Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

(інститут)

Факультет інформаційних технологій

(факультет)

Кафедра системного аналізу та управління

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеню магістра

(бакалавра, магістра)

студента Якуніної Анастасії Євгенівни

(ПІБ)

академічної групи 124м-21-1

(шифр)

спеціальності 124 - Системний аналіз

(код і назва спеціальності)

на тему «*Застосування нейронних мереж в задачі прогнозування індексу Доу Джонса»*

(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	Ус. С.А.			
розділів:				
Інформаційно-аналітичний	Ус. С.А.			
Спеціальній	Ус. С.А.			
Рецензент				
Нормоконтролер	Хом'як Т.В.			

Дніпро

2022

ЗАТВЕРДЖЕНО:

завідувач кафедри

Системного аналізу та управління
(повна назва)

_____ Желдак Т.А.
(підпис) (прізвище,ініціали)

«_____»_____20___року

Завдання

на кваліфікаційну роботу

ступеня магістра
(бакалавра, магістра)

студенту Якуніної А .Є. **академічної групи** 124м-21-1
(прізвище та ініціали) (шифр)

спеціальності 124 - Системний аналіз

на тему «*Застосування нейронних мереж в задачі прогнозування індексу Доу Джонса* »

Затверджена наказом ректора НТУ «Дніпровська політехніка» від 31.10.2022р .№ 1200-с

Розділ	**Зміст**	**Термін виконання**
Інформаційно-аналітичний	Виявити основні теоретичні передумови формування індексу Доу Джонса; Дослідити алгоритм нейронних мереж та різних методів прогнозування числових рядів.	
Спеціальний	Складання програмного апарату для розв'язання задачі прогнозування індуксу Доу Джонса із застосуванням нейронних мереж	

Завдання видано Ус. С.А.
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Якуніна А.Є.
(підпис студента)
(прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85с., 23 рис., 2 табл., 7 додатків, 30 джерел.

Об’єкт дослідження: процес функціювання ринку акцій.

Предмет дослідження: методи прогнозування індексу Доу Джонса прогнозування за допомогою нейронних мереж.

Мета дослідження: дослідження можливості застосування нейронної мережі до прогнозування Індекса Доу Джонса.

Економічна ефективність: очікується позитивною завдяки розробці програмного забезпечення, яке дозволяє на основі початкових даних про значення індексу за деякий період автоматично получити майбутні значення, і таким чином визначити позицію інвестора щодо придбання акцій тих чи інших компаній.

В інформаційно-аналітичному розділі наведена основна інформація про індекс Доу Джонса, процес формування алгоритму для побудови нейронної мережі, актуальність задачі прогнозування майбутнього значення індексу, та методи вирішення цієї задачі.

У спеціальному розділі виконано реалізацію побудови алгоритму машинного навчання.

Практична цінність роботи полягає в отриманні знань щодо майбутніх значень показника інвестування, чи треба покупати або продавати цінні папери .

Ключові слова: ШНДЕКС ДОУ ДЖОНСА, ЦІННІ ПАПЕРИ, ПРОГНОЗУВАННЯ, АНАЛІЗ, НЕЙРОННА МЕРЕЖА, БАГАТОШАРОВИЙ ПЕРСЕПТРОН, ТРЕНД, НЕЙРОН, МЕТОДИ ПРОГНОЗУВАННЯ, ЧАСОВИЙ РЯД.

ABSTRACT

Explanatory note: 85 p., 23 figures, 2 tables, 7 appendices, 30 sources.

The object of research: the process of functioning of the stock market.

Research subject: forecasting methods of the Dow Jones index forecasting using neural networks.

The purpose of the study: research on the possibility of applying a neural network to forecasting the Dow Jones Index.

Cost effectiveness: expected to be positive due to development

software that allows you to automatically obtain future values based on the initial data on the value of the index for a certain period, and thus determine the position of the investor regarding the purchase of shares of certain companies.

The informational and analytical section provides basic information about the Dow Jones index, the process of forming an algorithm for building a neural network, the relevance of the task of forecasting the future value of the index, and methods of solving this task.

In a special section, the construction of a machine learning algorithm is implemented.

The practical value of the work consists in gaining knowledge about the future values of the investment indicator, whether to buy or sell securities.

Keywords: DOW JONES INDEX, SECURITIES, FORECASTING, ANALYSIS, NEURAL NETWORK, MULTILAYER PERCEPTRON, TREND, NEURON, FORECASTING METHODS, TIME SERIES.

Зміст

ВСТУП

Актуальність теми: Займаючись інвестиціями, необхідно виробити певну політику, тактику і стратегію своїх дій, для цього потрібно від чогось відштовхуватись для прийняття рішення щодо купівлі продажу цінних паперів. Знання майбутніх значень інвестиційних показників таких як індекс Доу Джонса дуже корисне для інвестора. Дана робота присвячена актуальній для сучасного інвестора та фінансового діяча проблеми - проблеми прогнозування значення цінних паперів.

Одна з областей сучасної теорії інтелектуальних обчислень, що найбільш динамічно розвиваються, пов'язана з побудовою і застосуванням штучних нейронних мереж.

Нейронні мережі – це розділ штучного інтелекту, у якому обробки сигналів використовуються явища, аналогічні які у нейронах мозку живих істот. Найважливіша особливість нейронної мережі, що свідчить про її широкі можливості та величезний потенціал для вирішення обчислювальних завдань, полягає в паралельній обробці інформації всіма ланками ланцюга, що дозволяє прискорювати процес обробки інформації.

Інша не менш важлива властивість нейронної мережі - здатність до навчання та узагальнення накопичених знань. Нейронна мережа має риси штучного інтелекту. Натренована на обмеженій безлічі даних мережа здатна узагальнювати отриману інформацію та показувати адекватні результати на даних, які не використовувалися в процесі навчання [4].

Ця здатність, зокрема, використовується для прогнозування значень часових рядів.

Об'єктом дослідження: процес функціювання ринку акцій.

Предметом дослідження: методи прогнозування індексу Доу Джонса прогнозування за допомогою нейронних мереж.

Мета дипломної роботи: дослідження можливості застосування нейронної мережі до прогнозування Індекса Доу Джонса.

Для цього необхідно вирішити такі задачі:

1 Провести аналіз предметної області ринку цінних паперів, виявити фактори, що впливають на значення курсу акцій.
2. Провести аналіз методів прогнозування, за допомогою яких можна спрогнозувати майбутнє значення індексу.
3. Дослідити алгоритми нейронної мережі для задачі прогнозування.
4. Розробити програмний апарат для задачі прогнозування індексу Доу Джонса з використанням нейронної мережі.
5. Дослідити отриманий прогноз, як впливає структура мережі та початкові данні на отримані прогнозні значення.
6. На базі отриманого результату оцінити ефективність обраних методів для вирішення задачі прогнозування курсу акцій на біржі – а саме індексу Доу Джонса.

Методи дослідження: прогнозування на основы нейронної мережі, прогнозування методами Рухомого(ковзного) середнього та методом Зваженого середнього.

Теоретичне значення роботи полягає у застосуванні методів прогнозування для отримання майбутніх значень індексу Доу Джонса.

Наукова новизна отриманих результатів полягає у розробці програми, яка буде прогнозувати майбутні значення для заданого відрізку часового ряду за допомогою штучного інтелекту.

Практичне значення результатів кваліфікаційної роботи полягає у можливості застосування розробленого програмного забезпечення до розв’язання задачі прогнозування індексу Доу Джонса на основі попередніх даних

Дана робота складається зі вступу, трьох глав, висновку, списка використаних джерел, додатків.

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Огляд літератури

Заробити на фондовому ринку можна двома способами: за рахунок дивідендів та на коливаннях вартості акцій. У першому випадку все просто — компанія періодично сплачує частину прибутку власникам своїх паперів. Другий випадок складніший: треба купити, коли дешево, і продати, коли дорого. Найважче у цій схемі — вирішити, коли саме здійснити угоду. Від того, як точно інвестор передбачить динаміку цін, залежить прибуток.

Для вирішення задачі передбачення динаміку цін, була проаналізована велика кількість літератури та інформаційних джерел. Пошуки велись у трьох напрямках:

Перше завдання було набути знань в сфері економіки та інвестицій, що впливає на курси тої чи іншої компанії, деякі історичні приклади кардинальних змін на ринку цінних паперів, що стало причиною таких змін. В процесі обробки та вивчання в цьому напрямку мною було вирішено взяти для подальшого аналізу середній показник курсів акцій найбільших кампаній США - Індекс Доу-Джонса. Для більш глибокого знання індексу Доу-Джонса були вивчені історичні передмови виникнення цього індексу. Промисловий індекс Доу Джонса - найстаріший в історії фондової біржі Америки. Його автор - Чарльз Доу назавжди залишив свій слід у світовій економічній науці, вперше розрахувавши середній показник для оцінки руху ринку. Вже більше сторіччя це винахід не втратив своєї актуальності. DJIA є індикатором стану американської економіки і, разом з іншими показниками, використовуються для аналізу і прогнозування ринкової.

Прогнозування цього індексу є важливою задачі для інвесторів та для галузі економіки.

У книзі Айвазяна С.А., «Методи економетрики» були вивчені основні економічні поняття. Особливість даного видання полягає в тому, що в ньому в опис традиційних методів вирішення економетричних завдань вперше органічно вбудовані (там, де це дозволяє підвищити точність та глибину аналізу) сучасні методи багатовимірного статистичного аналізу, які раніше не включалися до інструментарію економетрики (зокрема, дискримінантний та кластер-аналізи, метод основних компонентів та ін.). Представлені в підручнику методи та моделі регресійного аналізу, бінарного та множинного вибору, аналізу часових рядів.

У роботі Пересада А.А. «Інвестиційний процес в Україні» були вивчена інвестиційна діяльність суб'єктів господарювання України.

Другим завданням було вивчення матеріалу щодо прогнозування числових рядів, бо значення індексу Доу Джонса являють собою числовий ряд. Були вивчені різні методи прогнозування числових рядів, їх недоліки та їх переваги.

Прогнозування часових рядів включає розробку та використання прогностичної моделі на основі даних, в яких існує впорядкований зв'язок між спостереженнями. Так як ряд який ми бажаємо спостерігати являє собою коливання курсу акцій, вплив на які дуже важко аналізувати, тому що дуже багато економічних, політичних і соціальних моментів оказують вплив на зростання або падіння курсу акцій. Тому було вирішено зробити алгоритм з використанням нейронної мережі, який будить мать ваги, які потрібно буде оптимізувати.

Третім завданням було вивчення що таке нейронна мережа, основні етапи її формування, різноманітні алгоритми, та створити власний алгоритм для задачі прогнозування індексу Доу Джонса.

У книзі Роберта Каллана «Нейронні мережі. короткий довідник» розглядаються основні моделі нейронних мереж, важливі для розуміння основ досліджуваного предмета, і обговорюються зв'язку між нейронними мережами і традиційними поняттями з області штучного інтелекту.

Навчання відбувається за рахунок адаптації мережі та налаштування ваг за допомогою числового алгоритму. За основу були взяті труди Ф. Уоссермен. “Нейрокомп'ютерна техніка”. У книзі американського автора у загальнодоступній формі викладаються основи побудови нейрокомп'ютерів. Описано структуру нейронних мереж та різні алгоритми їх настроювання. Окремі розділи присвячені питанням реалізації нейронних мереж.

На базі отриманих знань було прийнято рішення розробити власний алгоритм нейронної мережі для розв’язання задачі прогнозування індексу Доу Джонса.

1.2 **Індекс Доу Джонсона**

1.2.1 **Поняття індексу Доу Джонсона**

Індекс Доу-Джонса – це середній показник курсів акцій найбільших кампаній США, який публікує фірма Dow-Jones company з кінця 19 століття. Являє собою арифметичну середню (незважену) щоденних котирувань певної групи кампаній на момент закриття біржі. Метод обчислення ІНДЕКСУ ДОУ-ДЖОНСА неодноразово змінювався. У сучасних умовах публікується кілька індексів: для промислових кампаній на основі акцій 30 корпорацій (в т.ч. Амерікен телефоун енд телеграф компанії, Дюпон де Немур, Дженерал моторз і т.д.); залізничних кампаній (20 найбільших фірм) та комунальних кампаній (15 фірм). На основі зазначених індексів розраховується загальний ІНДЕКС. ІНДЕКС ДОУ-ДЖОНСА служить показником поточної господарської кон'юнктури США та відображає реакцію американських ділових кіл на різні економічні та політичні події

Зміна показань індексу Доу-Джонса

Офіційно опубліковане значення індексу Доу-Джонса дорівнювало 40.94. Показник складався як середньоарифметичне від вартості акцій підприємств, що становлять «кошик» показника. З розвитком економіки США, змін у складі компаній, його значення змінювалося кардинально. Так, максимальна планка в 1000 пунктів була досягнута в 1966 році, яка зберігалася протягом 15 років, протягом яких вартість індексу значно падала (до третини від максимуму) і поверталася до вихідної точки.

