Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Безпеки інформаційних технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Криптоаналіз простого шифру заміщення-перестановки за допомогою штучної нейронної мережі (тема)

Виконав:
Левченко Р. Р.
(прізвище, ініціали)

студент 2 курсу, групи БІКСм-19-1

Спеціальність 125 - Кібербезпека
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва освітньої програми)

Керівник Руженцев В.І.
(прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Халімов Г.З.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Безпеки інформаційних технологій
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)
Освітня програма Безпека інформаційних та комунікаційних систем
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри БІТ Халімов Г.З.

_____
(підпис)
«_____»_____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Левченко Роману Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Криптоаналіз простого шифру заміщення-перестановки за допомогою штучної нейронної мережі затверджена наказом по університету від "22" жовтня 2020 р. № 1412Ст
2. Термін подання студентом роботи 17.12.2020
3. Вихідні дані до роботи: опис моделі нейронної мережі, аналіз простого шифру заміщення-перестановки
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
1) вибрати та реалізувати прості перетворення шифрування та дешифрування шифру заміщення-перестановки
2) на основі аналізу літератури вибрати архітектуру нейронної мережі
3) вибрати бібліотеку для реалізації нейронної мережі
4) виконати обчислювальні експерименти криптоаналізу
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів): презентація з графічними слайдами.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Руженцев В.І.		

7. Дата видачі завдання 14 вересня 2020 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання на атестаційну роботу	07.09.2020	виконано
2.	Пошук та аналіз літературних джерел за темою дипломної роботи	07.09.2020	виконано
3.	Вибір та реалізація простого перетворення шифрування та дешифрування шифру заміщення-перестановки	21.09.2020	виконано
4.	Аналіз особливостей шифру заміщення-перестановки і використання нейронних мереж для криптоаналізу	01.10.2020	виконано
5.	Програмна реалізація	01.11.2020	виконано
6.	Опис програмної реалізації	23.11.2020	виконано
7.	Оформлення роботи	01.12.2020	виконано
8.	Представлення роботи до захисту	17.12.2020	виконано

Студент _____ Левченко Р.Р.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Руженцев В.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи: 73 с., 3 ч., 7 табл., 18 рис., 1 дод., 40 джерел.

НЕЙРОННІ МЕРЕЖІ, ПЕРСЕПТРОН, КРИПТОАНАЛІЗ, ШИФР ЗАМІЩЕННЯ-ПЕРЕСТАНОВКИ

Об'єкт дослідження – нейронні мережі.

Предмет дослідження – криптоаналіз простого шифру заміщення-перестановки за допомогою штучної нейронної мережі.

Мета роботи – дослідження нейронних мереж, аналіз можливості їх використання в криптоаналізі шифру заміщення-перестановки, програмна реалізація криптоаналізу за допомогою нейронних мереж.

У роботі розглянуто криптосистему нейронні мережі типу персептрон, їх використання у криптоаналізі.

Основні результати – проведено аналіз використання нейронних мереж типу персептрон для криптоаналізу заміщення-перестановки, створено програмну реалізація криптоаналізу за допомогою нейронних мереж.

ABSTRACT

Explanatory note to the certification work: 73 p., 3 p., 7 tables, 18 fig., 1 ap., 40 sources.

NEURAL NETWORKS, PERCEPTRON, CRYPTOANALYSIS, SUBSTITUTION PERMUTATION CYPHER

The object of study – neural networks.

The subject of research is cryptanalysis of a simple substitution-permutation cipher using an artificial neural network.

The purpose of the work – the study of neural networks, analysis of the possibility of their use in cryptanalysis of the substitution-permutation cipher, software implementation of cryptanalysis using neural networks.

The cryptosystem of neural networks of the perceptron type, their use in cryptanalysis is considered in the work.

The main results – an analysis of the use of neural networks such as perspectives for cryptanalysis of substitution-permutation cypher, created a software implementation of cryptanalysis using neural networks.

ЗМІСТ

ВСТУП

Нейрона криптографія – це галузь криптографії, присвячена аналізу застосування стохастичних алгоритмів, особливо алгоритмів штучної нейронної мережі, для використання в шифруванні та криптоаналізі [1].

Область використання штучних нейронних мереж (НМ) є слабо формалізованими багатофакторними проблемами. Як правило, чітких алгоритмів вирішення таких задач не існує. Аналіз криптографічної міцності блокових симетричних шифрів є однією з таких складних проблем.

У літературі є приклади успішного криптоаналізу деяких шифрів за допомогою нейронних мереж. У роботі [2] були описані атаки на алгоритми DES та Triple-DES. Для атак на DES потрібно лише 2^{11} пар відкритого тексту-зашифрованого тексту, щоб навчити нейронну мережу розшифровувати будь-який зашифрований текст. У той же час для найбільш відомих атак на DES потрібно понад 2^{40} таких пар, і цей алгоритм є одним з найбільш добре вивчених світовим криптографічним співтовариством протягом 40 років. Використання НМ дозволяє отримати результати, яких ще ніхто не отримував. Саме тому, НМ можуть стати актуальними як важливий елемент перевірки криптографічної міцності сучасних алгоритмів шифрування.

Метою даної роботи є вивчення можливості використання нейронних мереж для криптоаналізу шифрів заміщення-перестановки та пошук найбільш придатних для цієї архітектури нейронних мереж. Потрібно вибрати та реалізувати прості перетворення шифрування та дешифрування шифру заміщення-перестановки. На основі аналізу літератури вибрати архітектуру нейронної мережі. Вибрати бібліотеку для реалізації нейронної мережі. Виконувати обчислювальні експерименти криптоаналізу.

1 ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Визначення нейронної мережі

Штучні нейронні мережі (ШНМ) – це програмна імплементація нейронних структур мозку, що містить нейрони (рис. 1.1), які є свого роду органічними перемикачами [3]. Вони можуть змінювати тип переданих сигналів в залежності від електричних або хімічних сигналів, які в них передаються. Нейронна мережа у людському мозку – величезна взаємопов'язана система нейронів, де сигнал, який передається одним нейроном, може передаватися у тисячі інших нейронів. Навчання відбувається через повторну активацію деяких нейронних з'єднань. Через це збільшується імовірність виведення потрібного результату при відповідній вхідній інформації (сигналах). Такий вид навчання використовує зворотний зв'язок – при правильному результаті нейронні зв'язки, які виводять його, стають більш щільними.

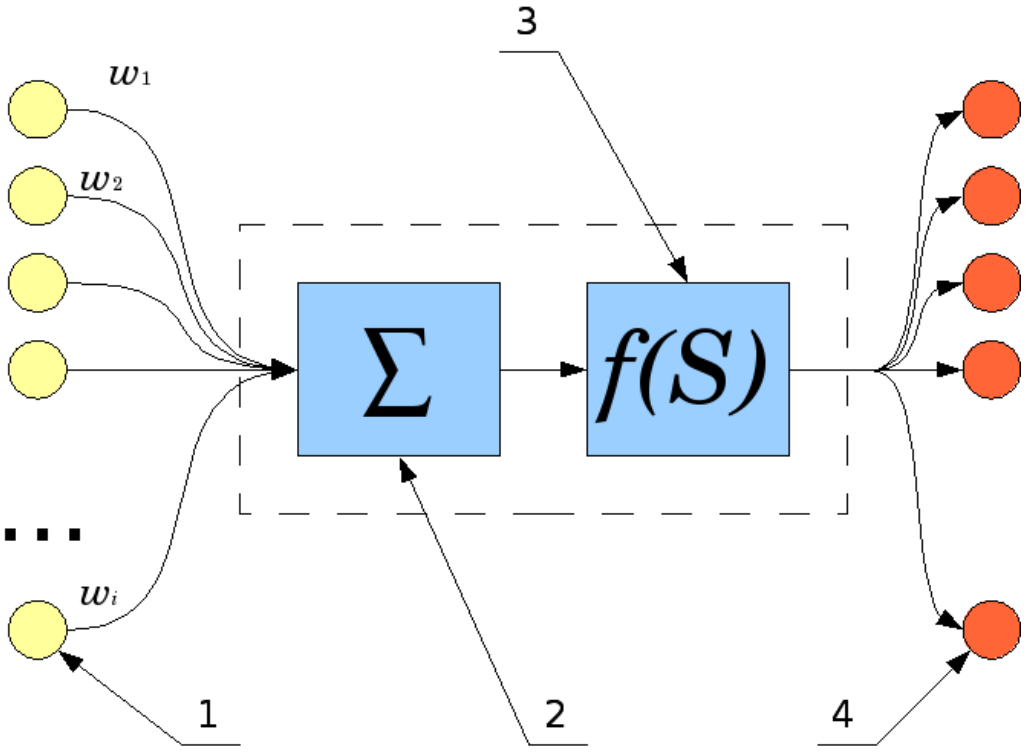

Рисунок 1.1 – Будова штучного нейрона

Штучні нейронні мережі імітують поведінку мозку у простішому вигляді. Вони можуть бути навчені контрольованим та неконтрольованим шляхами. У

контрольованій ШНМ, мережа навчається шляхом передавання відповідної вхідної інформації та прикладів вихідної інформації. Наприклад, спам-фільтр у електронній поштовій скриньці: вхідною інформацією може бути список слів, які зазвичай містяться у спам-повідомленнях, а вихідною інформацією – класифікація для відповідного повідомлення (спам, чи не спам). Такий вид навчання додає ваги зв'язкам ШНМ.

Неконтрольоване навчання у ШНМ намагається "змусити" ШНМ "зрозуміти" структуру переданої вхідної інформації "самостійно".

1.2 Принцип роботи нейронної мережі

Працює штучна нейронна мережа наступним чином: на входи нейронів надходять сигнали, які підсумовуються. При цьому враховується синаптична маса, тобто значимість кожного з входів. Далі, вхідні сигнали одних нейронів надходять на входи інших нейронів. Маса кожного такого зв’язку може бути позитивною (збуджуючі зв’язки) або негативною (гальмівні зв’язки). Вони визначають обчислення нейронної мережі, а значить її пам’ять та поведінку. Принцип імітує роботу нашого власного мозку.

Але для того, щоб штучна нейронна мережа виконувала складні завдання, її необхідно навчити.

1.3 Навчання штучної нейронної мережі

Штучні нейронні мережі не програмуються в звичайному розумінні цього слова, вони навчаються. Можливість навчання — одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно, навчання полягає в знаходженні коефіцієнтів зв’язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними й вихідними, а також здійснювати узагальнення.

Для процесу навчання необхідно мати модель зовнішнього середовища, у якій функціонує нейронна мережа – потрібну для вирішення задачі інформацію.

По-друге, необхідно визначити, як модифікувати вагові параметри мережі.

Існують три загальні парадигми навчання: кероване (“з вчителем”), “без вчителя” (самонавчання) та змішана. У першому випадку нейромережа має у своєму розпорядженні правильні відповіді (виходи мережі) на кожен вхідний приклад. Ваги налаштовуються так, щоб мережа виробляла відповіді найбільш близькі до відомих правильних відповідей. Навчання без вчителя не вимагає знання правильних відповідей на кожен приклад навчальної вибірки. У цьому випадку розкривається внутрішня структура даних та кореляція між зразками в навчальній множині, що дозволяє розподілити зразки по категоріях. При змішаному навчанні частина ваг визначається за допомогою навчання зі вчителем, у той час як інша визначається за допомогою самонавчання.

1.3.1 Контрольоване навчання

Алгоритми контрольованого навчання будують математичну модель набору даних, що містить як входи, так і бажані результати [4]. Дані відомі як дані навчання та складаються з набору навчальних прикладів. Кожен навчальний приклад має один або більше входів і бажаний вихід, також відомий як контрольний сигнал. У математичній моделі кожен приклад навчання представлений масивом або вектором, який іноді називають вектором ознак, а дані навчання представлені матрицею. Завдяки ітеративній оптимізації цільової функції алгоритми контрольованого навчання вивчають функцію, яка може бути використана для прогнозування результату, пов’язаного з новими входами [5]. Оптимальна функція дозволить алгоритму правильно визначити вихідні дані для входів, які не були частиною навчальних даних. Кажуть, що алгоритм, який покращує точність своїх результатів або прогнозів з часом, навчився виконувати це завдання [6].

Типи керованих алгоритмів навчання включають активне навчання, класифікацію та регресію [7]. Алгоритми класифікації використовуються, коли виходи обмежені кінцевим набором значень, а алгоритми регресії використовуються, коли виходи можуть мати будь-яке числове значення в межах діапазону. Як приклад, для алгоритму класифікації, який фільтрує електронні

листи, вхідним сигналом буде вхідне повідомлення електронної пошти, а вихідним буде ім'я папки, в яку слід подати електронне повідомлення.

Навчання на схожості – це область контрольованого навчання під наглядом, тісно пов’язана з регресією та класифікацією, але мета полягає в тому, щоб навчитися на прикладах, використовуючи функцію подібності, яка вимірює схожість або пов’язаність двох об’єктів. Він має програми для ранжирування, систем рекомендацій, візуального відстеження особи, перевірки обличчя та перевірки спікера.

1.3.2 Навчання без учителя

Алгоритми навчання без учителя беруть набір даних, який містить лише вхідні дані, і знаходять у них структуру, наприклад, групування або кластеризацію точок даних. Алгоритми вивчають дані тестів, які не були марковані, класифіковані чи поділені на категорії. Замість того, щоб відповідати на відгуки, алгоритми навчання без учителя визначають спільність даних і реагують на основі наявні чи відсутні спільні якості у кожному новому фрагменті даних. Центральне застосування навчання без учителя полягає в області оцінки щільності в статистиці, наприклад, пошуку функції щільності ймовірності [8]. Хоча навчання без учителя охоплює й інші сфери, що включають узагальнення та пояснення особливостей даних.

Кластерний аналіз – це розподіл набору спостережень у підмножини (так звані кластери), щоб спостереження всередині одного кластеру були схожими за одним або кількома попередньо визначеними критеріями, тоді як спостереження, отримані з різних кластерів, відрізняються. Різні методи кластеризації роблять різні припущення щодо структури даних, які часто визначаються певною метрикою подібності та оцінюються, наприклад, внутрішньою компактністю або подібністю між членами одного кластера та розділенням, різницею між кластерами. Інші методи засновані на розрахунковій щільності та зв’язковості графіків.

