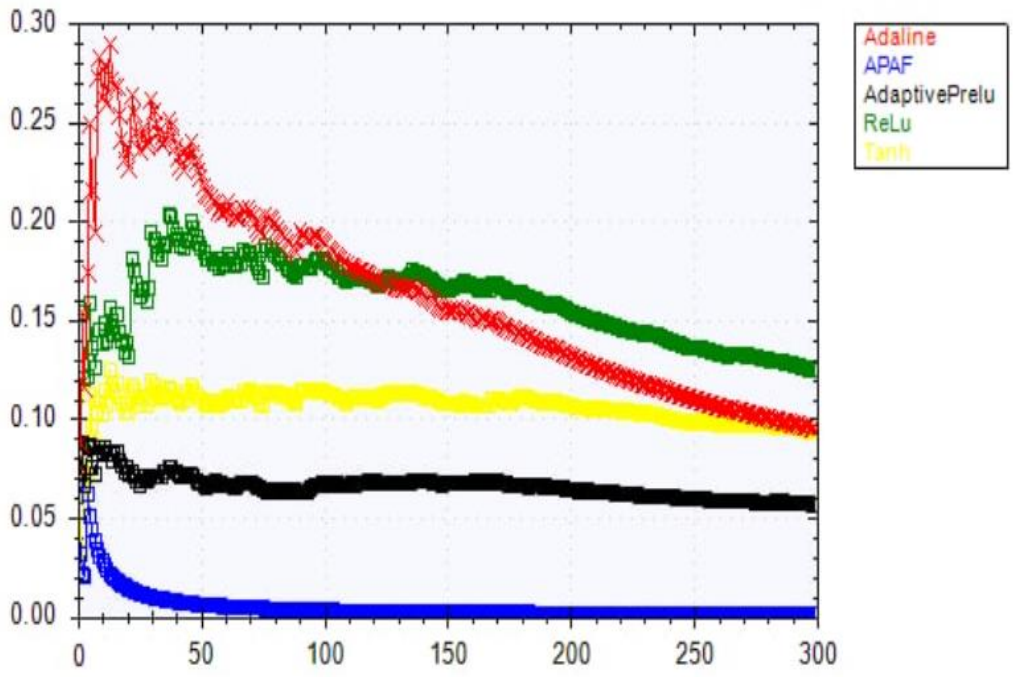

Рисунок 4.1 – Змінення середньої квадратичної помилки з вихідним сигналом (4.3)

Продовжимо експеримент з іншими вхідними даними, результати якого можна спостерігати на рисунку 4.2 та на рисунку 4.3.

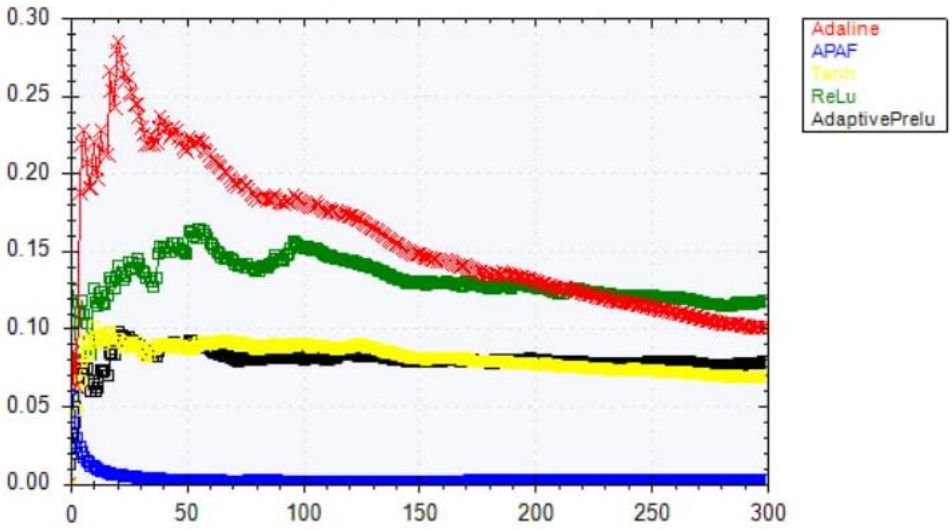

Рисунок 4.2 –Змінення середньої квадратичної помилки з вихідним сигналом (4.4)

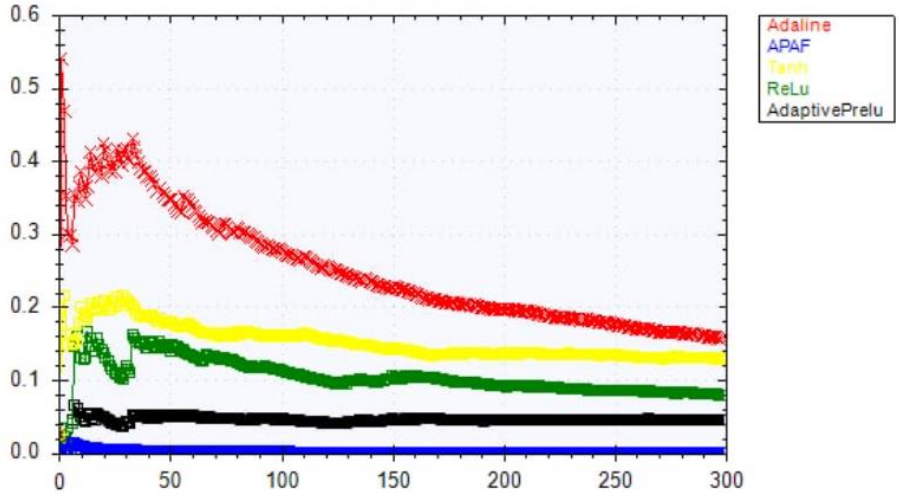

Рисунок 4.3 –Змінення середньої квадратичної помилки з вихідним сигналом (4.5)

Бачимо, що порівнюючи з іншими активаційними функціями, APAF має найвищу швидкість навчання та найкращі апроксимаційні здібності, порівнюючи не тільки з класичними активаційними функціями такими як адаліна, ReLu та гіперболічний тангенс, а також з розробленою AdPReLU.