У 1980-90-х роках XX століття спостерігалося вибухове зростання вартості індексу Доу-Джонса аж до позначки 11722.98 (січень 2000 року). Усередині цього періоду виявлялися падіння до 22%, але значного впливу на загальну динаміку ринкової вартості цього показника це не мало. Найпомітніший обвал був зафіксований у «чорний понеділок» 1987 року. На кінець 2017 року величина, що торгується, знаходилася в межах 23 000 одиниць.

Плюси та мінуси застосування індексу Доу-Джонса

Одним із ключових напрямків застосування індексу Доу-Джонса є його використання як торговий інструмент на фондовій біржі. Він легко піддається аналітиці, що призводить до включення індексу до інвестиційних портфелів великої кількості трейдерів. Частина учасників біржової торгівлі здійснюють угоди переважно у ньому.

Із плюсів відзначають таке:

Показник існує великий проміжок часу, що дозволяє об'єктивно порівнювати стан американської економіки будь-якому етапі.

Компанії, що входять до індексу, визнаються надійними та стійкими, включаючи їх акції, інші активи (що активно використовують трейдери).

Простота обчислення без застосування спеціальних формул (застосовується середнє арифметичне).

Існують і значні мінуси:

Усі компанії мають однакову «вагу» при розрахунку усередненого показника. При падінні курсу цінних паперів одного підприємства загальне значення може вільно компенсуватися зростанням котирувань інших фірм, що здатне ввести трейдера в оману.

Лише 30 компаній який завжди об'єктивно відбивають реальний стан ринку США. Значення добре допомагає при аналізі певних проміжків, але на ринку, що діє, має «невелику об'єктивність».

Останній фактор призводить до втрати популярності даного торгового інструменту, причому на користь аналогічного показника індексу S&P 500, що включає значення 500 американських компаній. Таке розширення «кошика» було потрібне для загальної оцінки внутрішнього ринку США, його взаємодії з іншими країнами. [1]

1.2.2 Фактори, які мають вплив на значення індексу Доу Джонсона

Промисловий індекс Доу Джонса (DJIA, Dow Jones Industrial Average) — це якийсь «збиральний образ» американського бізнесу з високою капіталізацією. Інвестування в нього здається надійним через тривале історичне зростання — всі компанії-учасники свого часу мали статус найбільш успішних підприємств. Однак індекс не є «Граалем», що гарантує, що інвестор отримає прибуток.

Від чого залежить ціна на промисловий індекс Доу Джонса

На індекс Dow Jones впливають як макроекономічні чинники, і стан американської економіки. Чим довша тривалість угоди, тим більшим пріоритетом буде аналіз глобальної економічної ситуації, чим коротше, тим важливіше дивитися на локальні економічні зміни.

1. Економічні події

DJIA є чутливим до ситуації на світовій арені. Плідна економічна співпраця між США та іншими країнами з великою економікою, укладання нових торгових

спілок та стабільна політична ситуація сприяють позитивній динаміці ціни індексу Dow Jones.

Вплив надає й ситуація всередині країни: чим здоровіша економіка США — низькі відсоткові ставки, низька інфляція, висока продуктивність праці, фактичний розвиток у межах чи краще прогнозованих показників, тим явнішим зростанням реагує Dow Jones.

2. Курс американського долара

Індекс Dow Jones має зворотну кореляцію з національною американською валютою. Приріст індексу збігається з ослабленням долара США, зниження DJIA — з зміцненням.

3. Новини

Найбільший вплив на майбутню вартість індексу Dow Jones мають важливі новини, пов'язані з американською економікою, включаючи засідання Федерального комітету відкритого ринку США (FOMC, Federal Open Market Committee), звіти з безробіття (NFP, Non-Farm Payrolls), індекс споживчих цін (CPI) , Consumer Price Index).

У короткостроковій перспективі вихід цих новин підвищує волатильність ціни індексу. У середньостроковій — новини позитивно впливають на Dow Jones, якщо фактичні значення рівні або кращі за прогнозні.

4. Звіти про доходи

Компанії, що входять до DJIA, щокварталу публікують звіт про доходи.

Зростання доходів компанії, зазвичай, мотивує інвестувати у її цінних паперів, оскільки передбачає виплату хороших дивідендів чи купонів у майбутньому. Збільшення попиту з боку інвесторів сприяє зростанню цін пайових інструментів компаній, що позитивно впливає на динаміку цін індексу

Dow Jones. Зниження чи стагнація доходів цих компаній діють індекс протилежним чином.

5. Вартість акцій окремих компаній

Оскільки DJIA є складовим інструментом, динаміка його ціни залежить від вартості акцій компаній-учасників. Індекс зростатиме, якщо зростання цін одних акцій перевищує величину падіння інших цін. Найбільше впливають дорогі акції, оскільки величина зростання чи падіння їх цін більше, ніж в дешевших.

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Методи прогнозування

Прогнозування (грец. – знання наперед) — процес передбачення майбутнього
стану предмета чи явища на основі аналізу його минулого і сучасного, систематично
оцінювана інформація про якісні й кількісні характеристики розвитку обраного предмета
чи явища в перспективі.

Метою прогнозування є визначення напрямів і тенденцій розвитку будь - якого
процесу, ймовірних термінів появи тих або інших подій, обсягів необхідних робіт.

Етапи розробки прогнозу:

1) отримання необхідної інформації;

2) її обробка, оцінка і аналіз;

3) визначення перспектив та ймовірність реалізації прогнозу. Всі етапи розробки
прогнозу мають бути узгоджені з поставленими цілями і завданнями, в процесі прогнозування визначаються також його предмети.

Метод поворотних значень (критерій піків)

Під випадковим часовим рядом будемо розуміти такий ряд, у якому значення є випадковими незалежними величинами, що мають один закон розподілу. Якщо вдається показати, що часовий ряд є випадковим, то подальше його дослідження можна припинити, вирахувавши, при необхідності, середнє значення та дисперсію ряду або інші статистичні характеристики.

Одним з методів перевірки часового ряду на випадковість є метод поворотних значень.

Для часового ряду $\{YT\}$ значення yi є поворотним, якщо $yi-1 < yi > yi+1$ або $yi-1 > yi < yi+1$. У першому випадку значення yi є «піком», у другому — «впадиною». Зауважимо, що перше й останнє значення не можуть бути поворотними.

Метод поворотних значень зводиться до підрахунку кількості впадин і піків у ряду $\{YT\}$ і порівнянні цієї кількості з теоретичним значенням, яке дорівнює математичному сподіванню кількості поворотних точок у «чисто випадковому» ряду, що складається з Т спостережень.

Позначимо загальне число поворотних точок через p. Математичне очікування числа точок повороту $\bar{p}$і дисперсія σ_p^2 виражаються формулами:

$$\bar{p}=\frac{2}{3}(T-2),\ \sigma_p^2=\frac{16T-29}{90}.$$

Критерієм випадковості з 5% -ним рівнем значущості, тобто з довірчою ймовірністю 95%, є виконання нерівності

$$p > [\bar{p} - 1{,}96\sqrt{\sigma_p^2}],$$

де квадратні дужки означають цілу частину числа. Якщо ця нерівність не виконується, то модель вважається неадекватною.

Метод рухомого (ковзного) середнього.

Методи згладжування використовуються для зменшення впливу випадкового компонента (випадкових коливань) у часових рядах. Вони дають можливість отримувати більш “чисті” значення, які складаються лише з детермінованих компонентів. Деякі з методів направлені на виділення лише деяких компонентів, наприклад, тренду.

До основних методів аналізу часових рядів можна віднести метод ковзного середнього, експоненціального згладжування і проектування тренда.

Метод ковзного середнього,є одним з найпростіших, який дозволяє виділити тренд. Для застосування цього методу дослідник повинен мати доволі довгий ряд спостережень. При визначенні тенденції за допомогою ковзаючої

середньої треба сформувати укрупнені інтервали, що складаються з однакової кількості рівнів. Кожен наступний інтервал отримують, поступово рухаючись від початкового рівня динамічного ряду на один рівень. За сформованим більш крупним інтервалом визначають суми значень рівнів і розраховують плинні середні. При згладжуванні ряду за допомогою ковзаючої середньої більш крупний інтервал складають з непарної кількості рівнів ряду.

Для загального випадку розрахункова формула виглядає так:

$$f_k \frac{x_{k-N} + x_{k-N+1} + \cdots + x_{k-1}}{N}$$

або

$$f_k = \frac{1}{N}\sum_{i=1}^{N} x_{k-i}$$

Де x_{k-i} – реальне значення показника у момент часу tk-i; N – число попередніх моментів часу; fk – прогноз на момент часу tk.

Більш гладкий тренд дозволяє виділити метод подвійного усереднення, яке двічі використовує усереднення часового ряду. При цьому кількість спостережень зменшується на два повних цикли сезонності, тому для використання методу необхідно мати часовий ряд, який складається щонайменше з 3-х повних циклів сезонності.

Метод зваженого (ковзного) середнього

При прогнозуванні методом усереднювання часто доводиться спостерігати, що ступінь впливу використаних при розрахунку реальних показників виявляється неоднаковим, при цьому звичайно більш «свіжі» дані мають більшу вагу. Математично метод зваженого рухомого середнього можна записати як

$$f_k = \frac{\sum_{i=1}^{N} w_{k-i} x_{k-i}}{\sum_{i=1}^{N} w_{k-i}}$$

де xk-i – реальне значення показника у момент часу;

N – кількість попередніх моментів часу, що використовуються при розрахунку;

fk – прогноз на момент часу tk;

wi – вага, з якою використовується показник xk-i при розрахунку.

Вага – це завжди додатне число. У разі, коли вся вага однакова, ми одержуємо формулу.

Метод експоненціального згладжування

Експоненціальне згладжування – це дуже популярний метод прогнозування багатьох часових рядів. Історично метод був незалежно відкритий Брауном і Холтом. Цей метод значно переважає попередні методи. Найкраще цей метод зарекомендував себе, коли дані мають дуже гладкий або навіть горизонтальний тренд.

Розрахунок прогнозу методом експоненціального згладжування проводиться за формулою, де α – стала згладжування (0 < α < 1):

$$S_1 = y_i,$$

$$S_t = \alpha y_t + (1-\alpha)S_{t-1}, t = \overline{2,T} \;\; 0 < \alpha < 1.$$

Єдина вага α може обиратися кількома шляхами. По-перше, якщо обирається значення близьке до 1, то будуть більш важливими при прогнозуванні останні дані часового ряду, при виборі α близьким до 0, більш впливовими будуть минулі значення. По-друге, можна прокласти $\alpha = \frac{2}{T+1}$. По-третє - вибір α, при якому мінімізується один з критеріїв точності прогнозів на n періодів. При цьому розрахунки повинні проводитися лише по перших T – n значеннях часового ряду, а отримані прогнози повинні бути порівняні з реальними даними. При використанні цієї методики відкидається обмеження

$$0 < \alpha < 1,$$

Прогноз значень часового ряду дорівнює останньому члену послідовності S_t:

$$\hat{y}_{T+p} = S_T, \quad p = 1,2, \dots .$$

Початкове значення послідовності можна вибрати як

$$S_1 = y_1 \;\; \text{або} \;\; S_1 = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Очевидно, результат згладжування залежить від параметра α. На практиці звичайно рекомендується брати α від 0.1 до 0,3. Крім того, параметр згладжування α часто знаходиться пошуком на сітці. Можливі значення параметра α розбиваються сіткою з певним кроком. Наприклад, розглядається сітка значень α від α = 0,1 до α = 0,9, з кроком 0,1. Потім вибирається той α, для якого сума квадратів (або середніх квадратів) залишків (значення, що спостерігаються, мінус прогнози на крок вперед) є мінімальною.