1.3.3 Напів-контрольоване навчання

Напів-контрольоване навчання – середнє між навчанням без учителя (без будь-яких позначених навчальних даних) та контрольованим навчанням (із повністю позначеними даними про навчання). У деяких прикладах навчання відсутні ярлики навчання, проте багато дослідників машинного навчання виявили, що дані без ярликів, використовуючи їх разом із невеликою кількістю мічених даних, можуть значно покращити точність навчання.

При напів-контрольованому навчанні ярлики тренувань бувають шумними, обмеженими або неточними; однак ці ярлики часто дешевше отримувати, що призводить до більших ефективних навчальних комплектів [9].

1.3.4 Підкріплене навчання

Підкріплене навчання – це область машинного навчання, яка стосується того, як програмні агенти повинні вживати дії в середовищі, щоб максимізувати деяке уявлення про сукупну винагороду. Завдяки своїй загальності, галузь вивчається в багатьох інших дисциплінах, таких як теорія ігор, теорія управління, дослідження операцій, теорія інформації, оптимізація на основі моделювання, багатоагентні системи, інтелект ройу, статистика та генетичні алгоритми. У машинному навчанні середовище зазвичай представляється як процес прийняття рішення Маркова. Багато алгоритмів підкріпленого навчання використовують методи динамічного програмування [10]. Алгоритми підкріпленого навчання не передбачають знання точної математичної моделі прийняття рішення Маркова і використовуються, коли точні моделі неможливі. Вони використовуються в автономних транспортних засобах або при навчанні грати проти людини.

1.3.5 Самонавчання

Самонавчання як парадигма машинного навчання було запроваджено в 1982 р. Разом із нейронною мережею, здатною до самонавчання, з назвою адаптивний поперечний масив (АПМ) [11]. Це навчання без зовнішніх винагород і поза зовнішніми порадами вчителя. Алгоритм самонавчання АПМ обчислює,

поперечно, як рішення щодо дій, так і емоції (почуття) щодо ситуацій наслідків. Система керується взаємодією між пізнанням та емоціями. Алгоритм самонавчання оновлює матрицю пам'яті $W = ||w(a,s)||$ такий, що в кожній ітерації виконується наступна рутинна програма машинного навчання:

У ситуації s виконати дію a;
Отримати наслідкові ситуації s';
Обчислити розпізнавання перебування в наслідковій ситуації $v(s')$;
Оновити пам'ять поперечного масиву $w'(a,s) = w(a,s) + v(s')$.

Це система, що має лише один вхід, ситуацію s і лише один вихід, дію (або поведінку) a. Немає ні окремого підкріплення, ні рекомендацій від навколишнього середовища. Значення зворотного розповсюдження (вторинне підкріплення) – це розпізнавання до ситуації наслідків. АПМ існує у двох середовищах, одне – поведінкове середовище, де він поводиться, а друге – генетичне середовище, звідки воно спочатку і лише один раз отримує початкові розпізнавання щодо ситуацій, з якими слід стикатися в поведінковому середовищі. Отримавши вектор генома (виду) із генетичного середовища, АПМ навчається цілеспрямованій поведінці в середовищі, що містить як бажані, так і небажані ситуації [12].

1.4 Види контрольованого навчання

Контрольоване навчання [13] в основному ділиться на дві частини, які є такими

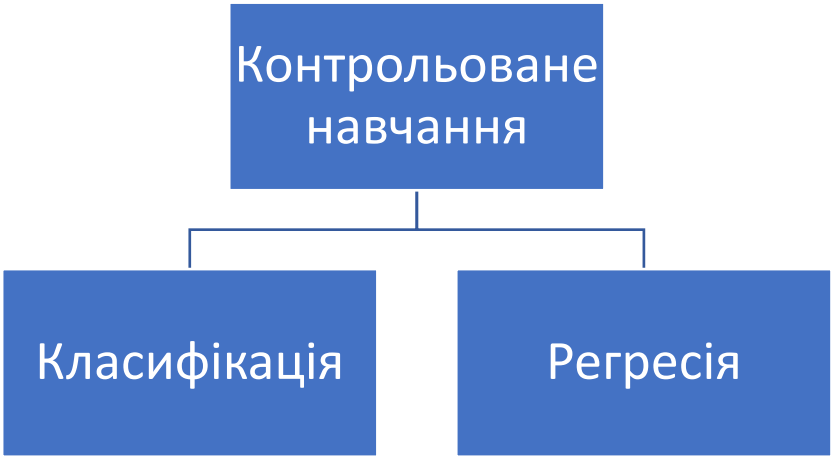

Рисунок 1.2 – Види контрольованого навчання

1.4.1 Регресія

Регресія – це тип контрольованого навчання, при якому використовуються позначені дані, і ці дані використовуються для прогнозування у безперервній формі. Вихід вхідних даних постійно триває, а графік є лінійним. Регресія – це форма техніки прогнозного моделювання, яка досліджує взаємозв'язок між залежною змінною [Виходи] та незалежною змінною [Входи]. Ця техніка використовується для прогнозування погоди, моделювання часових рядів, оптимізації процесів.

У машинному навчанні існує багато алгоритмів регресії, які будуть використовуватися для різних програм регресії. Деякі з основних алгоритмів регресії такі.

1.4.1.1 Проста лінійна регресія

У простій лінійній регресії ми прогнозуємо оцінки для однієї змінної з оцінок для другої змінної. Змінна, яку ми прогнозуємо, називається змінною критерію і позначається як Y. Змінна, на якій ми базуємо свої прогнози, називається змінною предиктора і позначається як X.

1.4.1.2 Багатолінійна регресія

Множинна лінійна регресія є одним із алгоритмів техніки регресії, і це найпоширеніша форма аналізу лінійної регресії. Як прогнозний аналіз множинна лінійна регресія використовується для пояснення зв'язку між однією залежною змінною з двома або більше, ніж двома незалежними змінними. Незалежні змінні можуть бути безперервними або категоріальними.

1.4.1.3 Поліноміальна регресія

Поліноміальна регресія – це ще одна форма регресії, при якій максимальна потужність незалежної змінної перевищує 1. У цій техніці регресії лінія, що найкраще підходить, не є прямою лінією, натомість вона має форму кривої.

1.4.1.4 Підтримуюча векторна регресія

Підтримуюча векторна регресія може застосовуватися не тільки до проблем регресії, але вона також застосовується у випадку класифікації. Вона містить усі функції, що характеризують алгоритм максимального відступу. Лінійне навчання машинного відображення накладає нелінійну функцію на простір ознак високої розмірності, індуковане ядром. Ємність системи контролювалася параметрами, які не залежать від розмірності простору ознак.

1.4.1.5 Крайня регресія

Крайня регресія – один з алгоритмів, що використовується в техніці регресії. Це методика аналізу даних множинної регресії, які страждають від мультиколінеарності. Додаючи ступінь упередженості до обчислень регресії, це зменшує стандартні помилки. Чистий ефект полягатиме в наданні більш надійних розрахунків.

1.4.1.6 Регресія Лассо

Регресія Лассо – це тип лінійної регресії, що використовує усадку. Зменшення - це місце, де значення даних зменшуються до центральної точки, як середнє значення. Процедура ласо заохочує прості, розріджені моделі (тобто моделі з меншою кількістю параметрів). Цей конкретний тип регресії добре підходить для моделей, що демонструють високий рівень мультиколінеарності, або коли ви хочете автоматизувати певні частини вибору моделі, такі як вибір змінних / усунення параметрів.

1.4.1.7 Еластична мережева регресія

Еластична мережева регресія поєднувала норми L1 (LASSO) та норми L2 (крайню регресію) у покарану модель для узагальненої лінійної регресії, і це надає їй розрідженість (L1) та властивості стійкості (L2).

1.4.1.8 Байєсова регресія

Байєсова регресія дозволяє досить природним механізмом пережити

недостатньо даних або погано розподілені дані. Це дозволить вам поставити коефіцієнти на пріоритет і шум, щоб пріоритети могли взяти на себе при відсутності даних. Що ще важливіше, ви можете запитати регресію Байєса, які частини (якщо такі є) його придатності до даних, у яких вона впевнена, а які частини дуже невизначені.

1.4.1.9 Регресія дерева рішень

Дерево рішень будує форму на зразок деревної структури з регресійних моделей. Він розбиває дані на менші підмножини, і в той час як пов'язане дерево рішень розробляється поступово одночасно. Результат – дерево з вузлами прийняття рішень та вузлами листя.

1.4.1.10 Випадкова лісова регресія

Випадковий ліс – це також один з алгоритмів, що використовується в техніці регресії, і це дуже гнучкий, простий у використанні алгоритм машинного навчання, який створює навіть без налаштування гіперпараметрів. Крім того, цей алгоритм широко використовується через його простоту та той факт, що він може використовуватися як для регресії, так і для класифікаційних завдань. Ліс, який він будує, являє собою ансамбль дерев рішень, які більшу частину часу навчали методом «мішків».

1.4.2 Класифікація

Класифікація – це тип контрольованого навчання, в якому можуть використовуватися марковані дані, і ці дані використовуються для прогнозування у неперервній формі. Вихід інформації не завжди є безперервним, а графік нелінійний. У техніці класифікації алгоритм навчається на введених йому даних, а потім використовує це навчання для класифікації нового спостереження. Цей набір даних може бути просто двокласним, а може бути і багатокласним.

У машинному навчанні існує багато алгоритмів класифікації, які використовуються для різних застосувань класифікації. Деякі з основних

алгоритмів класифікації наступні.

1.4.2.1 Логістична регресія / класифікація

Логістична регресія підпадає під категорію навчання під контролем; вона вимірює зв'язок між залежною змінною, яка є категоричною з однією або кількома незалежними змінними, оцінюючи ймовірності за допомогою логістичної / сигмоїдної функції. Логістична регресія, як правило, може використовуватися там, де залежною змінною є двійкова або дихотомічна. Це означає, що залежна змінна може приймати лише два можливі значення, такі як “Так чи Ні”, “Живий чи мертвий”.

1.4.2.2 K-Найближчі сусіди

Алгоритм KNN є одним з найпростіших алгоритмів у класифікації, і це один з найбільш часто використовуваних алгоритмів навчання. Більшість голосів об’єкта класифікується його сусідами, при цьому мета присвоюється класу, найпоширенішому серед його найближчих сусідів. Він також може використовуватися для регресії – вихід – це значення об'єкта (передбачає безперервні значення). Це значення є середнім показником (або медіаною) переваг його найближчих сусідів.

1.4.2.3 Машини опорного вектору

Машини опорного вектору – це тип класифікатора, в якому дискримінаційний класифікатор, формально визначений розділовою гіперплощиною. Алгоритм виводить оптимальну гіперплощину, яка класифікує нові приклади. У двовимірному просторі ця гіперплощина являє собою лінію, що розділяє площину на дві частини, де кожен клас лежить по обидві сторони.

1.4.2.4 Машини опорного вектору ядра

Алгоритм машин опорного вектору ядра або Kernel-SVM – це один із алгоритмів, що використовується в техніці класифікації, і саме набір математичних функцій визначається як ядро. Мета ядра – взяти дані як вхідні

дані та перетворити їх у необхідну форму. Різні алгоритми SVM використовують різні типи функцій ядра. Ці функції можуть бути різних типів. Наприклад, лінійні та нелінійні функції, поліноміальні функції, радіальна базисна функція та сигмоїдні функції.

1.4.2.5 Простий Байєс

Простий Байєс – це тип класифікаційної техніки, який базується на теоремі Байєса з припущенням незалежності серед провісників. Простіше кажучи, класифікатор Наївського Байєса припускає, що наявність певної ознаки в класі не пов’язана з наявністю будь-якої іншої функції. Наївна модель Байєса доступна для побудови і особливо корисна для великих наборів даних.

1.4.2.6 Класифікація дерева рішень

Дерево рішень робить класифікаційні моделі у вигляді деревної структури. Пов’язане дерево рішень поступово розвивається і одночасно воно розбиває великий набір даних на менші підмножини. Кінцевим результатом є дерево з вузлами прийняття рішень та вузлами листя. Вузол прийняття рішення (наприклад, Root) має дві або більше гілок. Листовий вузол представляє класифікацію або рішення. Перший вузол прийняття рішення у дереві, який відповідає найкращому провіснику, який називається кореневим вузлом. Дерева рішень можуть обробляти як категоріальні, так і числові дані.

1.4.2.7 Класифікація випадкових лісів

Випадковий ліс – це керований алгоритм навчання. Він створює ліс і робить його якимось чином випадковим. Дерево, яке він будує, являє собою ансамбль дерев рішень, що більшу частину часу будучи алгоритмом дерева рішень, навченого методом “мішків”, що є поєднанням моделей навчання, збільшує загальний результат.

1.5 Види навчання без учителя

Навчання без учителя в основному ділиться на дві частини.

1.5.1.Кластеризація

Кластеризація – це тип неконтрольованого навчання, при якому використовуються немічені дані, і це процес групування подібних сутностей разом, а потім згруповані дані використовуються для створення кластерів. Метою цієї безконтрольної техніки машинного навчання є знайти схожість у точці даних та об'єднати подібні точки даних разом, а також з'ясувати, що до яких кластерів мають належати нові дані.

У машинному навчанні існує багато алгоритмів кластеризації, який використовується для різних кластерних програм. Деякі з основних алгоритмів кластеризації є такими.

1.5.1.1 K-засіб кластеризації

K-засіб кластеризації – це один з алгоритмів техніки кластеризації, в якому подібні дані групуються в кластер. K-засіб – це ітеративний алгоритм кластеризації, який спрямований на пошук локальних максимумів у кожній ітерації. Починається з K як вхідних даних, тобто скільки груп ви хочете бачити. Введіть k центроїдів у випадкових місцях у вашому просторі. Тепер, використовуючи метод евклідової відстані, обчислюють відстань між точками даних та центроїдами і призначають точку даних кластеру, який знаходиться близько до нього. Перерахуйте центри кластера як середнє значення прикріплених до нього точок даних. Повторюйте, поки подальших змін не відбудеться.