ВИСНОВКИ

У цій кваліфікаційній роботі був запропонований формальний нейрон глибинної нейронної мережі з адаптивною поліноміальною активаційною функцією, параметри якої налаштовуються одночасно з синаптичними вагами.

Застосування запропонованої активаційної функції дає можливість значно покращити аппроксимуючі властивості.

Окрім того, використання введеної функції APAF в глибинних нейронних мережах захищає процес навчання від ефектів «затухаючого і вибухаючого градієнтів», при цьому використовуються алгоритми налаштування оптимізовані за швидкодією тобто, істотно скорочують час навчання мережі в цілому.

Таким чином, в кваліфікаційній роботі представлені результати, які є відповідно до поставленої мети рішенням актуальної проблем глибинних нейронних мереж, пов’язаної з вибором активаційної функції для вирішення задач в рамках інтелектуального аналізу даних та оптимізації процесу навчання за швидкодією.

Проведені теоретичні та практичні дослідження дозволили зробити наступні висновки:

– в рамках роботи була розроблена архітектура нейрона з адаптивною поліноміальною активаційної функцією (APAF);

– проведено імітаційне моделювання на різних наборах даних, аби дослідити швидкість навчання нейрону з APAF, в порівнянні з нейроном Адаліни, нейроном з PReLU, AdPReLU та нейроном з активаційною функцією гіперболічного тангенса;

– розроблений нейрон з активаційною функцією APAF показав покращені апроксимаційні властивості та підвищення швидкості процесу навчання;

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Karayiannis N. Accelerating training of feedforward neural network using generalized hebbian rules for initializing the internal representation IEEE Proc. ICNN, Orlando, 1994. 1242–1247 p.

2. Barron A. R. Approximation and estimation bounds for artificial neural networks. Machine learning. Boston. Manufactured int the Netherlands: Khuwer Academic Publishers, 1994. 115–133 p.

3. Thintm G., Fiesler E. Neural network initialization. From Natural to Artificial Neural Computation. Eds. J. Mira, F. Sandoval. Malaga: IWANN, 1995. 533–542 p.

4. Widrow B., Stearns S. Adaptive signal processing, New York: Prentice Hall, 1985.

5. Golub G., Van Loan C. Matrix computations, New York: Academic Press, 1991.

6. Haykin S. Neural networks, a comprehensive foundation, New York: Macmillan College Publishing Company, 1994.

7. Cichocki A., Unbehauen R Neural networks for solving systems of linear equations and related problems. IEEE Trans. Warszawa: Tech. Univ., Koszykova, 1992. 124–138 p.

8. Osowski S., Stodolski M, Bojarczak P. Fast second order learning algorithm for feedforward multilayer neural networks and its applications. Neural Networks. Washington, 1995. 1583–1596 p.

9. Rumelhart D. E., Hinton G. E., Williams R. J. Learning internal representations by error propagation parallel distributed processing: Explorations in the Microstructures of Cognition. D. E. Rumelhart, J. L. McClelland, (Eds.). Cambridge: MIT Press, 1986.

10. Hebb D. Organization of behaviour. New York: J. Wiley, 1949.

11. Osowski S. Sieci neuronowe w ujeciu algoiytmicznym. Warszawa: WNT, 1996.

12. Oja E., Ogawa H, Wangviwattana J. Learning in nonlinear constrained hebbian networks. Espoo, Finland, 1991. 385–390 p.

13. Wierzbicki A., Findeisen W., Szymanowski J. Teoria i metody obliczeniowe optymalizacji. Warszawa: WNT, 1977.

14. Widrow B., Hoff M. Adaptive switching circuits. California: Proc. IRE WESCON Convention Record, 1960. 107–115 p.

15. Gill P., Murray W., Wright M. Practical Optimization. New York: Academic Press, 1981.

16. Weymaere N., Martens J.P. On the initialization and optimization of multilayer perception IEEE Trans. Neural Networks Cambridge: 1994. 73–75 p.

17. Minsky M, Papert S. Perceptrons: an introduction to computational geometry. Cambridge: University of Cambridge, 1988.

18. Marquardt D. An algorithm for least squares estimation of nonlinear parameters. Oakland: SIAM, 1963. 431–442 p.

19. Demuth H., Beale M. Neural Network Toolbox for use with Matlab. Natick: The MathWorks, 1992.

20. Rosenblatt F. Principle of neurodynamics, New York: Spartan, 1992.

21. Denoeux J., Lengalle R. Initializing back propagation networks with prototypes. Neural Networks. New York: Pergamon Press, 1993. 351–363 p.

22. Klimauskas G. Neural Ware, User manual. Natick: Neural Ware Inc., 1992.