Подвійне експоненціальне згладжування Брауна

Цей метод будується аналогічно попередньому, тільки процес згладжування робиться двічі:

$$S_1 = y_1,$$

$$S_t^{`} = \alpha y_t + (1-\alpha) S_{t-1}^{`},$$

$$S_t^{``} = \alpha S_t^{`} + (1-\alpha) S_{t-1}^{``}, \qquad t = \overline{2,T} \;\; 0 < \alpha < 1.$$

Метод використовується, коли дані часового ряду нестаціонарні. Прогноз будується як останнє значення другої послідовності:

$$\hat{y}_{T+p} = S_t^{``}, \quad p = 1,2, \ldots .$$

Модель Холта застосовують, якщо необхідно враховувати тренд. Основна ідея методу: додавання значення тренда до прогнозованого моделлю експоненціального згладжування значенням.

Метод Хольта використовується для прогнозування часових рядів, коли є тенденція до зростання або падіння значень часового ряду. А також для рядів, коли дані тобто не за повний цикл, і сезонність ще не виділити (наприклад, за неповний рік для прогнозу по місяцях).

Якщо часовий ряд має тенденцію до зростання або падіння, то разом з оцінкою поточного рівня ряду (як в простому експоненціальному згладжуванні) варто виділити тренд. Для управління рівнем і нахилом в моделі Хольта вводиться 2 коефіцієнта згладжування - коефіцієнт згладжування ряду і тренда.

$$S_2^{`} = y_2, \qquad S_2^{``} = y_2 - y_1,$$

$$S_t^{`} = \alpha y_t + (1-\alpha)(S_{t-1}^{`} + S_{t-1}^{``}), 0 < \alpha < 1,$$

$$S_t^{\ddot{e}`} = \beta(S_t` - S_{t-1}`) \quad + (1-\beta)S_{t-1}``, 0 < \beta < 1,$$

Прогноз на наступні періоди:

$$\hat{y}_{T+p} = S_t`` + pS_t``, \quad p = 1,2, \dots .$$

Метод проектування тренда

Алгоритм Фаррара-Глобера

Найповніше дослідити мультиколінеарність можна з допомогою алгоритму Фаррара — Глобера. Цей алгоритм має три види статистичних критеріїв, згідно з якими перевіряється мультиколінеарність всього масиву незалежних змінних (- «хі» — квадрат); кожної незалежної змінної з рештою змінних (F-критерій); кожної пари незалежних змінних (t-критерій).

Усі ці критерії при порівнянні з їх критичними значеннями дають змогу робити конкретні висновки щодо наявності чи відсутності мультиколінеарності незалежних змінних.

Опишемо алгоритм Фаррара — Глобера.

Крок 1. Стандартизація (нормалізація) змінних.

Позначимо вектори незалежних змінних економетричної моделі через $x_1, x_2, x_3 \dots x_m$. Елементи стандартизованих векторів обчислио за формулою:

$$x_{ik}^{*} = \frac{x_{ik} - \overline{x_k}}{\sqrt{n\sigma_{x_k}^2}},$$

де n — число спостережень $(i = \overline{1,n})$;

m —число пояснювальних змінних, $(k = \overline{1,m})$;

$\overline{x_k}$ —середнє арифметичне *k*-ї пояснювальної змінної;

$\sigma_{x_k}^2$ —дисперсія *k*-ї пояснювальної змінної.

Крок 2. Знаходження кореляційної матриці

$$r = X^{*} \leqslant X^{*},$$

де X^{*} — матриця стандартизованих незалежних (пояснювальних) змінних, X^{*0} — матриця, транспонована до матриці X^{*}.

Крок 3. Визначення критерію χ^2(«хі»-квадрат):

$$\chi^2 = -\left[n - 1 - \frac{1}{6}(2m+5)\right]\ln|r|,$$

де $|r|$ — визначник кореляційної матриці*r*.

Значення цього критерію порівнюється з табличним при $\frac{1}{2}m(m-1)$ ступенях свободи і рівні значущості α. Якщо $\chi^2_{\text{факт}} > \chi^2_{\text{табл}}$, то в масиві пояснювальних змінних існує мультиколінеарність.

Крок 4. Визначення оберненої матриці:

$$C = r^{-1} = (X^* X)^{-1}.$$

Крок 5. Очислення *F*-критеріїв:

$$F_k = (c_{kk} - 1)\frac{n - m}{m - 1},$$

де c_{kk} — діагональні елементи матриці *C*. Фактичні значення критеріїв порівнюються з табличними при *n* – *m* і *m* – 1 ступенях свободи і рівні значущості а. Якщо F_kфакт > *F*табл, то відповідна *k*-та незалежна змінна мультиколінеарна з іншими.

Коефіцієнт детермінації для кожної змінної

$$R^2_{x_k} = 1 - \frac{1}{c_{kk}}.$$

Крок 6. Знаходження частинних коефіцієнтів кореляції:

$$r_{kj} = \frac{-c_{kj}}{\sqrt{c_{kk}\ c_{jj}}},$$

де c_{kj} — елемент матриці*C*, що міститься в *k*-му рядку і *j*-му стовпці; c_{kk} і c_{jj} — діагональні елементи матриці*C*.

Крок 7. Обчислення *t*-критеріїв:

$$t_{kj} = \frac{r_{kj}\sqrt{n - m}}{\sqrt{1 - r_{kj}^2}}.$$

Фактичні значення критеріїв t_{kj} порівнюються з табличними при $n - m$ ступенях свободи і рівні значущості α. Якщоt_{kj}(ф) > *t* табл, то між незалежними змінними x_k і x_j існує мультиколінеарність.

2.2 Нейронні мережі

Шту́чні нейро́нні маре́жі (ШНМ, англ. artificial neural networks, ANN), або конективістські системи (англ. connectionist systems) — це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу. Наприклад, у розпізнаванні зображень вони можуть навчатися ідентифікувати зображення, які містять котів, аналізуючи приклади зображень, мічені[en] як «кіт» і «не кіт», і використовуючи результати для ідентифікування котів в інших зображеннях. Вони роблять це без жодного апріорного знання про котів, наприклад, що вони мають хутро, хвости, вуса та котоподібні писки. Натомість, вони розвивають свій власний набір доречних характеристик з навчального матеріалу, який вони оброблюють.

ШНМ ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами (аналогічно до біологічних нейронів у головному мозку тварин). Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, й потім сигналізувати штучним нейронам, приєднаним до нього.

В поширених реалізаціях ШНМ сигнал на з'єднанні між штучними нейронами є дійсним числом, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу[en], яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів. Сигнали проходять від першого (входового) до останнього (виходового) шару, можливо, після проходження шарами декілька разів.

Первинною метою підходу ШНМ було розв'язання задач таким же способом, як це робив би людський мозок. З часом увага зосередилася на відповідності певним розумовим здібностям, ведучи до відхилень від біології. ШНМ використовували в ряді різноманітних задач, включно з комп'ютерним баченням, розпізнаванням мовлення, машинним перекладом, соціально-мережевим фільтруванням, грою в настільні та відеоігри, та медичним діагностуванням. [2]

Штучна нейронна мережа (ІНС) (англ. Artificial neural network (ANN)) — спрощена модель біологічної нейронної мережі, що є сукупністю штучних нейронів, що взаємодіють між собою.

Основні принципи роботи нейронних мереж були описані ще в 1943 Уоррен Мак-Каллоком і Уолтером Піттсом . В 1957 нейрофізіолог Френк Розенблатт розробив першу нейронну мережу, а в 2010 великі обсяги даних для навчання відкрили можливість використовувати нейронні мережі для машинного навчання. На даний момент нейронні мережі використовуються у численних галузях машинного навчання та вирішують проблеми різної складності.

2.3. Багатошаровий персептрон

2.3.1 Штучний нейрон

На малюнку 1 показано структуру штучного нейрона з n входами. Кожен вхідний канал і може передавати речове значення. Активаційна функція обчислюється в тілі нейрона і може бути довільно обрана. Як правило

вхідні сигнали мають відповідну вагу, що означає, що компоненти вектора вхідного сигналу множиться на відповідну вагу. Передана інформація зазвичай об'єднується в нейроні за допомогою зваженого підсумовування, а потім обчислюється активаційна функція.

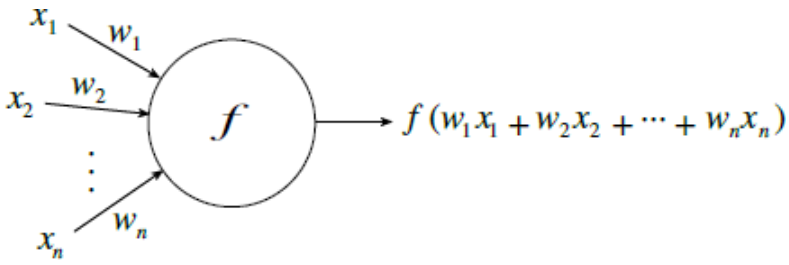

Рис. 2.1. Модель абстрактного нейрона.

Якщо ми сприймаємо кожен нейрон в нейронній мережі як елементарну функцію зі здатністю трансформувати свій вхідний сигнал на точно визначений вихідний сигнал, то нейронна мережа є композицією активаційних функцій з набором параметрів, якими є вагові коефіцієнти зв'язків. Різні моделі штучних нейронних мереж різняться переважно активаційними функціями нейронів мережі, методом міжнейронних зв'язків і за часом передачі.

Найчастіше передбачається, кожен нейрон надає адитивний внесок для вихідного нейрона, з яким з'єднаний. Повний вихідний сигнал k-го нейрона є значенням функції активації від виваженої суми окремих виходів від кожного підключеного нейрона плюс зміщення або зсув:

$$s_k(t) = f(\sum_{I} w_{jk}(t)y_j(t) + \theta_k(t)), \qquad (2.1)$$

Внесок позитивних ваг w_{jk} буде вважатися збудженням, а внесок негативних ваг w_{jk} гальмуванням. В некоторых случаях используются более сложные правила для объединения входящих сигналов, которые делятся между собой на возбуждающие и тормозящие входные сигналы.

Часто функция активации является неубывающей функцией от всего входящего сигнала нейрона:

$$y_k(t+1) = \nearrow_k(s_k(t)) = \nearrow_k (\sum_{J} w_{jk}(t)y_j(t) + \theta_k(t)), \qquad (2.2)$$

Як функції активації (рис.2) часто використовуються такі типи

функцій: жорстко обмежена порогова функція (sign - функція), або лінійна, або напівлінійна або плавно-обмежена порогова функція. Для плавно-обмежених функцій часто sigma-функція виглядає так:

$$y_k = \nearrow(s_k) = \frac{1}{1 + e^{-s_k}}, \tag{2.3}$$

У деяких додатках використовується гіперболічний тангенс, що отримує вихідні значення діапазону [-1, +1].

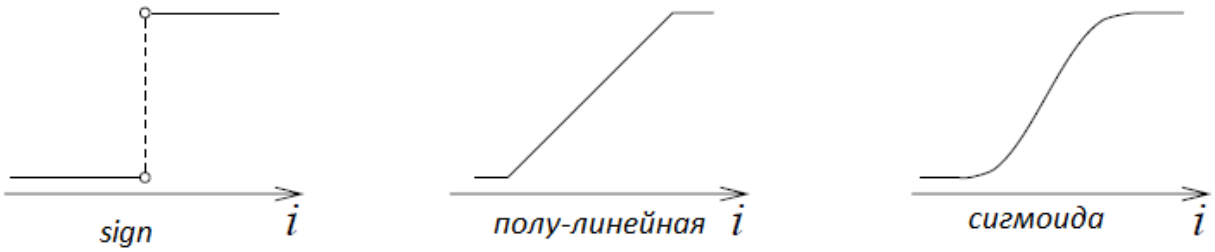

Рис. 2.2 Графіки деяких функцій активації.

У деяких випадках вихідний сигнал може бути функцією стохастичної функцією вхідного сигналу. У такому разі активація нейрона не визначається детерміновано - певними вхідними нейронами, але вхідні нейрони визначають можливість p, того, що нейрон отримає високе значення активації:

$$p(y_k \to 1) = \frac{1}{1+e^{-s_k/T}}, \tag{2.4}$$

де T (температура) є параметром, який визначає нахил імовірнісної функції.

2.3.2 Архітектура багатошарового персептрону

У 1958 році, американський психолог, запропонував персептрон, як більш

загальну обчислювальну модель, ніж модель Маккалока- Істотний нововведення стало введення чисельних ваг і спеціальної конфігурації сполучних зв'язків. В оригінальній моделі Розенблатта обчислювальні нейрони – це порогові та сполучні елементи, визначені стахостично. Навчання відбувається за рахунок адаптації мережі та налаштування ваг за допомогою числового алгоритму. Модель Розенблатта була доопрацьована та вдосконалена у 1960 роки, та її обчислювальні властивості були ретельно проаналізовано та . Надалі модель Розенблатта почала називатися класичним персептроном.