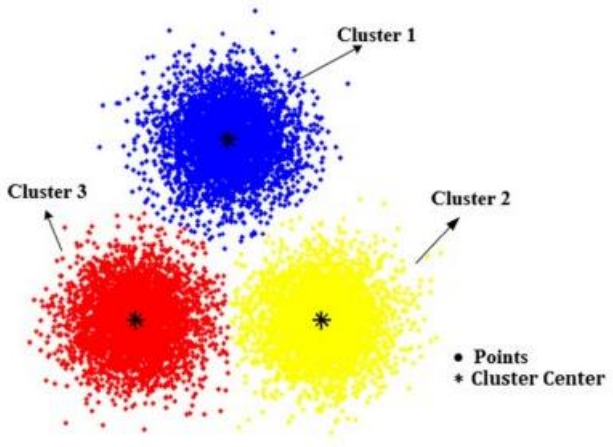

Рисунок 1.3 – K-засіб кластеризації, що показує 3 кластери

1.5.1.2 Ієрархічна кластеризація

Ієрархічна кластеризація – один із алгоритмів техніки кластеризації, в якому подібні дані групуються в кластер. Це алгоритм, який будує ієрархію кластерів. Цей алгоритм починається з усіх точок даних, призначених їх власній множині. Потім дві найближчі групи об'єднуються в один і той же кластер. Зрештою, цей алгоритм припиняється, коли залишився лише один кластер. Починається з призначення кожної точки даних для своєї множині. Потім пошук найближчої пари групи за допомогою евклідової відстані та об'єднання їх в єдине скупчення. Обчислення відстані між двома найближчими кластерами та об'єднання, поки всі елементи не об'єднаються в єдиний кластер.

1.5.2 Зменшення розмірності

Зменшення розмірності – це тип навчання без учителя, при якому розміри даних зменшуються, щоб видалити небажані дані з вводу. Ця методика використовується для видалення небажаних особливостей даних. Це стосується процесу перетворення набору даних, що мають великі розміри, у дані з однаковими даними та малими розмірами. Ці методи використовуються під час вирішення задач машинного навчання для отримання кращих характеристик.

У машинному навчанні існує багато алгоритмів зменшення розмірності, які застосовуються для різних програм зменшення розмірності. Деякі основні алгоритми зменшення розмірності такі.

1.5.2.1 Аналіз основних компонентів

Аналіз основних компонентів є одним із алгоритмів зменшення розмірності, у цій техніці він перетворений у новий набір змінних зі старих змінних, які є лінійною комбінацією реальних змінних. Конкретний новий набір змінних відомий як основні компоненти. У результаті трансформації перший первинний компонент має найбільш істотну можливу дисперсію, і кожен наступний елемент має найбільшу різницю потенціалів за умови, що він ортогональний вищевказаним інгредієнтам. Зберігання лише перших $m < n$ компонентів зменшує розмірність даних, зберігаючи більшу частину інформації.

1.5.2.2 Лінійний дискримінантний аналіз

Лінійний дискримінантний аналіз – це один із алгоритмів зменшення розмірності, у якому він також створює лінійні комбінації вихідних рис. Однак, на відміну від аналізу основних елементів, лінійний дискримінантний аналіз не максимізує пояснювану дисперсію. Натомість це оптимізує поділ між класами. Лінійний дискримінантний аналіз може покращити прогнозуючу ефективність вилучених функцій. Крім того, він пропонує варіанти для подолання конкретних перешкод.

1.5.2.3 Аналіз основних компонентів ядра

Аналіз основних компонентів ядра є одним з алгоритмів зменшення розмірності, і змінні, які перетворюються у змінні нового набору, які є нелінійною комбінацією вихідних змінних, означають нелінійну версію аналізу основних компонентів, що називається аналізом основних компонентів ядра. Він здатний фіксувати частину статистики вищого порядку, таким чином надаючи більше інформації з вихідного набору даних.

1.6 Види алгоритмів підкріпленого навчання

У машинному навчанні існує багато алгоритмів армірувального навчання,

які застосовуються для різних додатків підкріпленого навчання. Деякі основні алгоритми такі.

1.6.1 Q-навчання

Q-навчання – це один з алгоритмів Підкріпленого навчання, в якому агент намагається вивчити оптимальну стратегію зі своєї історії спілкування з навколишнім середовищем. Запис агента – це послідовність стан-дія-винагорода. Q-навчання вивчає оптимальну лінія поведінки, незалежно від того, якої процедури дотримується агент, якщо немає обмежень на кількість разів, коли він намагається виконати дію в будь-якому стані. Оскільки вона засвоює оптимальну лінію поведінки, незалежно від того, яку стратегію вона проводить, це називається методом, що не стосується лінії поведінки.

1.6.2 SARSA (Стан-дія-винагорода-стан-дія)

SARSA – це один із алгоритмів підкріпленого навчання, у якому він визначає його оновленим значенням дії. Це незначна різниця між реалізацією SARSA та Q-навчання, але це викликає глибокий ефект. Метод SARSA приймає інший параметр, action2, який є дією, здійсненою агентом із другого стану. Це дозволяє агенту явно знайти майбутнє значення винагороди. Далі, що слідувало, замість того, щоб припускати, що буде використана оптимальна дія і що найвища винагорода.

1.6.3 Глибока мережа Q

Глибока Q-мережа є одним із алгоритмів підкріпленого навчання, хоча Q-навчання є дуже надійним алгоритмом, його головною вадою є відсутність загальності. Якщо розглядати Q-навчання як відновлення чисел у двовимірному масиві (Дієвий простір $*$ Простір станів), воно, насправді, слідує динамічному програмуванню. Це вказує на те, що для станів, яких агент Q-навчання раніше не бачив, він не знає, яку дію вжити. Іншими словами, агент Q-навчання не може оцінити значення для станів, що він не бачив. Для вирішення цієї проблеми глибока Q-мережа позбавляється двовимірного масиву, представляючи

нейронну мережу.

1.6.4 Процеси рішення Маркова

Процес прийняття рішень Маркова – один із алгоритмів підкрипленного навчання, в якому він містить набір можливих світових станів s, набір моделей, сукупність можливих дій a, дієву функцію винагороди $R(s, a)$, лінія поведінки рішення процесу прийняття рішень Маркова. Для досягнення мети використовується Процес прийняття рішень Маркова – це пряме формулювання проблеми навчання на взаємодії. Агент підбирав дії та середовище, що відповідає на ці дії, а агент і середовище постійно взаємодіють і представляють агенту нові ситуації.

1.6.5 DDPG (Глибокий детермінований градієнт лінії поведінки)

Глибокий детермінований градієнт лінії поведінки – це один із алгоритмів підкріпленого навчання, в якому він спирається на дизайн актора-критика з двома однойменними компонентами – актор та критик. Актор використовується для налаштування параметра θ для функції лінії поведінки, тобто вирішення найкращої дії для конкретного стану. Ідеї окремої цільової мережі та відтворення досвіду також запозичені у глибокій Q-мережі. Інша проблема для DDPG – це рідкісне проведення дослідних робіт. Рішенням цього є додавання шуму до простору параметрів або простору дій.

1.7 Види штучних нейронних мереж

Штучні нейронні мережі – обчислювальні моделі, натхненні біологічними нейронними мережами, і використовуються для наближення функцій, які, як правило, невідомі. Зокрема, вони надихаються поведінкою нейронів та електричними сигналами, які вони передають між входом (наприклад, від очей або нервових закінчень у руці), обробкою та виходом з мозку (наприклад, реакцією на світло, дотик або тепло). Спосіб семантичного спілкування нейронів є сферою постійних досліджень. Більшість штучних нейронних мереж мають лише деяку схожість із своїми більш складними біологічними аналогами, але

дуже ефективні при виконанні своїх запланованих завдань (наприклад, класифікація або сегментація).

Деякі штучні нейронні мережі є адаптивними системами і використовуються, наприклад, для моделювання популяцій та середовищ, які постійно змінюються.

Нейронні мережі можуть бути апаратними (нейрони представлені фізичними компонентами) або програмними (комп’ютерні моделі), і можуть використовувати різні топології та алгоритми навчання.

1.7.1 Прямого поширення

Нейронна мережа прямого поширення була першим і найпростішим типом. У цій мережі інформація переміщується лише від вхідного шару безпосередньо через будь-які приховані шари до вихідного шару без циклів. Мережі прямого зв'язку можуть бути побудовані з різними типами одиниць, такими як двійкові нейрони Мак-Каллока – Піттса, найпростішим з яких є персептрон. Безперервні нейрони, часто з сигмоїдальною активацією, використовуються в контексті зворотного розмноження [14].

1.7.2 Регулюючого зворотного поширення

Мережі регулюючого зворотного поширення почали працювати як модель для пояснення мозкових явищ, виявлених під час розпізнавання, включаючи загальний розрив мережі та труднощі зі схожістю, які часто зустрічаються в сенсорному розпізнаванні. Механізм для здійснення оптимізації під час розпізнавання створюється за допомогою з'єднань гальмуючого зворотного зв'язку назад до тих самих входів, які їх активують. Це зменшує вимоги під час навчання та дозволяє легше навчатись та оновлювати, а в той же час може виконувати складне розпізнавання.

1.7.3 Функція радіального (зоряного) базису

Функції радіального базису (ФРБ) – це функції, які мають критерій відстані відносно центру. Функції радіального базису були застосовані як заміна

сигмоїдальної характеристики прихованого шару в багатошарових персептронах [15]. Мережі ФРБ мають два шари: у першому вхідні дані відображаються на кожному ФРБ у "прихованому" рівні. Вибір ФРБ, як правило, є Гауссовим. У задачах регресії вихідний рівень являє собою лінійну комбінацію значень прихованого шару, що представляють середній прогнозований вихід. Інтерпретація цього значення вихідного рівня така ж, як і модель регресії в статистиці. У проблемах класифікації вихідний рівень, як правило, є сигмовидною функцією лінійної комбінації значень прихованого шару, що представляє задню ймовірність. Ефективність в обох випадках часто покращується методами усадки, відомими в класичній статистиці як регресія хребта. Це відповідає попередній вірі в малі значення параметрів (і, отже, плавні вихідні функції) в байєсівських структурах.

Мережі ФРБ мають ту перевагу, що уникають локальних мінімумів так само, як багатошарові персептрони. Це пояснюється тим, що єдиними параметрами, які коригуються в процесі навчання, є лінійне відображення від прихованого шару до вихідного шару. Лінійність гарантує, що поверхня похибки є квадратичною і, отже, має єдиний легко пошуковий мінімум. У задачах регресії це можна знайти в одній матричній операції. У класифікаційних задачах з фіксованою нелінійністю, введеною функцією сигмоїдного виходу, найефективніше розглядається використання ітеративно повторно зважених найменших квадратів.

Недолік ФРБ полягає в тому, що вони вимагають хорошого покриття вхідного простору радіальними базовими функціями. Центри ФРБ визначаються з посиланням на розподіл вхідних даних, але без посилання на завдання прогнозування. Як результат, представницькі ресурси можуть бути витрачені даремно на ділянки вхідного простору, які не мають значення для завдання. Загальним рішенням є асоціювання кожної точки даних із власним центром, хоча це може розширити лінійну систему, що вирішується в остаточному шарі, і вимагає техніки усадки, щоб уникнути переобладнання.

Пов’язування кожного вхідного даного з ФРБ природно веде до методів ядра, таких як підтримка векторних машин (SVM) та гауссових процесів (ФРБ –

це функція ядра). Всі три підходи використовують нелінійну функцію ядра для проектування вхідних даних у простір, де навчальну проблему можна вирішити за допомогою лінійної моделі. Подібно гауссовим процесам, на відміну від SVM, мережі ФРБ, як правило, навчаються в рамках з максимальною вірогідністю, максимізуючи ймовірність (мінімізуючи помилку). SVM уникають переобладнання, максимізуючи замість цього запас. SVM перевершують мережі ФРБ у більшості програм класифікації. У програмах регресії вони можуть бути конкурентоспроможними, коли розмірність вхідного простору відносно мала.

1.7.4 Рекурентна нейронна мережа

Рекурентні нейронні мережі (РНМ) поширюють дані вперед, але також і назад, від пізніших етапів обробки до попередніх етапів. РНМ може використовуватися як процесор загальної послідовності.

1.7.4.1 Повністю рекурентні

Ця архітектура була розроблена у 1980-х. Його мережа створює спрямований зв’язок між кожною парою одиниць. Кожен з них має різну в часі, реальну вартість (більше, ніж просто нуль або одиницю) активації (вихід). Кожне з'єднання має змінну реальну вагу. Деякі вузли називаються позначеними, деякі вихідні вузли, решта прихованими вузлами.

Для контрольованого навчання в умовах дискретного часу послідовності тренувань реальних вхідних векторів стають послідовностями активації вхідних вузлів, по одному вхідному вектору за раз. На кожному кроці часу кожна невхідна одиниця обчислює свою поточну активацію як нелінійну функцію зваженої суми активацій усіх одиниць, від яких вона отримує з'єднання. Система може явно активувати (незалежно від вхідних сигналів) деякі вихідні блоки на певних етапах часу. Наприклад, якщо вхідна послідовність є мовним сигналом, що відповідає вимовленій цифрі, кінцевим цільовим виходом в кінці послідовності може бути мітка, що класифікує цифру. Для кожної послідовності її похибка є сумою відхилень усіх активацій, обчислених мережею, від відповідних цільових сигналів. Для навчального набору з численних

послідовностей загальна помилка – це сума помилок усіх окремих послідовностей.

Щоб мінімізувати загальну похибку, градієнтний спуск можна використовувати для зміни кожної ваги пропорційно її похідній щодо похибки, за умови, що нелінійні функції активації можна диференціювати. Стандартний метод називається "зворотне поширення у часі" або BPTT (backpropagation through time), узагальнення зворотного поширення для прямих мереж [16] [17]. Більш обчислювально дорогий онлайн варіант називається "Повторне навчання в режимі реального часу" або RTRL (Real-Time Recurrent Learning) [18]. На відміну від BPTT, цей алгоритм локальний у часі, але не локальний у просторі [19]. Існує мережевий гібрид між BPTT та RTRL з проміжною складністю [20] з варіантами для безперервного часу [21]. Основною проблемою градієнтного спуску для стандартних архітектур РНМ є те, що градієнти помилок швидко зникають з величиною затримки між важливими подіями [22]. Архітектура довгострокової короткочасної пам'яті долає ці проблеми.

У налаштуваннях навчального підкріплення жоден учитель не подає цільові сигнали. Натомість для оцінки продуктивності час від часу використовують функцію фітнесу, функцію винагороди чи функцію корисності, яка впливає на її вхідний потік через вихідні блоки, підключені до виконавчих механізмів, що впливають на навколишнє середовище. Варіанти еволюційних обчислень часто використовуються для оптимізації вагової матриці.

Мережа Хопфілда (як і подібні мережі на основі атракторів) представляє історичний інтерес, хоча вона не є загальною РНМ, оскільки вона не призначена для обробки послідовностей шаблонів. Натомість для цього потрібні стаціонарні входи. Це РНМ, в якому всі з'єднання симетричні. Це гарантує зближення. Якщо зв’язки навчаються за допомогою Геббійського навчання, мережа Хопфілда може працювати як надійна пам'ять, що адресується змістом, стійка до змін з'єднання.