Класичний персептрон складніший, ніж одношарова мережа з граничною функцією активації. У своїй найпростішій формі вона складається з елементів N вхідного шару, які подаються на K елементів прихованого шару, а потім, на вихідний одиночний нейрон. Метою роботи персептрона є вивчення заданого перетворення 1 1 1 1 з використання навчальних прикладів з вхідними сигналами x і відповідними сигналами y=f(x). У початковому визначенні, активаційною може бути будь-яка функція, але процедура навчання регулює лише ваги зв'язків із вихідним нейроном. Залежно від функції активації персептрони можуть бути згруповані у різні сімейства. У 1969 році Мінський та Пейперт описали кількість таких сімейств та їх властивості.

В даний час під багатошаровим персептроном зазвичай розуміють мережу прямого поширення сигналів, без зворотних зв'язків, що складається з кількох шарів нейронів з функцією активації сигмоїдальної. Кожен нейрон має зв'язки з усіма нейронами попереднього шару та з усіма нейронами наступного шару. Нейрони, що належать одному шару, не мають зв'язків між собою. Всі шари багатошарового персептрона, крім останнього, називаються прихованими.

Таким чином, багатошаровий персептрон має шарувату структуру (рис. 1.4.2.1). Кожен шар складається з нейронів, які отримують свої вхідні сигнали від стоячи нижче нейронів, і посилають свої вихідні значення на шар, який відповідно знаходиться вище. Немає жодних сполучних зв'язків у межах одного шару. Вхідні сигнали подаються одночасно перший шар із прихованими нейронами, т.к. перший вхідний шар служити лише ініціалізації вхідного

вектора, ніякої обробки сигналу цьому шарі немає. Виходи прихованих нейронів у свою чергу розподіляються по наступному свою прихованих нейронів, і так відбувається доти, доки досягається останній шар прихованих нейронів. вихідний шар

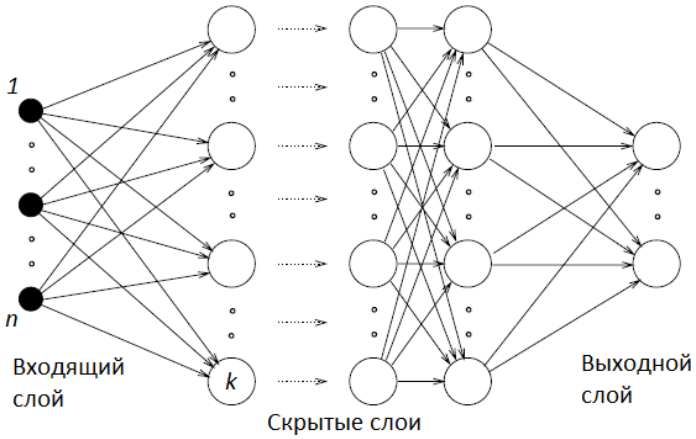

Рис. 2.3. Узагальнений вид багатошарового персептрону.

2.3.3. Завдання навчання багатошарового персептрону

Існує кілька формулювань завдання навчання для багатошарового персептрону. Формально завдання навчання багатошарового персептрона можна сформулювати як завдання наближеної інтерполяції.

Дано: На вхід нейронної мережі подається безліч навчальних пар:

$\{(x^i, d^i)\}$, $i=1,\ldots,k$, где $x^i = (x^i_1, x^i_2, , x^k_n)$, $d^i = (d^i_1, d^i_2, , d^i_m)$.

Також нам відома похибка s_1 для навчальної множини.

Необхідно: навести вагові коефіцієнти w_{ij} к такому виду, що:

$\|F(x^i) - d^i\| < s_1 \bigstar i = 1..k.$

Після знаходження вагових коефіцієнтів нейронна мережа зможе відновити вихідні дані з необхідною точністю. Тим не менш, головним для нас завданням залишається хороше відновлення даних, що не брали участь у навчанні і не входять у вихідне безліч вхідних сигналів. Тому необхідно

ввести додаткову контрольну множину даних, щоб виконували узагальнюючі властивості нейронної мережі. Нова сформована множина даних не буде використовуватися для навчання мережі, і знадобиться нам, тільки на етапі перевірки того, як добре навчилася нейронна мережа, вже після того, коли ми переконаємося, що ми навчили нейронну мережу належним чином. І обидва множини, як контрольна, так і навчальна, повинні бути досить різноманітні та ефективні для навчання мережі та подальшої її перевірки.

Дано: На вхід мережі подається дві множини навчальних даних:

навчальна множина $\{(x^i, d^i)\}$, $i=1,\dots,k$, и

контрольна множина $\{(x^i, d^i)\}$

$i=k+1,\dots,k+s$, де $x^i = (x^i_1, x^i_2, \dots, x^k_n)$, $d^i = (d^i_1, d^i_2, \dots, d^i_m)$. Також дана похибка s_1 на навчальній множині, похибка s_2 на контрольній множині.

Необхідно: навести вагові коефіцієнти w_{ij} до такого виду, що:

$$\|F(x^i) - d^i\| < s_1, \forall i = 1, \dots, k \text{ и } \|F(x^i) - d^i\| < s_2, \forall i = k+1, \dots, k+s.$$

Для обох формулюваннях $F(x)$ - функція, що реалізується нейронною мережею.

Таким чином, основним завданням, яке необхідно вирішити, є приведення нейронної мережі до такого виду, щоб різниця очікуваного та вихідного значення нейронної мережі не перевищувала заданої похибки навчання. Ще одне формулювання завдання навчання виглядає так:

Дано: безліч пар навчальних даних: $\{(x^i, d^i)\}$, $i=1,\dots,k$, також відома похибка для навчання s_1 0.

Знайти: вагові коефіцієнти w^k_{ij} такі, що:

$$E(w) = \frac{1}{2}\sum_{i=1}^{k} \|F(x^i) - d^i\|^2 = \frac{1}{2}\sum_{i=1}^{k} (F(x^i) - d^i)^2 \leq \frac{s^2}{2}k = s \qquad (2.3.3.1)$$

Таке завдання також можна назвати завданням про мінімізацію цільової функції $E(w)$.

Існує велика різноманітність методів оптимізації для вирішення завдання навчання нейронної мережі. У зв'язку з тим, що активаційні функції є гладкими, ми можемо використовувати спрямовані методи оптимізації, у тому числі градієнтні методи.

2.3.4. Алгоритм навчання багатошарового персептрону

Під час навчання нейронної мережі навчальні дані подаються на вхідний шар мережі. Цей етап називається прямим перебігом алгоритму зворотного поширення. Під час прямого ходу, кожен вузол у прихованому шарі отримує від решти вузлів вхідного шару значення, які перемножуються з відповідними ваговими коефіцієнтами і потім підсумовуються. Вихідні сигнали нейронів є нелінійними перетвореннями функції активації. Аналогічно, кожен вихідний вузол отримує від усіх нейронів в прихованому шарі значення, які виходять, які також перемножуються з відповідними ваговими коефіцієнтами і потім підсумовуються. Вихід кожного вихідного нейрона є функція нелінійного перетворення активаційної функції.

Вихідні значення останнього шару порівнюються з ідеальними вихідними значеннями. Ідеальними вихідними значеннями вважаються вихідні навчальні дані. Помилка між ідеальним вихідним значенням і фактичним, отриманим після проходження сигналу через мережу розраховується і поширюється до прихованого шару. Такий напрямок називається зворотним ходом алгоритму зворотного поширення помилки. Помилка використовується, щоб оновити вагові коефіцієнти між вузлами, наприклад, перераховуються ваги матриць між вхідним-прихованим і прихованим-вихідним шарами.

Алгоритм зворотного поширення використовується для розрахунку градієнта цільової функції - суми квадратів помилок щодо кожної

компоненти вихідного сигналу. З використанням градієнта цільової функції ми можемо знайти мінімум функції помилки у ваговому просторі за допомогою градієнтних методів. Знайдена комбінація терезів, за допомогою яких мінімізується функція помилки, вважається вирішенням завдання навчання. Оскільки цей спосіб вимагає

Обчислення градієнта функції помилки на кожному кроці ітерації, ми повинні гарантувати диференціювання функції помилки. Очевидно, що ми маємо використовувати гладкі функції активації, використання порогових функцій активації неможливе. Одна з найбільш популярних функцій активації - це речова сигма - функція, що визначається виразом:

$$s_c(x) = \frac{1}{1 + e^{-cx}}, \tag{2.5}$$

Постійна с може бути обрана довільно та її зворотна величина *1/c* називається температурним параметром у стохастичних нейронних мережах. Форма сигма - функції змінюється відповідно до значення параметра с, як можна побачити на малюнку 1.4.3.1. Графік показує форму сигма - функції при с=1, с=2, с=3. Найбільше значення робить сигма - функцію схожою на ступінчасту функцію, а в межі з сигма - функція є ступінчастою (пороговою) функцією. З метою спрощення всіх виразів, які ми отримуватимемо надалі, ми вважатимемо все з = 1. Надалі ми називатимемо сигма - функцію

$s1(x)$ просто $s(x)$.

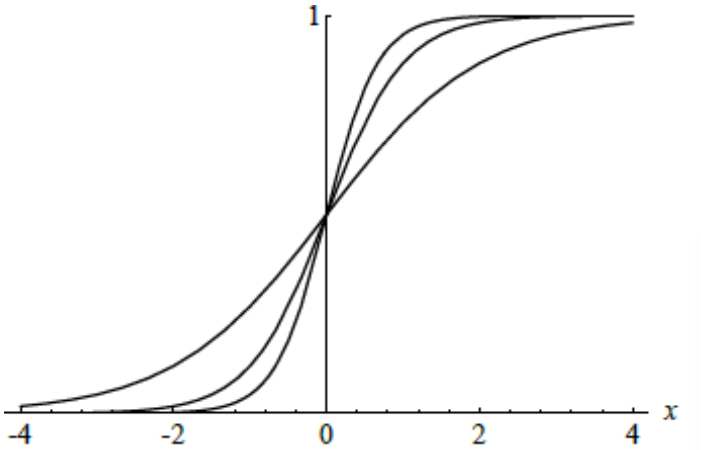

Рис. 2.4. Три сигма - функції (= 1, = 2, =).

Похідна від сигма – функції має вигляд:

$$\frac{d}{dx}s(x) = \frac{e^{-x}}{(1+e^{-x})^2} = s(x)(1-s(x)), \tag{2.6}$$

У разі персептрону симетрична функція активації має низку переваг для навчання. Альтернативою сигма – функції є біполярна (симетрична) сигма – функція, що визначається виразом:

$$S(x) = 2s(x) - 1 = \frac{1-e^{-x}}{1+e^{-x}}, \tag{2.7}$$

Це не що інше, як гіперболічний тангенс від аргументу x/2, форма якого показана на малюнку 2.5. На малюнку показані чотири типи безперервно - диференційованих функцій.

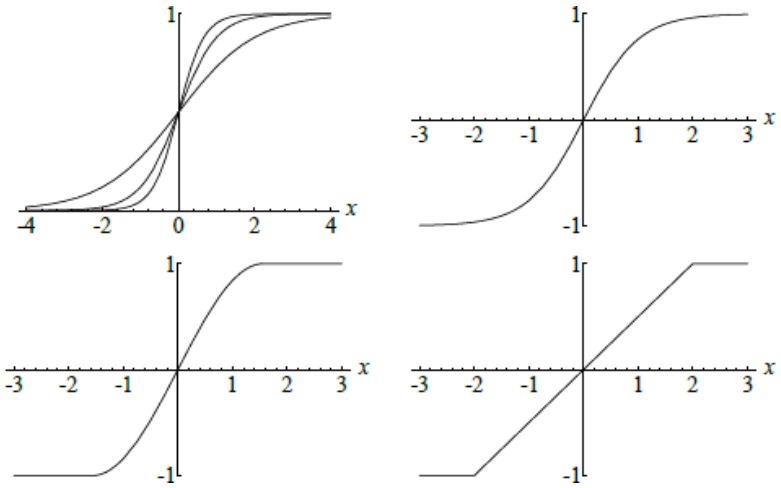

Рис. 2.5. Графіки деяких функцій активації, що диференціюються.

Так як спуск у просторі ваг здійснюється в напрямку градієнта, щоб знайти мінімум цієї функції, важливо, щоб не існувало таких ділянок, на яких цільова функція є плоскою.

Вихідний діапазон сигма - функції містить усі числа строго між 0 і 1. Обидва крайні значення можуть бути досягнуті лише асимптотично. Використовуючи вагові коефіцієнти - w_1, w_2,..., w_n, та зміщення - θ, кожен сигмоїдальний нейрон за вхідним вектором x_1, x_2,..., x_n, обчислює вихідне значення у вигляді:

$$\frac{1}{1 + exp\left(\sum_{i=1}^{n} w_i x_i - \theta\right)}, \tag{2.8}$$

Локальні мінімуми функції помилки - це недолік, який притаманний будь-якому спрямованому методу мінімізації, що використовується під час навчання нейронних мереж. На малюнку 1.4.3.3 показаний приклад локального мінімуму з вищим рівнем помилки, ніж у інших місцях функції. У поверхні відгуку функції помилки існує так звана "долина", і якщо градієнтний спуск починається там, алгоритм навчання зупиниться - відбувається так званий параліч навчання.

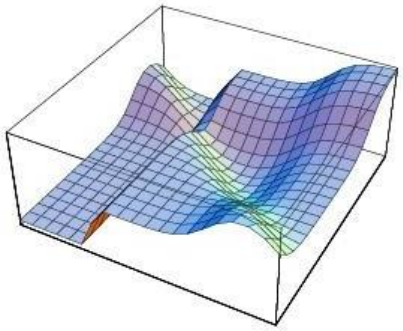

Рис. 2.6. Локальний мінімум функції помилки.

Локальні мінімуми з'являються дуже часто, тому що кінцеві значення вихідних сигналів є відмінними від нуля або одиниці.

Розглянемо завдання навчання багатошарового персептрона з n вхідними та m вихідними нейронами. Також нам дано навчальний набір $\{(x_1, t_1), \ ...,(x\ ,\ t\ )\}$, що складається з p упорядкованих пар m- на n-мірних векторів, які називаються вхідними та вихідними сигналами.. Ваги зв'язків будуть речовими числами, вибрані довільно. Коли вхідний сигнал x_i з навчальної множини надається на вхід даної мережі, то мережа вираховує інший вихідний сигнал i, який відрізняється від навчального вихідного сигналу t_i. Наша мета - зробити обидва сигнали t_i і $_i$ максимально схожими на i = 0,... , p, використовуючи алгоритм навчання. Точніше, ми хочемо зменшити функцію помилки мережі, яка визначається як:

$$E = \frac{1}{2}\sum_{i=1}^{p} "o_i - t_i"^2, \tag{2.9}$$

до наперед заданого значення. У цьому важливо зазначити, що завдання навчання передбачає зменшення цільової функції, а чи не її мінімізацію, тобто пошук локального чи глобального мінімуму. При цьому часто таке завдання називають завданням мінімізації.

Алгоритм зворотного поширення використовується пошуку локального

мінімуму функції помилок. Мережа ініціалізується випадковими ваговими коефіцієнтами. Розраховується градієнт функції помилки та коригуються початкові ваги з урахуванням градієнта.

Для мінімізації цільової функції E використовується ітераційний процес градієнтного спуску, для якого необхідно обчислити градієнт.

$$\mathrm{A}E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_l}\right), \tag{2.10}$$

Кожна вага змінюється на деяке збільшення :

$$\Delta w_{\mathrm{i}} = -\gamma \frac{\partial E}{\partial w_{\mathrm{i}}} \quad for\ \mathrm{i} = 1, \dots, l, \tag{2.11}$$

де γ є постійну навчання, тобто. параметр пропорційності, який визначає довжину кроку кожної ітерації в негативному напрямку градієнта. Цей параметр називається коефіцієнтом навчання та визначає вид використовуваного методу спуску. Найчастіше використовується спуск із постійним кроком чи спосіб якнайшвидшого спуску.

Алгоритм зворотного поширення помилки ось у чому. Ми будемо розглядати мережу з n вхідними нейронами, k прихованими та m вихідними. Вагу між вхідним нейроном i та прихованим нейроном j будемо позначати w_{ij}^{1}. Вагу між прихованим нейроном i та вихідним нейроном j називатимемо w_{ij}^{2}. Зміщення - θ кожного нейрона реалізовано як вагу додаткової компоненти вхідного сигналу. Вхідні ваги, таким чином розширюються за допомогою одиничної компоненти, і те саме робиться з вихідним вектором з прихованого шару.Вага між константою 1 та прихованим нейроном j називається w^{1},а вага між константою 1 та вихідним нейроном j визначається як w^{2} .

$k+1,\mathrm{j}$

$n+1,\mathrm{j}$

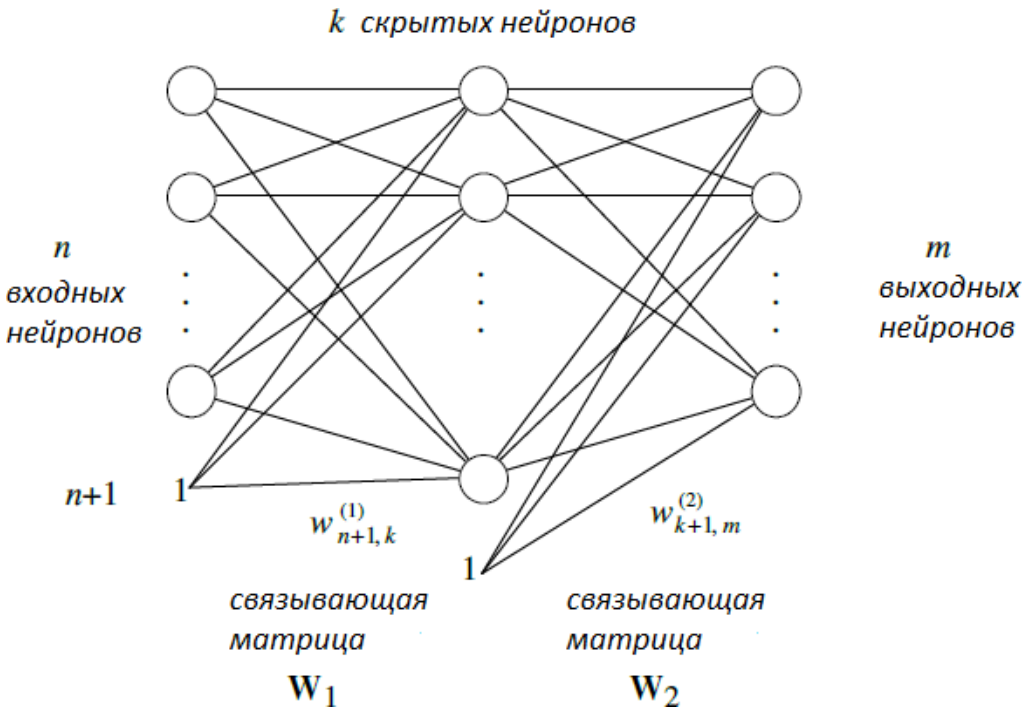

Рис. 2.7. Система позначень для нейронної мережі.

У нас є (+ 1) k ваг між вхідними та прихованими нейронами та (k + 1) між прихованими та вихідними нейронами. Позначимо (+1) k - мірну матрицю 1 з компонентами $w$ij1 для i-го рядка j-го стовпця.

Аналогічно, позначить (k + 1) - розмірну матрицю з компонентами $w$ij^2. Ми використовуємо такі позначення, щоб підкреслити, що останній рядок обох матриць містить усунення обчислених нейронів. Матриця ваги без цього останнього рядка буде необхідна на етапі зворотного поширення. N-вимірний вхідний вектор o^ = (o1, o2 … , on). Зважена сума вхідних сигналів нейрона на j-му прихованому шарі дорівнює:

$$net_{\mathbf{j}} = \sum_{\mathrm{i}=1}^{n+1} w_{\mathbf{ij}}^{(1)} o_i, \qquad (2.12)$$

Функція активації є сигмоїдальною та вихід $o_{\mathbf{j}}^{(1)}$ цього нейрона виглядає так:

$$o_j^{(1)} = s\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} o_i\right), \quad (2.13)$$

Значення суматорів всіх нейронів у прихованому шарі може бути обчислено за допомогою матричного перемножения o на W_1. Вектор $o^{(1)}$, компонентами якого є виходи прихованих нейронів, задається таким чином:

$$o^{(1)} = s(oW_1), \quad (2.14)$$

Користуючись угодою о використанні сигмоідальної функції к кожному компоненті вектора аргумента. Суматори нейронів у вихідному шарі обчислюється з використанням розширеного вектора $o^{(1)} = (o^{(1)}, \dots, o^{(1)}, 1)$.

Виходом мережі є m - розмірний вектор $o^{(2)}$:

$$o^{(2)} = s(o^{(1)}W_2), \quad (1.4.3.10)$$

Формули можуть бути узагальнені для будь-якої кількості прихованих шарів.

Для того, щоб спростити міркування, ми говоритимемо про одну навчальну пару (o, t). Це відповідає послідовному методу навчання мережі.

Алгоритм можна розкласти на чотири етапи:

1. Обчислення прямого ходу;
2. Зворотне поширення вихідного шару;
3. Зворотне поширення прихованого шару;
4. Оновлення ваг.

Алгоритм зупиняється, коли значення функції помилки стає досить малим.

Перший крок: обчислення прямого поширення.

Вектор o представлений мережі. Вектори $o^{(1)}$ та $o^{(2)}$ обчислені та збережені. Також розраховані похідні функції активації та збережені для

кожного нейрона.

Другий крок: зворотне поширення у вихідному шарі.

Ми шукаємо перший набір приватних похідних $\partial E/\partial w$. Шлях зворотнього поширення від виходу мережі до вихідного шару j, показаний на малюнку 2.8.

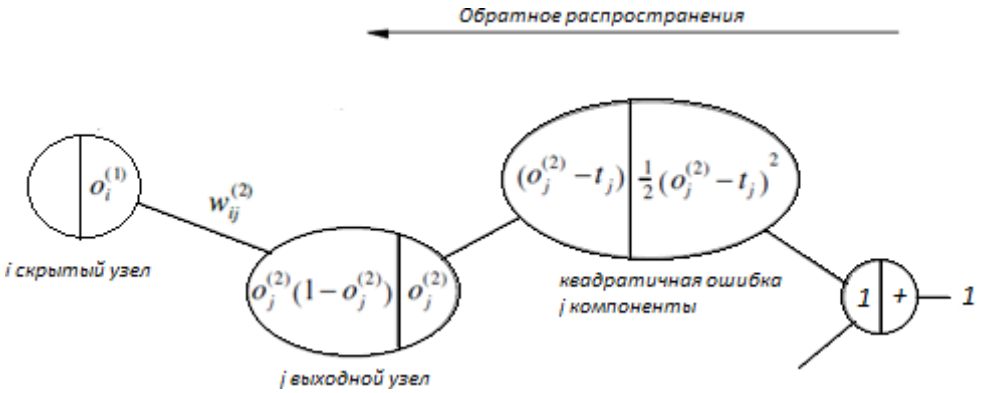

Рис. 2.8. Шлях зворотного розповсюдження до вихідного нейрона j

Помилка зворотного розповсюдження δ обчислюється за формулою:

$$\delta_j^{(2)} = o_j^{(2)}(1-o_j^{(2)})(o_j^{(2)}-t_j), \qquad (2.15)$$

приватні похідні обчислюються таким чином:

$$\frac{\partial E}{\partial w_{Ij}^{(2)}} = [o_j^{(2)}(1-o_j^{(2)})(o_j^{(2)}-t_j)]\, o_i^{(1)} = \delta_j^{(2)} o_i^{(1)}, \qquad (2.16)$$

Необхідно пам'ятати, що на останньому кроці ми вважаємо ваги $w_{ij}^{(2)}$ простими змінними, а їх виходи $o_i^{(1)}$ постійними.

На вхідному кінці зв'язку з вагою w_{ij} ми маємо $o_i^{(1)}$ та на вихідному кінці помилку зворотнього розповсюдження $\delta_j^{(2)}$.

Третій крок: зворотне поширення у прихованому шарі.

Тепер ми вважаємо приватні похідні $\partial E / \partial w_{ij}^{(1)}$

Кожен нейрон j в прихованому шарі приєднаний до нейрону q вихідного шару з вагою $w_{jq}^{(2)}$ для *q=1, .., m*. Помилка зворотного поширення до нейрона j у прихованому шарі повинна бути розрахована з урахуванням усіх можливих зворотних зв'язків, як

показано малюнку . 2.3.4.6. Помилка зворотного поширення має вигляд:

$$\delta_j^{(1)} = o_j^{(1)} (1 - o_j^{(1)}) \sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)}, \qquad (2.17)$$

Таким чином, приватну похідну ми знаходимо за формулою:

$$\frac{\partial E}{\partial w_{Ij}^{(1)}} = \delta_j^{(1)} o_i, \qquad (2.18)$$

Помилка зворотного розповсюдження розраховується так само для всіх прихованих шарів, і вираз для приватної похідної E зберігає таку ж аналітичну форму.