Машину Больцмана можна сприймати як галасливу мережу Хопфілда. Це одна з перших нейронних мереж, яка продемонструвала засвоєння прихованих змінних (прихованих одиниць). Спочатку машинне навчання Больцмана

моделювалось повільно, але протилежний алгоритм розбіжності прискорює навчання машин Больцмана та продуктів експертів.

Карта самоорганізації (SOM) використовує навчання без нагляду. Набір нейронів вчиться відображати точки у вхідному просторі в координати у вихідному просторі. Вхідний простір може мати різні розміри та топологію від вихідного простору, і SOM намагається їх зберегти.

Квантування векторного навчання (LVQ) можна інтерпретувати як архітектуру нейронної мережі. Прототипові представники класів параметризуються разом із відповідною мірою відстані у схемі класифікації на відстані.

1.7.4.2 Прості рекурентні

Прості рекурентні мережі мають три шари з додаванням набору "одиниць контексту" у вхідний рівень. Ці блоки з'єднуються із прихованого шару або вихідного шару з фіксованою вагою в один. [23] На кожному часовому кроці вхідні дані розповсюджуються стандартним способом прямої передачі, а потім застосовується правило навчання, подібне до зворотного розповсюдження (не виконуючи градієнтного спуску). Фіксовані зворотні зв’язки залишають копію попередніх значень прихованих одиниць у контекстних одиницях (оскільки вони поширюються по зв’язках до того, як застосовується правило навчання).

1.7.4.3 Обчислення резурвуару

Обчислення резурвуару – це обчислювальна система, яку можна розглядати як продовження нейронних мереж. [24] Зазвичай вхідний сигнал подається у фіксовану (випадкову) динамічну систему, яка називається резервуаром, динаміка якого відображає вхідні дані у більший вимір. Навчений механізм зчитування для відображення пласта до бажаного виходу. Навчання проводиться лише на етапі зчитування.

1.7.4.4 Довга короткочасна пам’ять

Довга короткочасна пам’ять або The long short-term memory (LSTM) [25]

дозволяє уникнути проблеми зникнення градієнта. Вона працює навіть при великих затримках між входами і може обробляти сигнали, що поєднують низькочастотні та високочастотні компоненти. LSTM РНМ перевершила інші РНМ та інші методи послідовного навчання, такі як НММ (Hidden Markov model), у таких програмах, як вивчення мови [26] та підключене розпізнавання рукописного вводу [27].

1.7.4.5 Двонаправлена

Двонаправлені РНМ, або ДРНМ, використовують кінцеву послідовність, щоб передбачити або позначити кожен елемент послідовності на основі як минулого, так і майбутнього контексту елемента. [28] Це робиться шляхом додавання виходів двох РНМ: одна обробляє послідовність зліва направо, інша справа наліво. Комбіновані результати – це передбачення цільових сигналів, поданих викладачем. Цей прийом виявився особливо корисним у поєднанні з LSTM.

1.7.4.6 Генетична шкала

РНМ (часто LSTM), де серія розкладається на ряд шкал, де кожна шкала повідомляє первинну довжину між двома послідовними точками. Масштаб першого порядку складається з нормальної РНМ, другий порядок складається з усіх точок, розділених двома індексами тощо. РНМ N-го порядку з'єднує перший і останній вузол. Виходи з усіх різних шкал розглядаються як Комітет машин [29], і пов'язані результати використовуються генетично для наступної ітерації.

1.7.5 Модульні

Біологічні дослідження показали, що мозок людини працює як сукупність невеликих мереж. Ця реалізація породила концепцію модульних нейронних мереж, в якій кілька малих мереж співпрацюють або конкурують для вирішення проблем.

1.7.5.1 Комітет машин

Комітет машин (КМ) – це сукупність різних нейронних мереж, які разом

"голосують" на даному прикладі. Як правило, це дає набагато кращий результат, ніж окремі мережі. Оскільки нейронні мережі страждають від локальних мінімумів, починаючи з тієї ж архітектури та навчання, але використовуючи випадково різні початкові ваги, часто дають значно різні результати КМ прагне стабілізувати результат.

КМ схожий із загальним методом машинного навчання мішків, за винятком того, що необхідна різноманітність машин у комітеті отримується шляхом навчання з різною стартовою вагою, а не навчанням на різних випадково вибраних підмножинах даних навчання.

1.7.5.2 Асоціативні

Асоціативна нейронна мережа (АНМ) – це розширення комітету машин, який поєднує в собі кілька нейронних мереж прямої передачі та техніку k-найближчого сусіда (kNN). Вона використовує кореляцію між відповідями ансамблю як міру відстані серед аналізованих випадків для kNN. Це виправляє зміщення ансамблю нейронних мереж. АНМ має пам’ять, яка може збігатися з навчальним набором. Якщо нові дані стають доступними, мережа миттєво покращує свої передбачувальні можливості та забезпечує апроксимацію даних (самостійне навчання) без перекваліфікації. Ще однією важливою особливістю АНМ є можливість інтерпретації результатів нейронної мережі шляхом аналізу співвідношень між випадками даних у просторі моделей [30].

1.7.6 Фізичні

Фізична нейронна мережа включає електрично регульований матеріал опору для імітації штучних синапсів. Приклади включають нейронну мережу ADALINE на основі мемристора. [31] Оптична нейронна мережа – це фізична реалізація штучної нейронної мережі з оптичними компонентами.

1.8 Найпоширеніші застосування нейронних мереж

Розпізнавання образів та класифікація. В якості образів можуть виступати

різні за своєю природою об’єкти: символи тексту, зображення, зразки звуків і т. д. При навчанні мережі пропонуються різні зразки образів із зазначенням того, до якого класу вони відносяться. Коли мережі пред’являється якийсь образ, на одному з її виходів повинна з’явитися ознака того, що образ належить цьому класу. У той же час на інших виходах повинна бути ознака того, що образ до даного класу не належить.

Прийняття рішень та управління. Це завдання близьке до задачі класифікації. Класифікації підлягають ситуації, характеристики яких надходять на вхід нейронної мережі. На виході мережі повинна з’явитися ознака рішення, яке вона прийняла.

Кластеризація. Під кластеризацією розуміється розбиття множини вхідних сигналів на класи, при тому, що ні кількість, ні ознаки класів заздалегідь не відомі. Після навчання така мережа здатна визначати, до якого класу належить вхідний сигнал.

Прогнозування. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення та виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень або якихось існуючих зараз чинників.

Стиснення даних і асоціативна пам’ять. Здатність нейромереж до виявлення взаємозв’язків між різними параметрами дає можливість висловити дані великої розмірності більш компактно, якщо дані тісно взаємопов’язані між собою. Зворотній процес – відновлення вихідного набору даних з частини інформації – називається асоціативною пам’яттю.

1.9 Машинне навчання як підхід для криптоаналізу

У [32] Чандра автори використовували нейронні мережі для класифікації шифротексту на основі алгоритму, який використовувався для його шифрування. Вони використовували нейромережу каскадної кореляції та мережу зворотного поширення для ідентифікації шифрових систем. Для навчання вони

використовували шифротексти, отримані від Enhanced RC6, блок-шифру, і від SEAL - потокового шифру. Вони використовували різні типи наборів даних з однаковими ключами, різними ключами, однаковими наборами відкритих текстів, різними наборами відкритих текстів і т.д., і дійшли висновку, що каскадна кореляція працює краще, ніж метод зворотного поширення.

Алані ММ [33] висунув ідею зламати Data Encryption Standard (DES) за допомогою нейронної мережі. Автор застосував атаку з відомим відкритим текстом, щоб отримати відкритий текст. Алгоритм, який використовався автором, не намагається знайти ключ, а навпаки, намагається безпосередньо знайти відкритий текст. Хоча цей підхід не вважається криптографічною атакою, робота автора заслуговує уваги, оскільки автор сконструював нейронну мережу для процесу ідентифікації відкритого тексту, використовуючи той самий відкритий текст і шифротекст с тим же ключем.

У роботі, написаній Альбассалом та Вагданом [34], автори описали, як їм вдалося використовувати нейронні мережі для злому гіпотетичного шифру Фейстеля, званого HypCipher. Раундова функція для HypCipher була обрана із Advanced Encryption Standard (AES). Використаний метод зворотного поширення демонструє успіх у 2 та 3 раундах шифру. Додатковий прихований рівень був доданий для 4 раундів. Модель була успішною тим, що вона використовувала просту нейронну мережу з такою простою функцією активації, як сигмоїдна функція. Автори запропонували використовувати розподілену систему для атаки на шифри з більшою кількістю раундів.

2 ОПИС ШИФРУЮЧОГО ПЕРЕТВОРЕННЯ І НЕЙРОННОЇ МЕРЕЖІ

2.1 Алгоритм шифрування

Був запропонований наступний алгоритм шифрування. Наш вхід – 10-бітове відкрите повідомлення. Спочатку циклічний зсув на 7 бітів уліво і додавання XOR з ключем використовуються для всього блоку. Потім 10-бітовий блок поділяється на два 5-бітних підблоки, і для кожного підблоку виконуються заміни 5-ти бітів на 5-біт. Таким чином на виході отримуємо 10 біт. Запропонований один раунд алгоритму шифрування представлений на рис. 2.1.

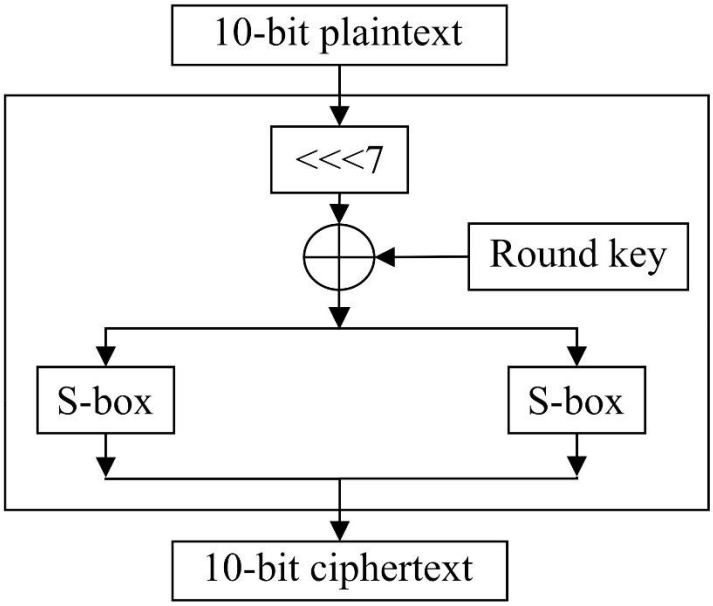

Рисунок 2.1 – Раунд алгоритму шифрування

Відповідний один раунд дешифрування з оберненими перетвореннями у зворотному порядку представлений на рис.2.2.

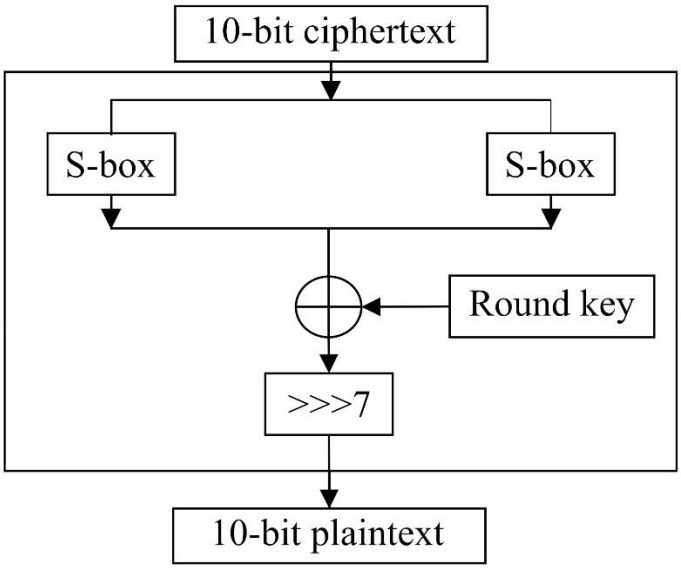

Рисунок 2.2 – Раунд алгоритму дешифрування

2.2 Персептрон

Перцептро́н, або персептро́н – математична або комп'ютерна модель сприйняття інформації мозком, запропонована Френком Розенблатом в 1957 році й реалізована у вигляді електронної машини «Марк-1» у. 1960 році [35]. Персептрон став однією з перших моделей нейромереж, а «Марк-1» – першим у світі нейрокомп'ютером. Незважаючи на свою простоту, персептрон здатен навчатися і розв'язувати досить складні завдання. Основна математична задача, з якою він здатний впоратися — це лінійне розділення довільних нелінійних множин, так зване забезпечення лінійної сепарабельності.

Персептрон складається з трьох типів елементів, а саме: сигнали, що надходять від давачів, передаються до асоціативних елементів, а відтак до реагуючих. Таким чином, персептрони дозволяють створити набір «асоціацій» між вхідними стимулами та необхідною реакцією на виході. В біологічному плані це відповідає перетворенню, наприклад, зорової інформації у фізіологічну відповідь рухових нейронів. Відповідно до сучасної термінології, персептрони може бути класифіковано як штучні нейронні мережі:

1) з одним прихованим шаром;
2) з пороговою передавальною функцією;

3) з прямим розповсюдженням сигналу.

Персептрон приймає кілька двійкових входів, x_1, x_2, ..., і видає єдиний двійковий вихід:

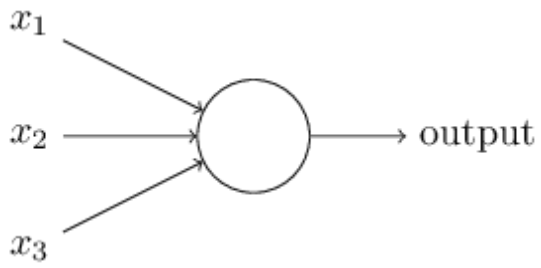

Рисунок 2.3 – Модель персептрона

У наведеному прикладі персептрон має три входи, x_1, x_2, x_3. Загалом він може мати більше або менше вхідних даних. Розенблат запропонував просте правило для обчислення результату. Він ввів міру ваги (надалі вага) – w_1, w_1, ..., реальні числа, що виражають важливість відповідних входів для виходу. Вихід нейрона, 0 або 1, визначається тим, чи зважена сума $\sum_j w_j x_j$ менша або більша за деяке порогове значення (надалі поріг). Як і ваги, поріг – дійсне число, яке є параметром нейрона. Якщо сказати точніше алгебраїчними термінами:

$$output = \begin{cases} 0 \; if \; \sum_j w_j x_j \leq \text{поріг} \\ 1 \; if \; \sum_j w_j x_j > \text{поріг} \end{cases} \tag{2.1}$$

Це основна математична модель.

У своїй книзі [36] Майкл Нільсен зазначає, що персептрон не є повною моделлю прийняття рішень людиною. Але персептрон може зважувати різні типи доказів для прийняття рішень. І повинно здатися правдоподібним, що складна мережа персептронів може приймати досить тонкі рішення.

На тлі зростання популярності нейронних мереж у 1969 році вийшла книга Марвіна Мінського та Сеймура Пейперта, що показала принципові обмеження персептронів. Це призвело до зміщення інтересу дослідників штучного інтелекту в протилежну від нейромереж область символьних обчислень. Крім того, через складність математичного аналізу персептронів, а також відсутність загальноприйнятої термінології, виникли різні неточності і помилки.

Згодом інтерес до нейромереж, і зокрема, робіт Розенблата, поновився.

Так, наприклад, зараз стрімко розвивається біокомп'ютинг, що у своїй теоретичній основі обчислень, зокрема, базується на нейронних мережах, а персептрон відтворюють на базі бактеріородопсинмісних плівок.

2.3 Багатошаровий персептрон

Багатошаровий персептрон (БШП) – це клас штучної нейронної мережі з прямим розповсюдженням (ШНМ) [37]. Термін БШП використовується неоднозначно, іноді вільно до будь-якої ШНМ з прямим розповсюдженням, іноді суворо для позначення мереж, що складаються з декількох шарів персептронів (з пороговою активацією). Багатошарові персептрони іноді називають "ванільними" нейромережами, особливо коли вони мають один прихований шар.

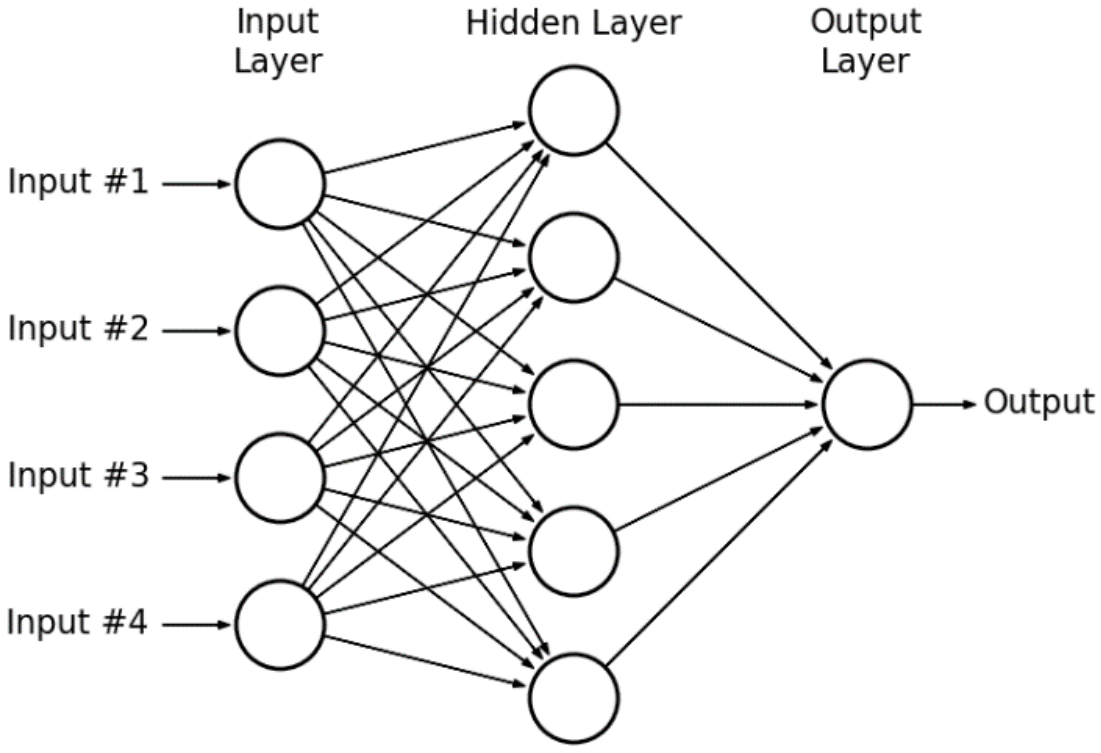

Рисунок 2.4 – Багатошаровий персептрон

БШП складається з щонайменше трьох шарів вузлів: вхідного шару, прихованого шару та вихідного шару. За винятком вхідних вузлів, кожен вузол є нейроном, який використовує функцію нелінійної активації. БШП використовує методику контрольованого навчання, яку називають навчання зі зворотним поширенням. Його багатошарові шари та нелінійна активація відрізняють БШП

від лінійного персептрона. Він може виділити дані, які не є лінійно відокремленими.

У мережі на рис. 2.4 перша колонка персептронів – те, що називається першим шаром персептронів – приймає чотири дуже простих рішення, зважуючи вхідні дані. Кожен із персептронів другого приймає рішення, зважуючи результати з першого рівня прийняття рішень. Таким чином персептрон у другому шарі може приймати рішення на більш складному і більш абстрактному рівні, ніж персептрони в першому шарі. І ще більш складні рішення може приймати персептрон у третьому шарі. Таким чином, багатошарова мережа персептронів може брати участь у складних процесах прийняття рішень.

У мережі вище персептрони виглядають так, ніби вони мають кілька виходів. Насправді вони мають лише один. Кілька стрілок виводу є лише корисним способом вказівки на те, що вихід персептрона використовується як вхід для кількох інших персептронів. Це менш громіздко, ніж малювання однієї вихідної лінії, яка потім розпадається.

Можна спростити спосіб опису персептронів. Умова $\sum_j w_j x_j > \text{поріг}$ громіздка, і можна внести дві нотаційні зміни, щоб спростити її. Перша зміна полягає в тому, щоб записати $\sum_j w_j x_j$як крапковий добуток , $w \cdot x \equiv \sum_j w_j x_j$, де w та x – це вектори, компонентами яких є ваги та входи відповідно. Друга зміна полягає у переміщенні порогу на іншу сторону нерівності та заміщення його тим, що відоме як зсув персептрона, $b \equiv -\text{поріг}$. Використовуючи зміщення замість порогового значення, правило персептрона можна переписати:

$$output = \begin{cases} 0 \; if \;\; w \cdot x \leq \text{поріг} \\ 1 \; if \;\; w \cdot x > \text{поріг} \end{cases} \qquad (2.2)$$

Зміщення можна сприймати як міру того, як легко змусити персептрон вивести 1. Або, кажучи більш біологічно, зміщення є мірою того, як легко змусити персептрон спрацьовувати. Для персептрона з дійсно великим зміщенням персептрону надзвичайно легко вивести 1. Але якщо зміщення дуже негативне, то персептрону важко вивести 1. Очевидно, що введення зміщення – це лише невелика зміна при описі персептрона, але пізніше можна побачити, що це веде до подальших нотаційних спрощень.

2.3.1 Функція активації

Якщо багатошаровий персептрон має лінійну функцію, яка відображає зважені входи на вихід кожного нейрона, то лінійна алгебра показує, що будь-яка кількість шарів може бути зменшена до двошарової моделі вхід – вихід [38].

Для моделювання нелінійної проблеми вводять нелінійність. Можливо зв’язати кожен вузол прихованого шару через нелінійну функцію.

У моделі, представленій наступним графіком, значення кожного вузла в прихованому рівні 1 подається за допомогою нелінійної функції перед передачею до зважених ваг наступного шару. Ця нелінійна функція називається функцією активації.

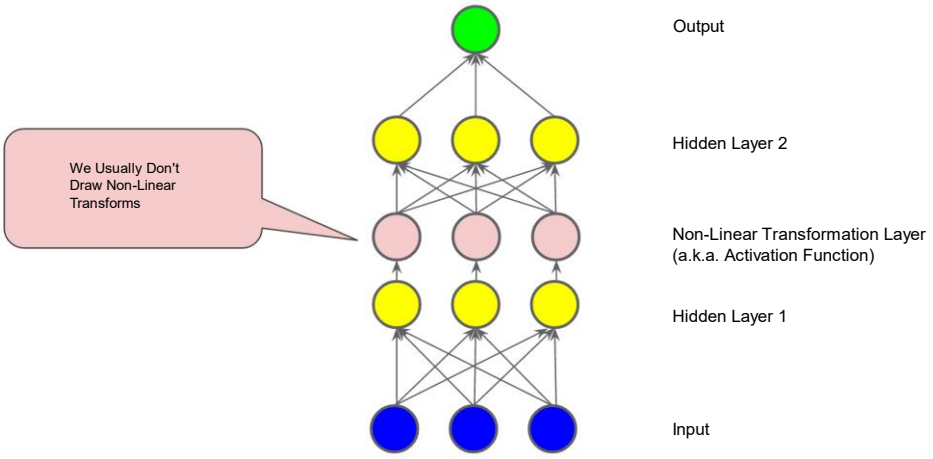

Рисунок 2.5 – Графік тришарової моделі з функцією активації.

Додавання шарів має більший вплив із використанням функції активації. Складання нелінійності на нелінійності дозволяє моделювати дуже складні взаємозв'язки між вхідними даними та прогнозованими результатами. Коротше кажучи, кожен рівень ефективно вивчає більш складну функцію вищого рівня над вихідними входами.

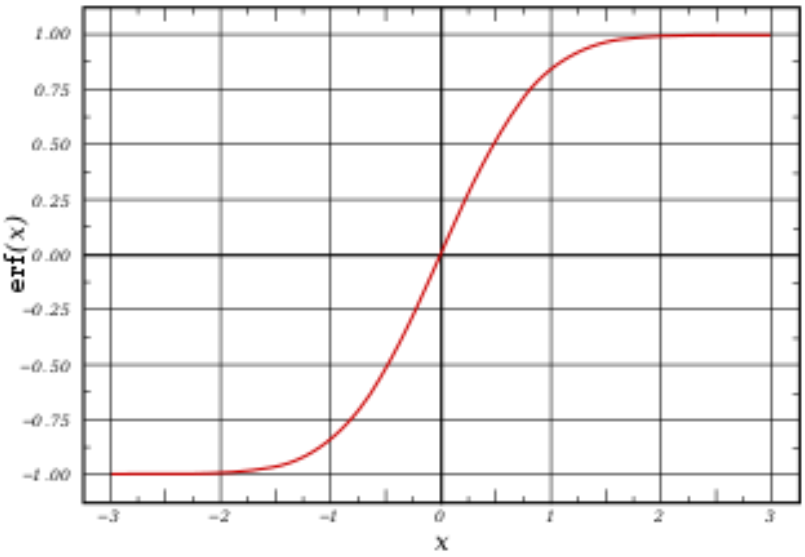

Рисунок 2.4 – Функція сигмоїди $y(v_i) = \tan v_i$

Дві історично поширені функції активації є обидві сигмоїдами і описуються ними

$$y(x) = \tan x \text{ та } y(x) = (1 + e^x)^{-1} \quad (2.1)$$

і (2.2)

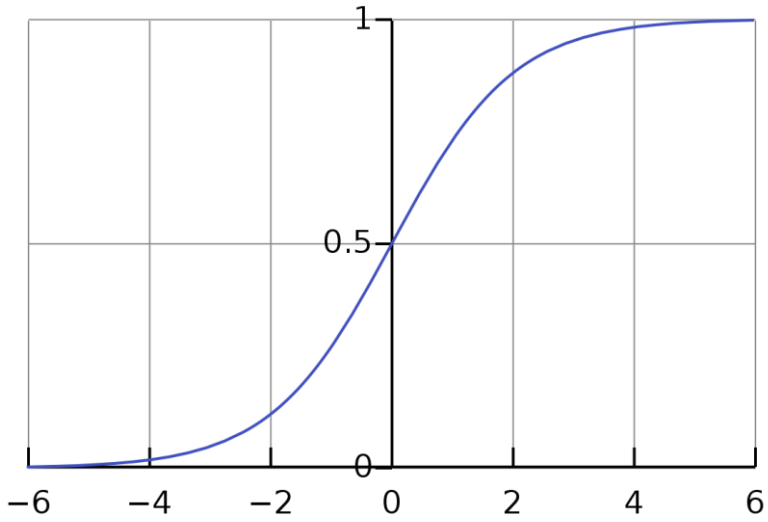

Рисунок 2.7 – Функція сигмоїди $y(x) = (1 + e^x)^{-1}$

Перший – це гіперболічна дотична, яка коливається від -1 до 1, а друга – логістична функція, схожа за формою, але коливається від 0 до 1. Тут y_i – вихід i-го вузла (нейрону) та v_i – зважена сума вхідних з'єднань.

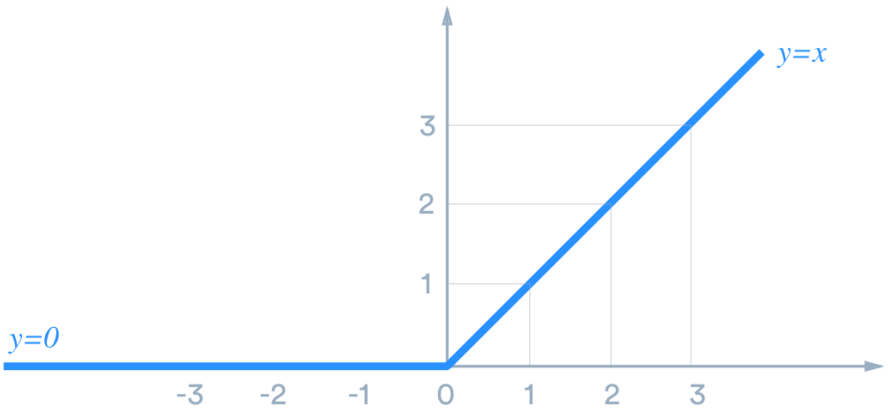

Рисунок 2.8 – Функція ReLU

В останніх розробках глибокого вивчення випрямляюча функція активації лінійної одиниці або ReLU (rectified linear unit) $y(x) = max(0, x)$ частіше використовується як один із можливих способів подолання чисельних проблем, пов'язаних із сигмоїдами.