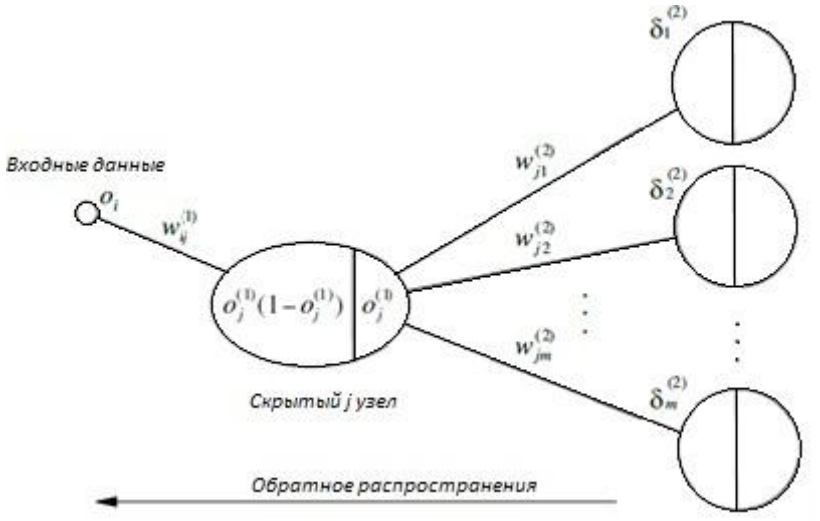

Рисунок 2.9. Всі шляхи до вхідної i-ої сторони.

Четвертий крок: зміна ваг.

Після обчислення всіх приватних похідних мережі ваги змінюються у напрямку антиградієнта. Коефіцієнт навчання γ визначає довжину кроку виправлення. Поправки ваги визначаються за формулою:

$$\Delta w_{ij}^{(2)} = -\gamma o_i^{(1)} \delta_j^{(2)}, \qquad i = 1, \dots, k+1 \quad = 1, \dots, , \tag{2.19}$$

або

$$\Delta w_{ij}^{(1)} = -\gamma o_i \delta_j^{(1)}, \qquad i = 1, \dots, n+1 \quad = 1, \dots, k, \tag{2.20}$$

де $o_{n+1} = o_{n+1} = 1$.

Дуже важливо додати поправки ваги тільки після того, як помилки зворотного розповсюдження будуть обчислені для всіх нейронів у мережі.

2.3.5. Проблеми навчання мережі

Локальний мінімум.

Насправді існують деякі особливості поверхні відгуку функції суми квадратів помилок, такі як " яри " і " плато " , що може бути труднощі мінімізації цільової функції. Наприклад, дві функції обробки помилок, показані малюнку 12, немає локальних мінімумів . Тим не менш, функція з лівого боку, як очікується, буде складнішою для оптимізації з методом градієнтного спуску.

По осі абсцис відкладено значення одного параметра, а осі ординат відповідає функція помилки. Незважаючи на те, що жодна з цих функцій не містить локальних мінімумів, функція зліва буде менш придатною для градієнтної

оптимізації спуску через плоскі ділянки.

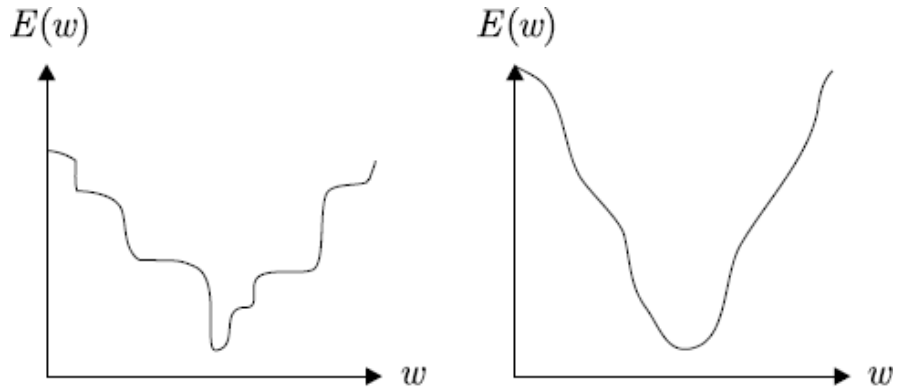

Рис. 2.10 Дві функції помилки однієї розмірності.

Узагальнення та перенавчання.

Властивість узагальнення показує те, наскільки добре мережа виконує свої функції нових даних, які брали участь у процесі навчання. Модель мережі навчається на заздалегідь відомих навчальних даних та узагальнення відповідає очікуваній ефективності моделі за новими даними для перевірки.Математично, мета навчання для багатошарового персептрона може бути сформульована як мінімізація функції "вартості":

$$E_{true} = \int_{x,d} e(f(x.\, w), d)p(x, d)dxdd, \qquad (2.21)$$

де e - локальна функція вартості, f - функція, що реалізується за допомогою багатошарового персептрона, x - вхідний сигнал мережі, d - це очікуваний, w

- відповідні ваги мережі, p - ймовірність розподілу. Метою навчання є оптимізація параметрів w, що зводить функцію E_{true} до мінімуму:

$$\hat{w} = argmin_{w} \int_{x,d} e(f(x.w), d)p(x, d)dxdd, \quad (2.22)$$

E_{true} - помилка узагальнення, тобто. очікувана продуктивність багатошарового персептрона на нових заздалегідь невідомих прикладах вхідних даних, обраних довільно з $p(x, d)$. На практиці $p(x, d)$ невідома величина. Натомість, нам дано навчальні пари даних, і функція, яку необхідно мінімізувати. Вона називається емпіричною функцією або функцією помилки навчання:

$$E = \sum_{p=1}^{Np} e(x_p, d_p), \quad (2.23)$$

Дуже важливе питання полягає в тому, щоб зрозуміти, наскільки добре модель мережі натренована для мінімізації функції E узагальнено (тобто наскільки мала функція E_{true}).

2.3.6. Проблема вибору кількості прихованих шарів та кількості нейронів у прихованому шарі

Створення архітектури нейронної мережі означає вказівку кількості шарів кожного типу та кількості нейронів у кожному з цих шарів.

Вхідний шар - один із найпростіших шарів, який має абсолютно будь-яка нейронна мережа без винятків. Кількість нейронів, що містяться в цьому шарі, повністю і однозначно визначається розмірністю вхідного сигналу. Важливо відзначити, що вхідний шар фактично не виконує жодних обчислювальних функцій і може вважатися шаром роздільників компонентів вхідного сигналу. Деякі конфігурації нейронних мереж додають додатковий вузол для зміщення.

Вибір числа нейронів для вихідного шару дуже подібний підходом для вхідного шару. Визначення кількості нейронів дуже простий. Воно повністю визначається розмірністю вихідного сигналу.

Існує кілька правил, які дозволяють встановити кількість шарів та розмір нейронного шару для обох вхідних та вихідних шарів. Основне питання архітектури багатошарового персептрона стосується прихованих шарів. Скільки прихованих шарів має бути у нейронній мережі? Добре, якщо завдання лінійно розділяється, тоді не потрібно якихось прихованих шарів взагалі. Один із проблем у цій темі, через яку існує консенсус, тобто. Різниця продуктивності від додавання додаткових прихованих шарів - ситуація, в якій підвищується продуктивність з додаванням другого (третього, четвертого, тощо) прихованого шару дуже малоймовірна. Один прихований шар достатній для переважної більшості завдань.

А ось для визначення числа нейронів у прихованому чи прихованих шарах існують емпірично-обґрунтовані правила. В цілому більшість покладаються на цей "оптимальний розмір прихованого шару, який ґрунтується на кількості нейронів у вхідному та вихідному шарі". Також вважається, що кількість нейронів у прихованому шарі має бути в кілька разів менше кількості навчальних прикладів, за умови наявності надмірності навчальних даних.

Підсумовуючи все вище сказане, для більшості завдань можна було б, ймовірно, отримати гідну продуктивність, встановивши конфігурацію прихованого шару, використовуючи лише два правила:

1) кількість прихованих шарі дорівнює одиниці;

2) кількість нейронів у цьому шарі є середнім між кількістю нейронів у вхідному та вихідному шарах і бути в кілька разів менше кількості навчальних прикладів, за умови наявності надмірності навчальних даних.

Для того, щоб нейронна мережа навчилася правильно на заданій вибірці даних із найбільшою точністю, необхідно підібрати цю кількість нейронів найбільш оптимально. Їх має бути небагато, але й достатньо для гарної апроксимації функції у просторі синоптичних ваг. У разі, якщо кількість нейронів у

прихованому шарі недостатньо, завдання навчання на навчальній множині не може бути вирішена. Якщо нейронів занадто багато, мережа може навчитися надмірно. При цьому фактично відбувається запам'ятовування різних шумів та похибок навчальних даних. При цьому процес навчання суттєво уповільнюється через більшу кількість вагових коефіцієнтів мережі. Тому налаштування оптимальної архітектури мережі застосовуються різні методи. Такі методи поділяються на дві групи: алгоритми скорочення та конструктивні алгоритми.

Висновки до розділу 2

В Спеціальному розділі були розглянути методи прогнозування. Було вивчено що таке нейронна мережа, для яких задач вона існує. Був розглянутий алгоритм навчання багатошарового персептрону, складності у виборі кількості прихованих шарів та кількості нейронів у прихованому шарі. Був проведений аналіз проблем застосування нейронних мереж.

3 ЕКСПЕРЕМЕНТАЛЬНО-АНАЛІТИЧНИЙ РОЗДІЛ

3.1 Прогнозування індексу Доу Джонса. Аналіз початкових даних

У цьому розділі ми застосуємо теорію, описану в попередніх параграфах, для формування математичного апарату для прогнозування Індексу Доу Джонса. Побудуємо власну нейронну мережу. Для цього нам будут потрібні деякі реальні дані, на основі яких, ми будемо проводити наше дослідження. Візьмемо початкові данні з сайту https://ru.investing.com/indices/us-30-historical-data з даними о значеннях індексу Доу Джонса за будь-який період. Ці початкові данні можна знайти у Додатку Г. За допомогою програми на Python приведемо початкові дані до належного виду Додаток Д. Ці данні будемо застосовувати як данні на вхід нашої нейронної мережі, саму нейронну мережу ми побудуємо в програмі C++ - Додаток Ж.

1) Побудуємо нашу модель на основі даних за кожен день у продовж чотирьох років. Після приведення початкових даних до належного для аналізу виду за допомогою Python отримуємо такі значення у файл:

27462 27492 27492 27674 … 32037
32861 32734 32650 32146 31999 32404

На основі цих даних побудуємо графік (Рисунок 3.1)

Рис. 3.1. Щоденні значення індексу Доу Джонса

3.2 Рішення. Прогнозування індексу Доу Джонса за допомогою нейронної мережі

Для рішення цієї задачі була розроблена програма в середовищі C++ (Додаток Ж). Програма представляє собою нейронну мережу, де перший шар нейронів - це початкові данні, кількість нейронів в першому шарі відповідає кількості значень індексу Доу Джонса які буде аналізувати програма; останній шар нейронів- це вихідні значення – прогнозовані значення нашого індексу, кількість нейронів в останньому шарі відповідає кількості прогнозованих значень.

Програма дає можливість користувачу обирати самому кількість прихованих шарів для мережі та кількість нейронів на кожному з них. Також користувач задає розмір навчальної множини та кількість прогонів. Ваги для першого прогону задаються довільно програмою, після першого проходження мережі отримаємо прогнозне значення та значення помилки, після цього мінімізуємо функцію та змінюємо ваги. На наступному прогоні ваги будуть такими як на попередньому, тому і помилка на наступних прогонах очікується меншою.