Випрямляюча функція активації лінійної одиниці часто працює трохи краще, ніж така плавна функція, як сигмоїдна, в той же час є значно легшою для обчислення.

Перевага ReLU базується на емпіричних висновках, мабуть, зумовлених ReLU, який має більш корисний спектр реагування. Чутливість сигмоїди порівняно швидко падає з обох сторін

Запропоновані альтернативні функції активації, включаючи випрямляч і м’якопозитивну функцію. Більш спеціалізовані функції активації включають радіальні базові функції (використовуються в радіальних базових мережах, інший клас моделей нейронних мереж, що контролюються).

2.3.2 Навчання

Навчання здійснюється шляхом зміни з’єднувальної ваги після обробки кожної частини даних, виходячи з кількості помилок на виході порівняно з очікуваним результатом. Це приклад контрольованого навчання і здійснюється шляхом зворотного розповсюдження, узагальнення алгоритму найменших середніх квадратів у лінійному персептроні.

Ми можемо представити ступінь помилки у вихідному вузлі j у n-ій точці даних (з вибірки навчання) за допомогою $e_j(n) = d_j(n) - y_j(n)$, де d – цільове

значення, а y – значення, отримане персептроном. Ваги вузлів можуть бути потім відрегульовані на основі коректівів, що мінімізують похибку у всьому виході, задані

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2\,(n) \tag{2.3}$$

Використовуючи градієнтний спуск, зміна кожної ваги є

$$\Delta\omega_{ij}(n) = -\eta\frac{\delta\varepsilon(n)}{\delta v_i(n)}y_j(n) \tag{2.4}$$

Де y_j – вихід попереднього нейрона та η – швидкість навчання, яка вибирається для того, щоб ваги швидко переходили у відповідь без коливань.

Похідна для обчислення залежить від збудженого локального поля v_i, яке само по собі змінюється. Неважко довести, що для вихідного вузла цю похідну можна спростити

$$\frac{\delta\varepsilon(n)}{\delta v_i(n)} = e_j(n)\phi'(v_i(n)), \tag{2.5}$$

де ϕ' – похідна описаної вище функції активації, яка сама по собі не змінюється. Аналіз є складнішим для зміни у вагах на прихованому вузлу, але можна показати, що відповідна похідна є

$$-\frac{\delta\varepsilon(n)}{\delta v_i(n)} = \phi'(v_i(n))\sum_k -\frac{\delta\varepsilon(n)}{\delta v_k(n)}\omega_{kj}(n) \tag{2.6}$$

Це залежить від зміни ваг k-их вузлів, які представляють вихідний шар. Отже, щоб змінити приховані ваги шару, ваги вихідного шару змінюються відповідно до похідної функції активації, і тому цей алгоритм являє собою зворотне розповсюдження функції активації.

2.4 Отримана модель нейронної мережі

Модель запропонованої нейромережі є персептроном, на вхід якого подається вектор $\bar{X}$ ($\bar{X} = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\}$), на виході отримуємо вектор $\bar{Y}$ ($\bar{Y} = \{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}\}$). Модель можна побачити на рис. 2.9.

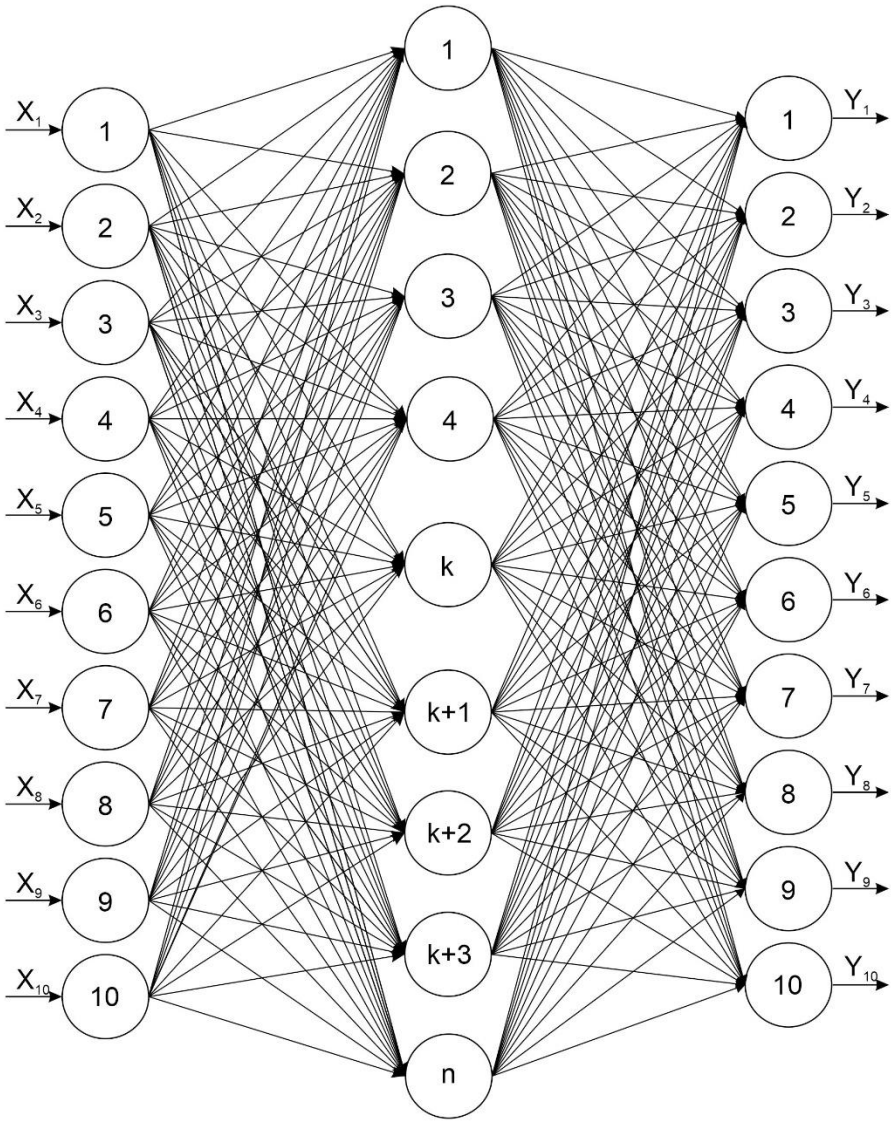

Рисунок 2.9 – Схема нейронної мережі

Таким чином маємо структуру БШП: вхідний шар, який має 10 вузлів, прихований із n кількістю нейронів та вихідний із 10 вузлів. Схема зв’язку вузлів наступна: кожен зв’язаний із сусіднім, тобто кожен сингал нейрону першого шару зважується у кожному нейроні прихованого шару, вихідний сигнал яких зважується у кожному нейроні вихідного шару. На вхід БШП подається шифроване повідомлення у вигляді 10 бітів. На виході отримуємо 10 бітів, що уявляє собою відкрите повідомлення. За допомогою даної моделі виконуємо криптоаналіз алгоритму.

Для цієї цілі використана бібліотека ml5 для мови JavaScript, що дозволяє створювати багатошарові персептрони, також відомі як ШНМ з прямим

розповсюдженням [39]. Вони складаються з послідовності шарів, кожен повністю пов'язаний з наступним.

ml5.js має на меті зробити машинне навчання доступним для широкої аудиторії художників, творчих програмістів та студентів. Бібліотека забезпечує доступ до алгоритмів та моделей машинного навчання у браузері, будуючи поверх TensorFlow.js.

3 ОПИС ЕКСПЕРИМЕНТІВ І ПРОГРАМ

3.1 Суть експерименту

Результати експерименту, отриманні у роботі [40] мали зменшену модель, де ми мали обмеження у 8 бітів, загальна навчальна вибірка становила 256 зашифрованих повідомлень, включаючи нуль. У якості ключів були обрані випадкові 8-бітові значення, і не виявлено кореляції між значенням ключа та кількістю помилок. Ключове значення не змінювалось від початку навчання до виконання всіх експериментів. Той самий ключ використовувався у всіх раундах шифрування. Тестування проводили з використанням 3, 5, 7 та 10 раундів. Спочатку експерименти проводили на 70% повної виборки.

Результати експериментів із використанням 70% повної вибірки для вивчення НМ наведені в таблиці 3.1

Таблиця 3.1 – Результати комп'ютерних досліджень на 70% від повної вибірки

N	**Кількість помилок (з 256 експериментів)**				**Час тренувань, сек.**
	3 раунди	***5 раундів***	***7 раундів***	***10 раундів***	
32	80	104	136	157	91.7
48	74	76	77	89	285.1
56	75	77	76	76	339.9
64	76	77	80	74	373.6
72	75	76	76	82	364.4
80	72	76	76	76	589.3
96	71	76	77	76	547.8
112	76	76	75	77	680
120	75	76	76	76	690.6
128	72	76	76	75	738.1
256	74	76	76	74	1396.5
512	71	77	78	86	3609.5

Перший стовпець позначається N і означає кількість нейронів у прихованому шарі. Отже, кількість нейронів за шарами становить 8-N-8. Стовпці від двох до п’яти вказують кількість помилок (з 256 можливих), які нейронна мережа робить, використовуючи 3, 5, 7 та 10 раундів шифрування. Останній, шостий стовпець, вказує середній час у секундах на навчання нейронної мережі.

Експерименти демонструють кореляцію між кількістю нейронів у прихованому шарі нейронної мережі та отриманням правильного відкритого тексту. Видно, що найменший відсоток помилок можна отримати при використанні трирівневої нейронної мережі з кількістю нейронів у внутрішньому прихованому рівні більше 48. Найкраща кількість нейронів у прихованому шарі – 96 нейронів з часом тренування 547,8 секунди ≈ 9 хвилин. Подальше збільшення кількості нейронів не дає позитивного ефекту, а лише збільшує час тренувань.

Експерименти для інших розмірів навчальної вибірки проводили для 96 нейронів у прихованому шарі, а результати представлені в таблицях 3.2, 3.3 та 3.4.

Таблиця 3.2 – Результати обчислювальних досліджень для 80% від повної вибірки з 96 нейронами в прихованому шарі

Кількість раундів	**Кількість помилок**	**Середнє число бітових помилок**	**Час тренувань, сек.**
3	49	2.9	699
5	52	3.9	567.4
7	53	4	555.2
10	53	3.9	29430

Таблиця 3.3 – Результати обчислювальних досліджень для 90% від повної вибірки з 96 нейронами в прихованому шарі

Кількість раундів	**Кількість помилок**	**Середнє число бітових помилок**	**Час тренувань, сек.**
3	26	3	656.2
5	37	3.1	644.5
7	40	3.5	634.6
10	30	3.6	633.2

Таблиця 3.4 – Результати обчислювальних досліджень для повної вибірки з 96 нейронами в прихованому шарі

Кількість раундів	**Кількість помилок**	**Середнє число бітових помилок**	**Час тренувань, сек.**
3	1	5	759
5	29	1.1	762.1
7	67	1.4	753.2
10	24	1.3	716.3

Можна бачити, що із зростанням вибірки навчань загальна кількість помилок і кількість бітових помилок зменшуються. При цьому час тренувань дещо збільшується.

Більш чітко видно залежність кількості помилок від кількості раундів шифрування. За меншу кількість раундів помилок завжди стає менше.

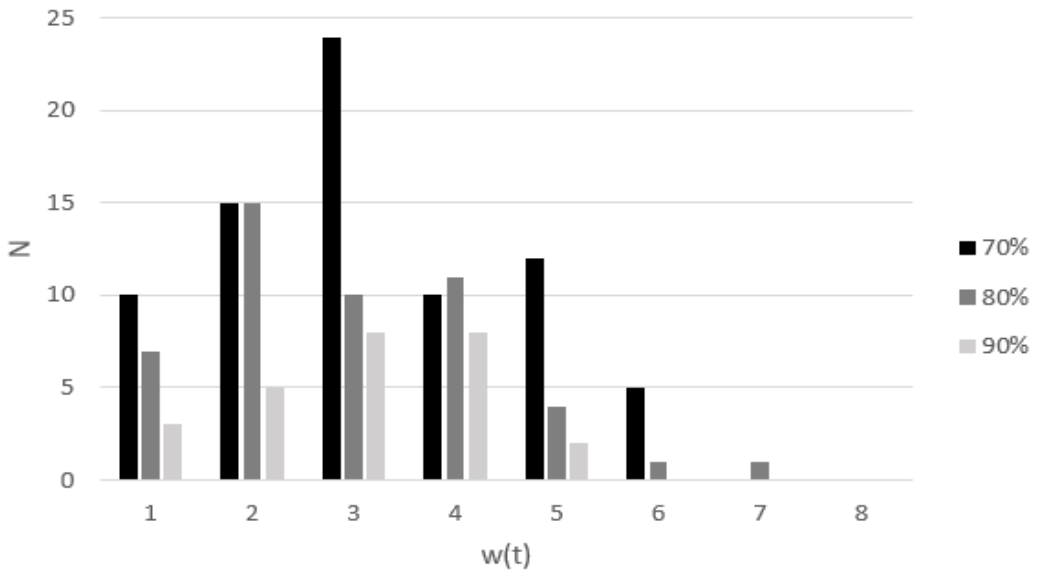

Рисунок 3.1 – Діаграма кореляції ваги Хеммінга та кількості помилок з використанням 70%, 80%, 90% і повної вибірки за 3 раунди.

На діаграмі ви можете бачити кореляцію ваги Хеммінга (w (t)) та кількості помилок (N), використовуючи 70%, 80%, 90% повної вибірки за 3 раунди.