Структура мережі така, що кожен нейрон на попередньому шарі подає сигнал на кожен нейрон наступного шару. (Рисунок 3.2)

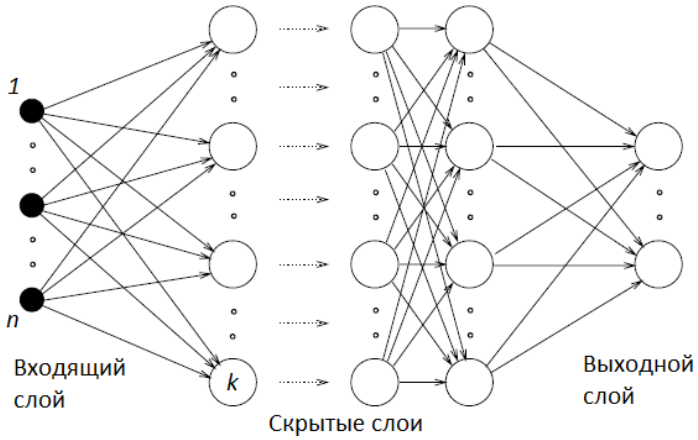

Рис. 3.2. Структура мережі

функцій активації, що використовується - це речова сигма - функція, що визначається виразом:

$$s_c(x) = \frac{1}{1 + e^{-cx}}, \qquad (3.1)$$

Спробуємо запустити програму для наших даних (Рисунок 3.3)

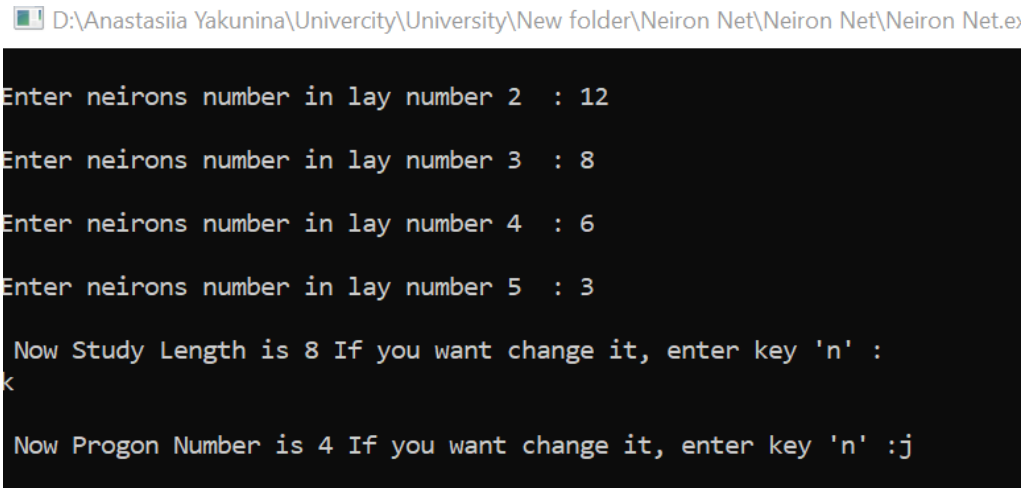

Рис. 3.3. Початок роботи програми

Результати програми проаналізуємо за допомогою Excel

Структура нашоЇ мережі:

```
Net has 5 layers
 In lay number 1 there are 12 neirons
 In lay number 2 there are 12 neirons
 In lay number 3 there are 8 neirons
 In lay number 4 there are 6 neirons
 In lay number 5 there are 3 neirons
```

Програма приймає на вхід 12 значень індексу на першому шарі, потім ці данні йдуть на другий шар з 12 нейронами, потім на 3й з 8 нейронами та на 4й з 6. Останній 5й шар має 3 нейрони отже прогнозувати ми будемо також 3 значення. Кількість прогонів дорівнює 15. Покажемо еволюцію помилки для кожного прогону (Таблиця 3.1).

Таблиця 3.1

Результат навчання

Evolution of error :									
Progon number 1 :									
42.3272	38.0503	33.6666	29.7362	27.5132	28.0176	28.0826	27.3678	25.1604	23.4009
22.4811	21.2825	19.5634	17.6596	16.2658	15.5273	15.2088	14.7734	14.3429	13.4076
12.3664	11.0852	9.56644	8.57797	7.72413	7.09203	6.30074	5.77988	5.53319	5.27459
Progon number 2 :									
10.7696	9.92488	8.98398	8.10816	7.72469	8.2302	8.59952	8.69012	8.21161	7.85704
7.79672	7.60528	7.16654	6.60535	6.22734	6.10897	6.16531	6.15874	6.14007	5.8606
5.5062	5.00403	4.35038	3.94407	3.59034	3.33849	2.98662	2.76842	2.69312	2.60611
Progon number 3 :									
5.90607	5.50396	5.02715	4.57113	4.40808	4.80385	5.11831	5.25468	5.01439	4.84755
4.87253	4.80585	4.56466	4.23027	4.01625	3.97854	4.06232	4.10077	4.12883	3.96644
3.74569	3.41201	2.96142	2.6871	2.44826	2.28173	2.03838	1.89158	1.84965	1.79822
Progon number 4 :									
4.34518	4.06123	3.71568	3.38157	3.27419	3.60793	3.87951	4.01074	3.83972	3.7257
3.7656	3.73064	3.55239	3.29548	3.1354	3.11865	3.20185	3.2476	3.28447	3.16249
2.99079	2.72336	2.35645	2.13533	1.94306	1.81045	1.61281	1.49516	1.46489	1.42649
Progon number 5 :									
3.60168	3.36863	3.0816	2.80265	2.71804	3.01503	3.25919	3.38232	3.24208	3.15083
3.19394	3.17128	3.02242	2.80334	2.66888	2.66016	2.73974	2.78625	2.82483	2.72239
2.57541	2.34282	2.02115	1.8284	1.66102	1.54632	1.37364	1.27161	1.2469	1.21493

Продовження табл. 3.1

Progon number 6 :									
3.16856	2.96348	2.70931	2.46158	2.38897	2.66195	2.88758	3.00389	2.88074	2.80184
2.8453	2.82867	2.69664	2.49988	2.38024	2.37536	2.45144	2.49721	2.53571	2.44465
2.31248	2.10137	1.80812	1.63302	1.48115	1.3775	1.22059	1.12829	1.10675	1.07854
Progon number 7 :									
2.88457	2.69716	2.46405	2.23645	2.17118	2.42726	2.63959	2.75048	2.63818	2.56696
2.60995	2.59674	2.4756	2.2936	2.18361	2.18085	2.25396	2.29868	2.33657	2.25294
2.13065	1.93417	1.66049	1.49748	1.35623	1.26011	1.1141	1.02847	1.00896	0.983215
Progon number 8 :									
2.68347	2.50826	2.28984	2.07635	2.01602	2.25954	2.46187	2.56844	2.46362	2.39763
2.43992	2.42886	2.31535	2.14387	2.04068	2.03919	2.10984	2.1535	2.19068	2.11229
1.99707	1.81122	1.5519	1.39772	1.26423	1.17359	1.03559	0.954834	0.93674	0.912724
Progon number 9 :									
2.53319	2.36693	2.15938	1.95635	1.89958	2.13338	2.3279	2.43097	2.33164	2.26944
2.31101	2.3014	2.19354	2.02995	1.93183	1.93116	1.99977	2.04245	2.07892	2.00443
1.89454	1.71679	1.46849	1.32106	1.19351	1.10704	0.975201	0.898179	0.881121	0.858393
Progon number 10 :									
2.41635	2.25697	2.0578	1.86287	1.80879	2.03483	2.22308	2.32325	2.22813	2.1688
2.20968	2.2011	2.0976	1.94019	1.84598	1.84588	1.91277	1.95458	1.99039	1.91891
1.81319	1.64184	1.40229	1.2602	1.13736	1.05417	0.927238	0.853168	0.836906	0.815174
Progon number 11 :									
2.32273	2.1688	1.97631	1.78785	1.73588	1.95558	2.13866	2.2364	2.14462	2.08754
2.12779	2.11997	2.01996	1.8675	1.77643	1.77673	1.84216	1.88319	1.91839	1.84933
1.74697	1.58081	1.34838	1.21064	1.09162	1.01111	0.888166	0.816499	0.800867	0.77993
Progon number 12 :									
2.2459	2.09641	1.90938	1.72622	1.67595	1.89035	2.0691	2.16478	2.07571	2.02046
2.06013	2.05289	1.95572	1.80734	1.71885	1.71944	1.78361	1.82396	1.85861	1.79153
1.69192	1.53007	1.30357	1.16944	1.05359	0.975297	0.855685	0.786013	0.770893	0.750606
Progon number 13 :									
2.18163	2.03584	1.85336	1.67461	1.62575	1.83565	2.01073	2.10462	2.01781	1.96406
2.00322	1.99643	1.90163	1.75668	1.67033	1.67114	1.73422	1.77396	1.80812	1.74268
1.6454	1.48718	1.26569	1.13461	1.02145	0.945023	0.82823	0.760245	0.74555	0.725804
Progon number 14 :									
2.12701	1.98434	1.80572	1.63073	1.58304	1.78909	1.96098	2.05333	1.96841	1.91593
1.95462	1.9482	1.85541	1.71337	1.62885	1.62983	1.69195	1.73114	1.76487	1.70082
1.60552	1.4504	1.23322	1.10476	0.993898	0.91907	0.804698	0.738159	0.723823	0.704536
Progon number 15 :									
2.07998	1.93998	1.76468	1.59292	1.54623	1.74892	1.91805	2.00903	1.92575	1.87434
1.91261	1.90648	1.81542	1.6759	1.59295	1.59406	1.65534	1.69404	1.72736	1.66452
1.57092	1.4185	1.20506	1.07886	0.97	0.896558	0.784289	0.719006	0.704979	0.686085

Відобразим еволюцію навчання нашою мережі на графіку (Рисунок 3.4)

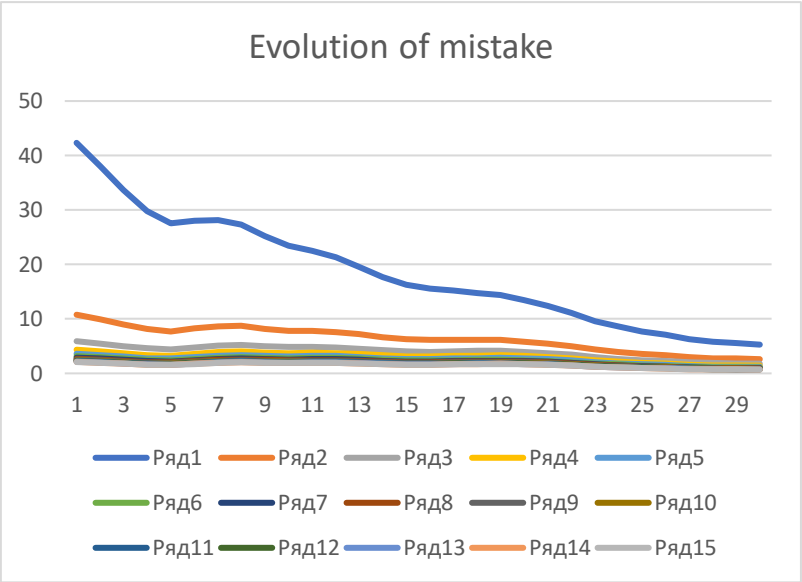

Рис. 3.4. Еволюція помилки за прогонами

Отже, з цього графіку можна побачити як зменшується кожна помилка з кожного прогону за кількістю навчань.

Також проілюструємо зміну результуючої помилки з кожного прогону

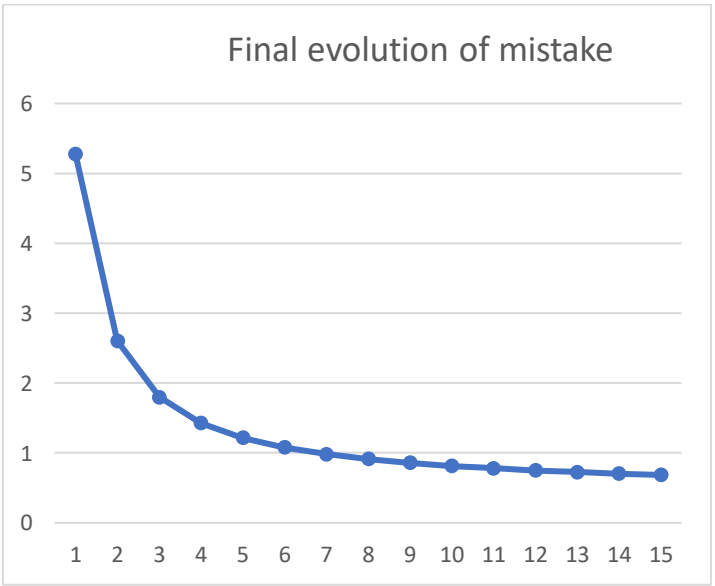

Рис. 3.5. Зменшення результуючої помилки з кожного прогону

Як можна побачити з графіка на першому прогоні ми мали значення помилки десь 5.3, а на останньому мережа навчилась та помилка стала близькою до 0.