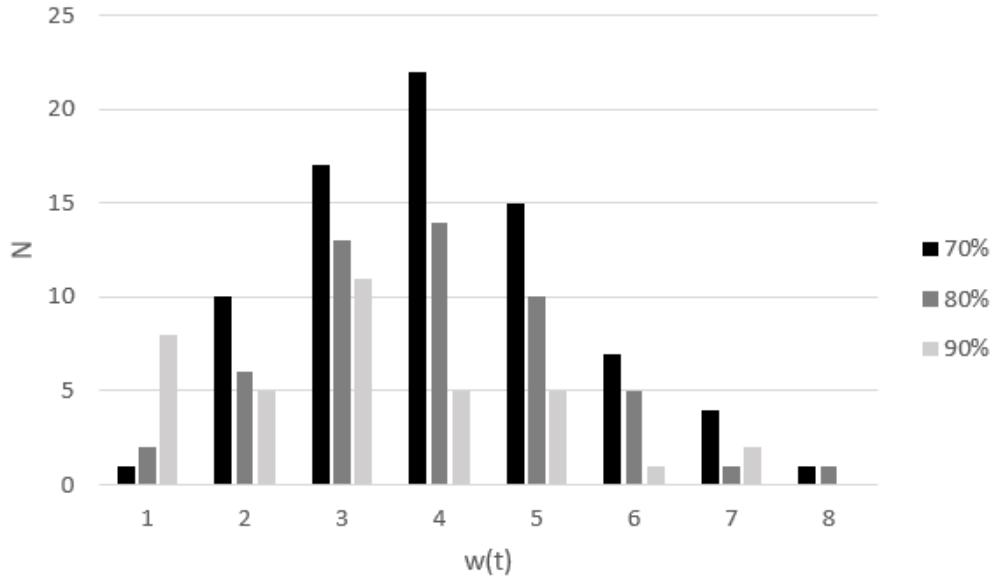

Рисунок 3.2 – Діаграма кореляції ваги Хеммінга та кількості помилок з використанням 70%, 80%, 90% і повної вибірки за 5 раундів.

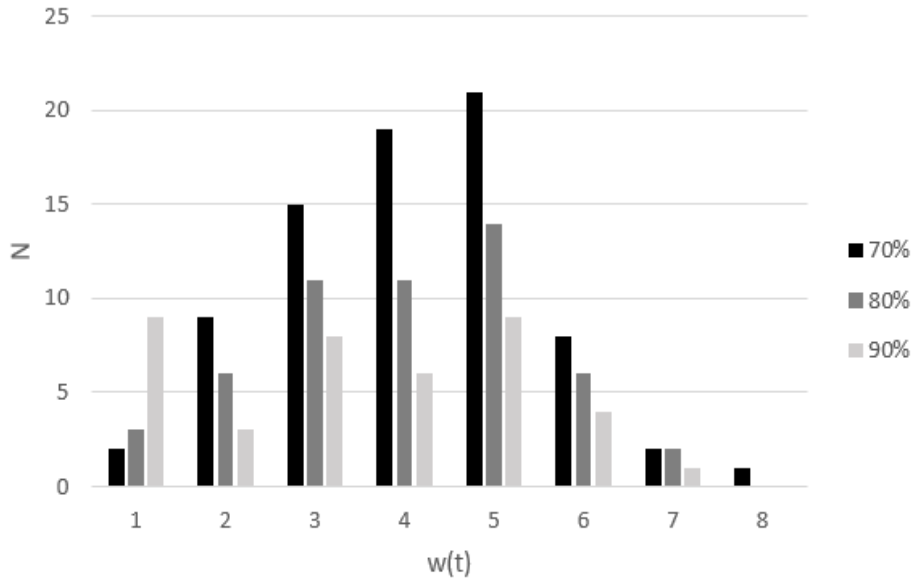

Рисунок 3.3 – Діаграма кореляції ваги Хеммінга та кількості помилок з використанням 70%, 80%, 90% і повної вибірки за 7 раундів.

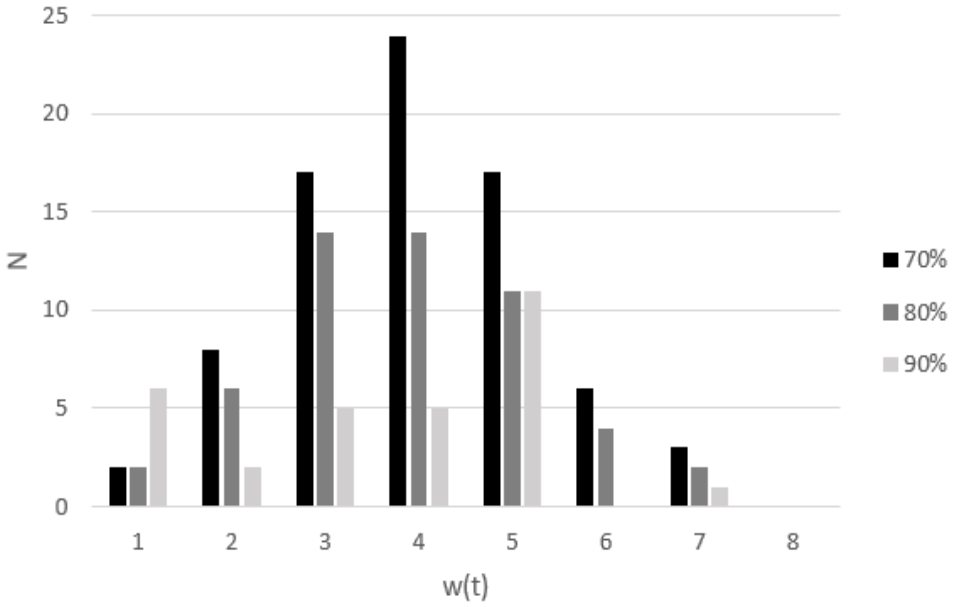

Рисунок 3.4 – Діаграма кореляції ваги Хеммінга та кількості помилок з використанням 70%, 80%, 90% і повної вибірки за 7 раундів.

Робота використовувала бібліотеку Synaptic2.0 , яка мала певні обмеження.

Таблиця 3.5 – Результати обчислювальних досліджень для повної вибірки з шагом у 256 нейронів у прихованому шарі

Раундів	n	Відсоток помилок	Середня помилка
1	256	0%	0
	512	0%	0
	768	0%	0
	1024	0%	0
2	256	35%	1,9
	512	12%	3,66
	768	11%	3,1
	1024	11%	3,34
3	256	57%	2,086
	512	22%	2,95
	768	11%	4,52
	1024	12%	4,57
5	256	73%	2,54
	512	13%	4,32
	768	13%	4,42
	1024	12%	4,49
7	256	72%	2,21
	512	27%	2,94
	768	11%	4,99
	1024	11%	4,68
9	256	70%	2,26
	512	20%	3,23
	768	11%	5,06
	1024	13%	4,42
11	256	62%	2,15
	512	14%	4,14
	768	11%	4,89
	1024	11%	4,94

Продовженням роботи була протестована розширена модель із 10 бітів на вході НМ, та навчальною вибіркою в 1024 шифрованих повідомлень і відповідних їм простому тексті. У 11 раундах із константним ключем експерименти також вказали на взаємозв’язок необхідності збільшення кількості нейронів у прихованому шарі та кількості циклів шифрування. Результати експериментальних даних таблиці 3.5 вказують на то, що навіть на повній вибірці ми бачимо дані, які підтверджують, що кількість нейронів у прихованому шарі впливає на покращення результатів нейронної мережі, а збільшення кількості

нейронів з 768 до 1024 покращує результати тільки на декілька відсотків, залишаючи велику кількість помилок. Це наглядно видно на діаграмі на рис. 3.5

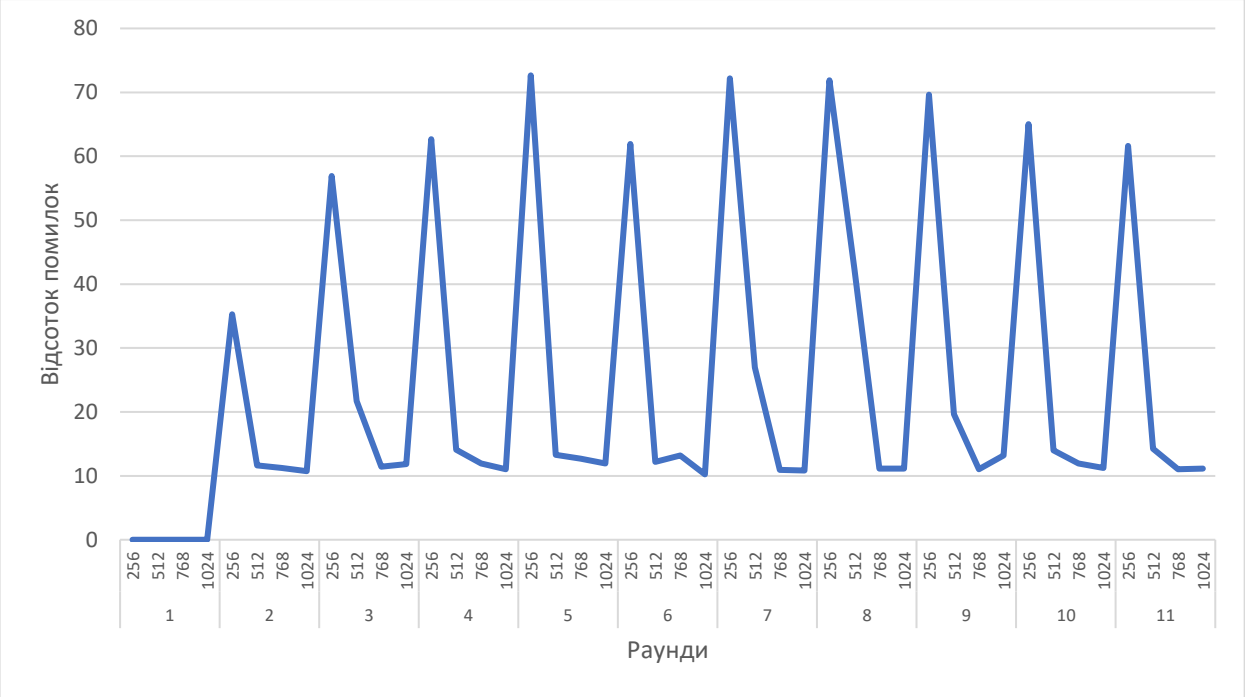

Рисунок 3.5 – Діаграма взаємозв’язку кількості нейронів у прихованому шарі та відсотком помилок

З іншого боку, якщо ми поглянемо на кореляцію між відсотком помилок і середньою величиною помилки у 10 бітах (рис. 3.6), ми приходимо до очевидного висновку, що чим менше значення помилок, тим більше середнє значення помилки серед 10 бітів і навпаки.

Таблиця 3.6 – Результати обчислювальних досліджень для вибірки 70% з шагом у 256 нейронів у прихованому шарі для п’яти раундів.

Раундів	N	Кількість помилок (серед 1024 можливих)	Середня помилка
1	256	0	0
	512	0	0
	768	0	0
	1024	0	0
2	256	525	3,48
	512	383	4,46
	768	392	3,96
	1024	375	4,08
3	256	583	2,09
	512	385	5,10
	768	389	4,87
	1024	382	5,05
4	256	592	3,59
	512	408	4,99
	768	410	5,14
	1024	382	5,04
5	256	649	3,73
	512	385	5,17
	768	384	5,25
	1024	381	5,08

Тепер розглянемо результати з табл. 3.6. Експеримент проводився для 70% від загальної вибірки. Незважаючи на це, для одного раунду нейронна мережа показує 0% помилок результатів навіть з 256 бітами у прихованому шарі. Але починаючи з другого раунду, кількість помилок зростає, як і значення середньої помилки. А різниця у кількісті помилок із моделями з 512 і 1024 нейронів у прихованому шарі складає лише декілька значень або максимум 1% для усіх раундів.

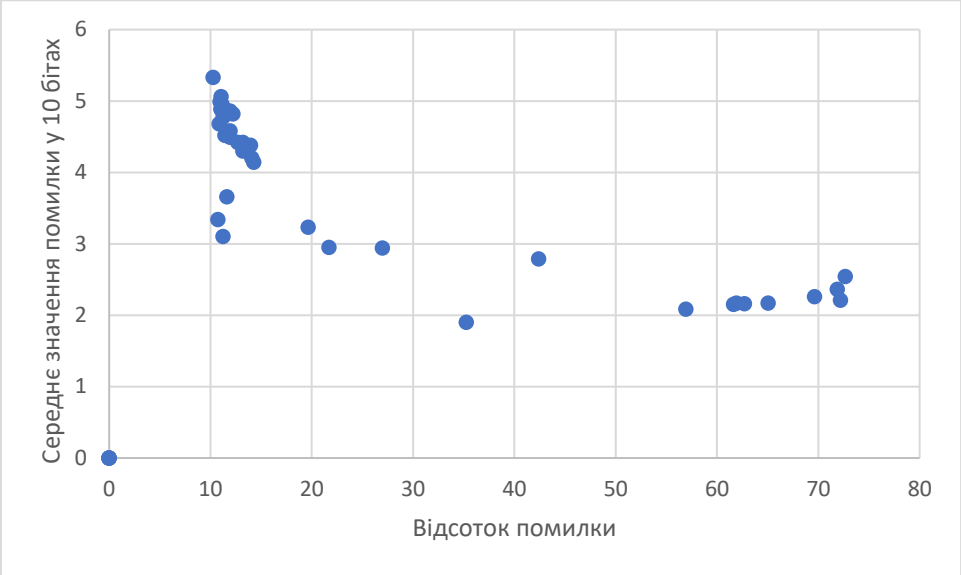

Рисунок 3.6 – Кореляція відсотка помилок і середньої помилки у 10 бітах

У моделі 8-96-8 використана була логістична функції активації, тоді як для 10 бітного блоку шифрування – ReLu, що вплинуло на точність результатів. Проте помилка у моделі з 8 вхідними вузлами на вибірці 70% давала 27,7% помилок, в моделі в 10 вхідними вузлами, ми отримуємо 37%.

З усіх даних можливо зробити декілька висновків.

По-перше, криптоаналіз за допомогою нейронних мереж для слабких шифрів типу заміщення-перестановки можливий, але потребує подальшого розгляду та аналізу.

По-друге, якщо навчитися обробляти та корегувати помилку на рівні бітів, то можна досягти швидкого навчання для блоку у 10 бітів із використанням лише 256 бітів у прихованому шарі.

По-третє, найменший відсоток помилок можна отримати при використанні тришарової повністю підключеної нейронної мережі з достатньою кількістю нейронів у внутрішньому прихованому шарі.

У той же час розглянута модель не показала явної залежності складності криптоаналізу з використанням нейронних мереж від кількості раундів при перетворенні шифрування.

Питання про оптимальність обраної архітектури нейронних мереж залишалося відкритим. Для пошуку відповіді на це питання необхідно

розглянути більше варіантів нейронних мереж та перетворень шифрування. Це завдання може бути предметом майбутніх досліджень.

3.2 Загальні відомості програмної реалізації

Для написання курсової роботи було реалізовані 3 файли: index.html файл, який підключає усі потрібні залежності та містить результати у вигляді посилань на файли з результатами роботи; 10_10.js, що містить увесь потрібний код для створення різних моделей нейронної мережі для різних раундів шифру, їх навчання, тестування; 10_bit_nth_round_cypher.js – має усі необхідні для шифрування функції та змінні.

Усі компоненти написані на мові програмування JavaScript з використанням бібліотеки ml5 у середі розробки Visual Studio Code версії 1.49.3 та запуску на Google Chrome версії 83.0.4103.97. Для їх запуску необхідно мати будь-який браузер, не пізніше 2004 року. Програма тестувалася на ПК з ОС 64 розрядної Windows 10, процессор Intel® Core™ i5-7200 CPU @2.5 GHz 2.7 GHz, 8 ГБ ОЗП.

3.3 Функціональне призначення та обмеження

Застосування як цільного продукту не рекомендоване через необхідність в доробці та збільшенні кількості тестів та варіативності, можливо інтерфейсі для користувача та презентабельного вигляду. Даний програмний код не є повноцінним продуктом і є лише наглядним засобом демонстрації моєї роботи.

3.4 Опис логічної структури

В програмах реалізовано:

Тестування нейронної мережі та виводу даних

Функція для ініціювання, навчанні та обробка даних

Алгоритм шифрування

Код розбито на файли наступним чином:

test.html – файл, який підключає усі потрібні залежності та містить результати у вигляді посилань на файли з результатами роботи;

10_10.js – файл, що містить увесь потрібний код для створення різних моделей нейронної мережі для різних раундів шифру, їх навчання, тестування;

10_bit_nth_round_cypher.js – – файл, який має усі необхідні для шифрування функції та змінні.

Опис основних функцій, що використовуються.

start(rounds) – генерує початкові параметри та викликає усі функції;

reverseBoxes() – створює зворотні s-boxи;

shuffleArray(array) – замішує данні;

randomInteger(min, max) – генерує число у діапазоні від min до max;

nth_round_encryption(rounds) – виконує шифрування для rounds кількості раундів;

decryption(rounds) – виконує дешифрування для rounds кількості раундів і перевіряє початкове повідомлення з дешифрованим;

3.5 Опис створення нейронної мережі за допомогою ml5

За допомогою ml5.neuralNetwork можливо створити свою власну нейронну мережу та навчити її у браузері. Для цього потрібно зібрати дані для тренування нейронної мережі або використовуйте наявні дані для тренування нейронної мережі в режимі реального часу. Після навчання нейронна мережа може виконувати завдання класифікації або регресії.

Крок 1: завантажуємо дані або створюємо деякі дані

```
for (let i = 0; i < mes.length; i++) {
                data[i] = new Object();
                data[i].m = mes[i];
                data[i].c = perm[i];
        }
```

Крок 2: встановлюємо параметри нейронної мережі

```
const options = {
                task: "regression",
                learningRate: 0.0009765625,
```

```
        debug: true,
        layers: [{
                type: "dense",
                units: 10,
                activation: "relu",
        },
        {
                type: "dense",
                units: hidden,
                activation: "relu",
        },
        {
                type: "dense",
                units: 10,
                activation: "relu",
        },
        ],
};
```

Крок 3: ініціалізація нейронної мережі

```
model = ml5.neuralNetwork(options);
```

Крок 4: додайте дані до нейронної мережі

```
for (let i = 0; i < mes.length; i++) {
        const inputs = {
                x1: (data[i].c >> 9) & 1,
                x2: (data[i].c >> 8) & 1,
                x3: (data[i].c >> 7) & 1,
                x4: (data[i].c >> 6) & 1,
                x5: (data[i].c >> 5) & 1,
                x6: (data[i].c >> 4) & 1,
                x7: (data[i].c >> 3) & 1,
                x8: (data[i].c >> 2) & 1,
                x9: (data[i].c >> 1) & 1,
                x10: data[i].c & 1,
        };

        const outputs = {
                y1: (data[i].m >> 9) & 1,
                y2: (data[i].m >> 8) & 1,
                y3: (data[i].m >> 7) & 1,
                y4: (data[i].m >> 6) & 1,
                y5: (data[i].m >> 5) & 1,
                y6: (data[i].m >> 4) & 1,
                y7: (data[i].m >> 3) & 1,
                y8: (data[i].m >> 2) & 1,
                y9: (data[i].m >> 1) & 1,
```

```
                y10: data[i].m & 1,
            };

            model.addData(inputs, outputs);
        }
```

Крок 5: нормалізуйте свої дані;

```
model.normalizeData();
```

Крок 6: Тренуйте свою нейронну мережу

```
const trainingOptions = {
            epochs: 1300,
            batchSize: 32,
        };
model.train(trainingOptions, finishedTraining);
```

batchSize відноситься до розміру підмножин даних, які модель побачить на кожній ітерації навчання. Загальні розміри партій, як правило, знаходяться в межах 32-512, але, насправді, не існує ідеального розміру партії для всіх проблем.

epochs – це кількість разів, коли модель збирається переглянути весь набір даних, який ви йому надаєте.

Крок 7: використовуйте навчену модель

```
function finishedTraining(){
    classify();
}
```

Крок 8: Виконуйте регресію

```
for (let i = 0; i < data.length; i++) {
            let input = {
                x1: (data[i].c >> 9) & 1,
                x2: (data[i].c >> 8) & 1,
                x3: (data[i].c >> 7) & 1,
                x4: (data[i].c >> 6) & 1,
                x5: (data[i].c >> 5) & 1,
                x6: (data[i].c >> 4) & 1,
                x7: (data[i].c >> 3) & 1,
                x8: (data[i].c >> 2) & 1,
                x9: (data[i].c >> 1) & 1,
                x10: data[i].c & 1,
            };
```

```
        model.predict(input, handleResults);
    }
```

Крок 9: Визначте функцію для обробки результатів вашої регресії

```
function handleResults(error, result) {
    if (error) {
        console.error(error);
        return;
    }

    let a = 0;
    for (let i = 0; i < 10; i++) {
        a |= Math.round(result[i].value) << (9 - i);
    }

    //checking for 10 bits

    if (a === data[counter].m) success++;
    else {
        let er = 0;
        for (let i = 0; i < 10; i++) {
            if (Math.round(result[i].value) !== ((data[counter].m >> (9 -
i)) & 1))
                er++;
        }

        meanErr.push(er);

        let n = [];
        let cyph = [];
        for (let i = 0; i < 10; i++) {
            n.push(Math.round(result[i].value));
            cyph.push((data[counter].m >> (9 - i)) & 1);
        }
    }

    counter++;
}
```

3.6 Технічні засоби, що використовувалися

Учбова версія програми не накладає обмежень на технічні засоби, що використовуються. Технічним вимогам програми задовольняє ПК з мінімальними характеристиками. Найбільшу обчислювальну складність мають

процедура навчання нейронної мережі, час їх виконання можна побачити у таблиці 4.

3.7 Вхідні та вихідні дані

Вхідні данні генеруються або задані всередині програмного коду.

Вихідні дані – 44 файлів, які містять данні із роботи нейронної мережі за 11 раундів розшифрування із кількістю нейронів 256, 512, 768, 1024 для кожного раунду.

ВИСНОВКИ

У роботі було розглянуто можливість навчання нейронних мереж розшифруванню зашифрованих повідомлень за допомогою пар шифротекст – повідомлення при фіксованому секретному ключі. Досліджено можливість використання багатошарової архітектури персептронів нейронної мережі для атаки експериментального 8 і 10-бітового шифру заміщення-перестановки.

Для розглянутих конструкцій, одна з яких складається з двох 4-бітових S- блоків, а інша з двох 5-бітових S-блоків, додавання XOR з ключем та циклічного зсуву, кількість помилок найбільше залежить від розміру навчальної вибірки.

Для поставлених цілей була використана модель персептрона. Були протестовані різні структури, які відрізнялися кількістю нейронів у прихованому шарі. Для 8-бітного блоку оптимальна модель НМ склала 96 нейронів. Для 10-бітного блоку – 768. Збільшення кількості вузлів, збільшення кількості прихованих шарів у середньому не покращувало отримані результати, хоча в свою чергу вимагало більшу обчислювальну потужність і часу навчання НМ.

Таким чином, найменший відсоток помилок можна отримати при використанні тришарової повнозв'язної нейронної мережі з достатньою кількістю нейронів у внутрішньому прихованому шарі.

Вибрана бібліотека ml5 показала більшу зручність, а також більшу кількість налаштувань для отримання результатів. Обрана випрямляюча функція активації у кожному шарі покращила точність значення вихідних значень.

Таким чином, криптоаналіз за допомогою нейронних мереж для слабких шифрів типу заміщення-перестановки є перспективним. Але для застосування у більш складних шифрах потребує подальшого розгляду та аналізу.

Питання про оптимальність обраної архітектури нейронних мереж залишається відкритим. Для пошуку відповіді на це питання необхідно розглянути більше варіантів нейронних мереж та перетворень шифрування. Це завдання може бути предметом майбутніх досліджень.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. Aspects of Neural Cryptography Alex Moltzau, https://medium.com/@alexmoltzau/neural-cryptography-31f4a204206
2. стаття «Нейронні мережі – шлях до глибинного навчання», https://codeguida.com/post/739.
3. стаття «Штучні нейронні мережі: що це таке?», https://futurum.today/shtuchni-neironni-merezhi-shcho-tse-take/
4. Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (Third ed.). Prentice Hall. ISBN 9780136042594.
5. Mohri, Mehryar; Rostamizadeh, Afshin; Talwalkar, Ameet (2012). Foundations of Machine Learning. The MIT Press. ISBN 9780262018258.
6. Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2. ISBN 978-0-07-042807-2.
7. Alpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press. p. 9. ISBN 978-0-262-01243-0.
8. Jordan, Michael I.; Bishop, Christopher M. (2004). "Neural Networks". In Allen B. Tucker (ed.). Computer Science Handbook, Second Edition (Section VII: Intelligent Systems). Boca Raton, Florida: Chapman & Hall/CRC Press LLC. ISBN 978-1-58488-360-9.
9. Alex Ratner; Stephen Bach; Paroma Varma; Chris. "Weak Supervision: The New Programming Paradigm for Machine Learning". hazyresearch.github.io. referencing work by many other members of Hazy Research. Retrieved 2019-06-06.
10. van Otterlo, M.; Wiering, M. (2012). Reinforcement learning and markov decision processes. Reinforcement Learning. Adaptation, Learning, and Optimization. 12. pp. 3–42. doi:10.1007/978-3-642-27645-3_1. ISBN 978-3-642-27644-6.
11. Bozinovski, S. (1982). "A self-learning system using secondary reinforcement". In Trappl, Robert (ed.). Cybernetics and Systems Research: Proceedings of the Sixth European Meeting on Cybernetics and Systems Research. North Holland. pp. 397–402. ISBN 978-0-444-86488-8.

12. Bozinovski, S. (2001) "Self-learning agents: A connectionist theory of emotion based on crossbar value judgment." Cybernetics and Systems 32(6) 637-667.

13. Machine Learning For Beginners by Divyansh Dwivedi, https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab

14. Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. ISSN 0893-6080. PMID 25462637.

15. Broomhead, D. S.; Lowe, David (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks (Technical report). RSRE. 4148.

16. Werbos, P. J. (1988). "Generalization of backpropagation with application to a recurrent gas market model". Neural Networks. 1 (4): 339–356. doi:10.1016/0893-6080(88)90007-x.

17. David E. Rumelhart; Geoffrey E. Hinton; Ronald J. Williams. Learning Internal Representations by Error Propagation.

18. A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

19. Schmidhuber, J. (1989). "A local learning algorithm for dynamic feedforward and recurrent networks". Connection Science. 1 (4): 403–412. doi:10.1080/09540098908915650. S2CID 18721007.

20. Schmidhuber, J. (1992). "A fixed size storage O(n3) time complexity learning algorithm for fully recurrent continually running networks". Neural Computation. 4 (2): 243–248. doi:10.1162/neco.1992.4.2.243. S2CID 11761172.

21. Pearlmutter, B. A. (1989). "Learning state space trajectories in recurrent neural networks" (PDF). Neural Computation. 1 (2): 263–269. doi:10.1162/neco.1989.1.2.263. S2CID 16813485.

22. S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, 1991.

23. Neural Networks as Cybernetic Systems 2nd and revised edition, Holk Cruse

24. Schrauwen, Benjamin, David Verstraeten, and Jan Van Campenhout "An overview of reservoir computing: theory, applications, and implementations." Proceedings of the European Symposium on Artificial Neural Networks ESANN 2007, pp. 471–482.

25. Hochreiter, S.; Schmidhuber, J. (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.

26. F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages IEEE Transactions on Neural Networks 12(6):1333–1340, 2001.

27. A. Graves, J. Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. Advances in Neural Information Processing Systems 22, NIPS'22, p 545-552, Vancouver, MIT Press, 2009.

28. Schuster, Mike; Paliwal, Kuldip K. (1997). "Bidirectional recurrent neural networks". IEEE Transactions on Signal Processing. 45 (11): 2673–2681. Bibcode:1997ITSP...45.2673S. CiteSeerX 10.1.1.331.9441. doi:10.1109/78.650093.

29. HAYKIN, S. Neural Networks - A Comprehensive Foundation. Second edition. Pearson Prentice Hall: 1999.

30. "Associative Neural Network". www.vcclab.org. Retrieved 2017-06-17.

31. Anderson, James A.; Rosenfeld, Edward (2000). Talking Nets: An Oral History of Neural Networks. ISBN 9780262511117.

32. B. Chandra and P. P. Varghese, “Applications of cascade correlation neural networks for cipher system identification,” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 1, no. 2, pp. 369 – 372, 2007. [Online]. Available: http://waset.org/Publications?p=2

33. M. M. Alani, “Neuro-cryptanalysis of des,” in World Congress on Internet Security (WorldCIS-2012), June 2012, pp. 23–27.

34. A. M. B. Albassal and A. M. A. Wahdan, “Neural network based cryptanalysis of a feistel type block cipher,” in Electrical, Electronic and Computer Engineering, 2004. ICEEC ’04. 2004 International Conference on, Sept 2004, pp. 231–237.

35. Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386—408.

36. free online book “Neural Networks and Deep Learning” by Michael Nielsen, http://neuralnetworksanddeeplearning.com/chap1.html

37. Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2009.

38. Haykin, Simon (1998). Neural Networks: A Comprehensive Foundation (2 ed.). Prentice Hall. ISBN 0-13-273350-1.

39. Гітхаб https://github.com/ml5js/, сайт http://ml5js.org

40. Cryptanalysis of simple Substitution-Permutation Cipher using Artificial Neural Network by Victor Ruzhentsev, Roman Levchenko, Oleksandr Fediushyn