Input data :
28267 28239 28376 28455 28551 28515 28621 28645 28462 28538 28868 28634
Right result :
28703 28583 28745
Prognoz :
28532.3 28702.9 28604.2

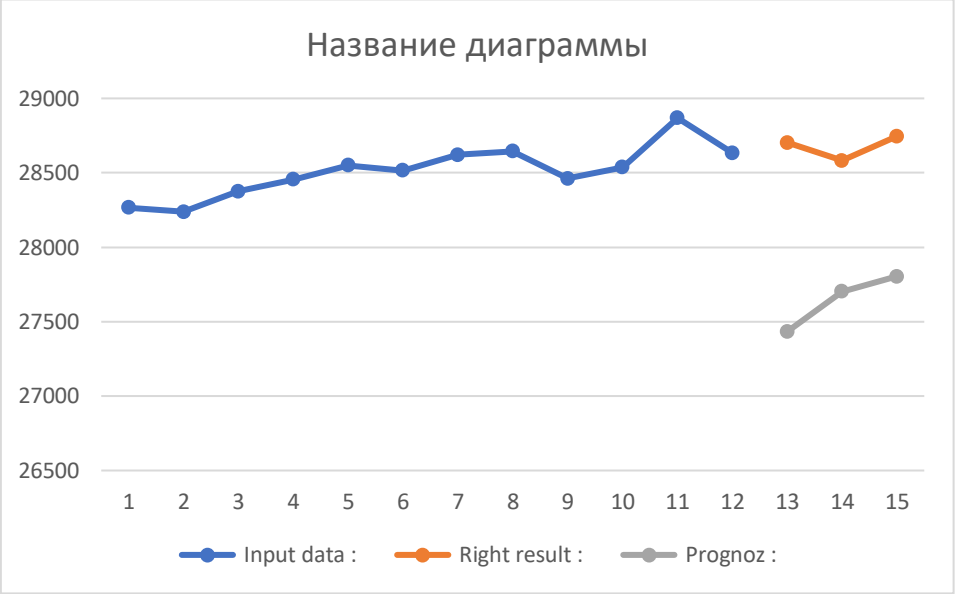

Рис. 3.6. Зображення прогнозного та очікуваного значення

Спробуємо змінити структуру нашої мережі та подивимося як це вплине на наш результат.

```
Enter lay number : 7

Enter neirons number in lay number 1  : 20

Enter neirons number in lay number 2  : 5

Enter neirons number in lay number 3  : 3

Enter neirons number in lay number 4  : 3

Enter neirons number in lay number 5  : 3

Enter neirons number in lay number 6  : 3

Enter neirons number in lay number 7  : 3

 Now Study Length is 5 If you want change it, enter key 'n' :k

 Now Progon Number is 4 If you want change it, enter key 'n' :j

 Error = 0.682209
 Error = 0.661662
 Error = 0.651385
 Error = 0.643317
 Error = 0.647306
```

Рис. 3.7. Нова мережа

```
Prognoz Result :   27584.8  27601.8  27725.5
Right Result :   27881  27911  28132
If you want to calculate it one more, enter key 'y' :
```

Рис. 3.7. Нова мережа, результат

Можна побачити, що коли ми збільшили кількість шарів наша помилка стала меншою, прогнозні значення мають більшу сходимість:

Prognoz Result : 27584.8 27601.8 27725.5

Right Result : 27881 27911 28132

Усі приклади вище працювали з даними, графік яких мав певну тенденцію до зростання (Рисунок 3.8) тому прогнозування з використанням нейронної мережі працювало досить успішно. Що якщо ми спробуємо зробити прогноз на іншому відрізку на графіку, тоді коли значення спочатку зменшуються, а потім зростають (Рисунок 3.9)

Рис. 3.8. Інтервал значень індексу для попередніх розрахунків

Рис. 3.9. Інтервал для наступного навчання мережі

Спробуємо розрахувати прогнозні значення для відрізку, що на рисунку 3.9. Кількість шарів залишимо 7 як і в попередньому вдалому прикладі, кількість прогнозованих значень теж залишимо 3. (Рисунок 3.10)

```
Enter lay number : 7

Enter neirons number in lay number 1  : 70

Enter neirons number in lay number 2  : 10

Enter neirons number in lay number 3  : 8

Enter neirons number in lay number 4  : 5

Enter neirons number in lay number 5  : 4

Enter neirons number in lay number 6  : 3

Enter neirons number in lay number 7  : 3

 Now Study Length is 8 If you want change it, enter key 'n' :n

 Enter new Study Lengh : 20

 Now Progon Number is 4 If you want change it, enter key 'n' :h

 Error = 1.01273
 Error = 0.963441
 Error = 0.919099
 Error = 0.849044
 Error = 0.789049
 Error = 0.741146
 Error = 0.745849
 Error = 0.871984
 Error = 0.954379
 Error = 1.02012
 Error = 0.953457
 Error = 0.92538
 Error = 0.907058
 Error = 0.929615
 Error = 0.934489
 Error = 0.930915
 Error = 0.955343
 Error = 0.969662
 Error = 1.03741
```

Рис. 3.10. Мережа для відрізка без чіткого тренду

```
Prognoz Result :   28255.2  27575.8  27916.4
Right Result :   25015  25595  25812
```

Рис. 3.10. Мережа для відрізка без чіткого тренду. Результат

Побудуємо графік для ілюстрації зміни помилки під час машинного навчання (Рисунок 3.11)

Рис. 3.11.Помилка під час навчання мережі

Як можна побачити на графіку значення помилки то зменшується то збільшується. Прогнозні значення, які ми отримуємо дорівнюють:

Prognoz Result : 28255.2 27575.8 27916.4

Right Result : 25015 25595 25812

Прогнозні та правильні значення суттєво відрізняються. Отже можна зробити висновок, що ця нейронна мережа не готова до роботи з даними, які не мають чіткої лінії тренду.

3.3. Рішення. Прогнозування індексу Доу Джонса за допомогою методів прогнозування числових рядів метод Рухомого(ковзного) середнього та метод Зваженого середнього

Спробуємо застосувати методи прогнозування числових рядів для нашого прикладу. Для цього візьмемо данні курсу індексу Доу Джонса за 60 днів для прогнозу скористуємось інтервалом без різких змін ряду (рисунок 3.12)

Рис. 3.12. Обраний інтервал для прогнозування

За методом рухомого(ковзного) середнього та методом зваженого середнього знаходимо прогнозні значення

Таблиця 3.2.

Прогнозування Метод рухомого (ковзного) середнього та Метод зваженого середнього

Індекс Доу Джонса	Метод рухомого(ковзного) середнього	Метод зваженого середнього	
27462			
27492			
27492			
27674	27482.0	27486.0	
27681	27552.7	27583.0	
27691	27615.7	27641.1	
27691	27682.0	27684.6	
27783	27687.7	27689.0	
27781	27721.7	27737.0	
28004	27751.7	27763.6	
28036	27856.0	27892.9	
27934	27940.3	27975.4	
27821	27991.3	27978.6	
27766	27930.3	27897.9	
27875	27840.3	27816.1	
28066	27820.7	27831.5	
28121	27902.3	27948.7	

Продовження табл. 3.2.

28164	28020.7	28055.3	
28051	28117.0	28131.5	
27783	28112.0	28098.9	
27502	27999.3	27939.6	
27649	27778.7	27696.1	
27677	27644.7	27631.7	
28015	27609.3	27633.6	
27909	27780.3	27840.4	
27881	27867.0	27894.4	
27911	27935.0	27916.2	
28132	27900.3	27901.6	
28135	27974.7	28015.5	
28235	28059.3	28089.3	
28267	28167.3	28184.4	
28239	28212.3	28231.0	
28376	28247.0	28246.6	
28455	28294.0	28313.1	
28551	28356.7	28388.1	
28515	28460.7	28487.2	
28621	28507.0	28513.8	
28645	28562.3	28575.2	
28462	28593.7	28611.8	
28538	28576.0	28548.7	
28868	28548.3	28536.6	
28634	28622.7	28687.8	
28703	28680.0	28685.0	
28583	28735.0	28715.3	
28745	28640.0	28629.2	
28956	28677.0	28688.0	
28823	28761.3	28818.1	
28907	28841.3	28847.3	
28939	28895.3	28891.6	
29030	28889.7	28906.2	
29297	28958.7	28978.1	
29348	29088.7	29145.3	
29196	29225.0	29269.1	
29186	29280.3	29261.8	
29160	29243.3	29221.4	
28989	29180.7	29175.0	
28535	29111.7	29079.7	
28722	28894.7	28796.2	
28734	28748.7	28719.3	
28859	28663.7	28690.6	
	28771.7	28794.1	

28256

Очікуване значення індексу:

Метод рухомого(ковзного) середнього: 28771.66667

Метод зваженого середнього: 28794.1

Продемонструємо на графіку різницю початкових значень індексу Доу Джонса та прогнозованих значень числового ряду методом рухомого(ковзного) середнього та методом зваженого середнього (Рисунок 3.13)

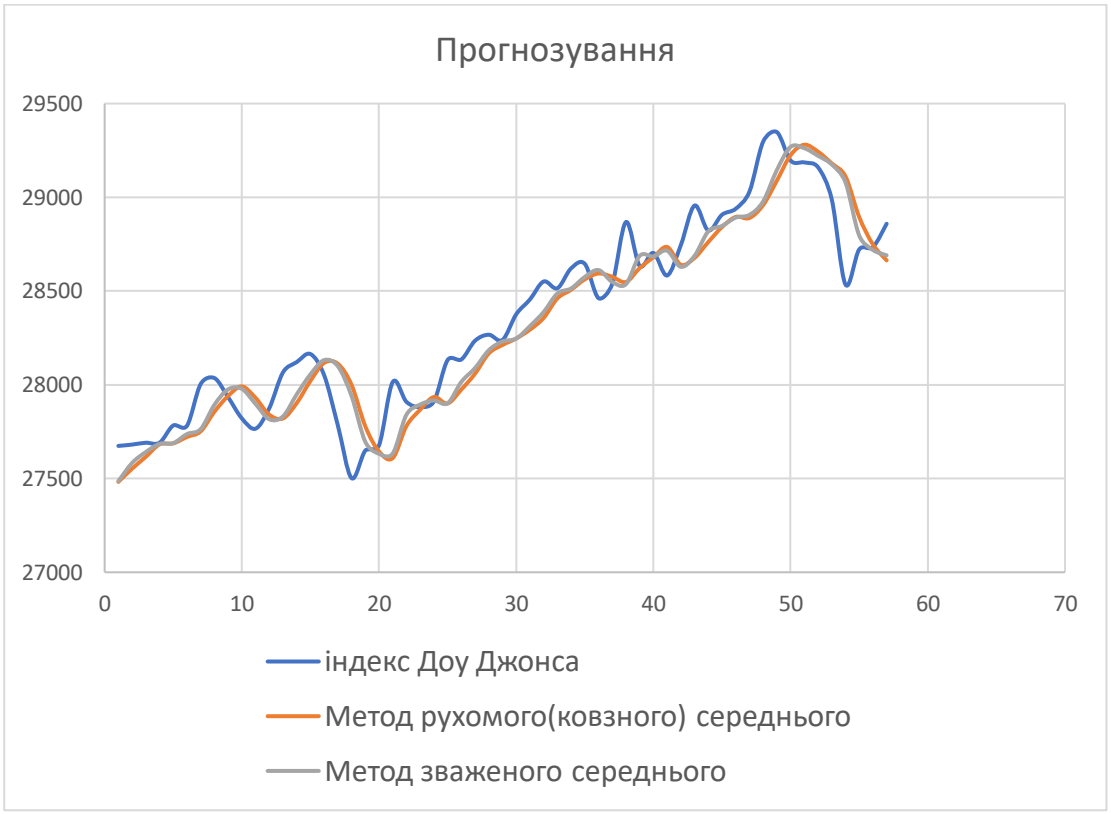

Рис. 3.13. метод рухомого(ковзного) середнього, метод зваженого середнього та початковий ряд

З графіка 3.13 можна побачити що прогнозні значення набувають меншого значення ніж очікується, це коїться за причиною різкого спаду а потім підйому величини індексу. Метод рухомого(ковзного) середнього та метод зваженого середнього не можуть швидко адаптуватися до цих змін.

Висновки до розділу 3

У цьому розділі була розроблена нейронна мережа за допомогою програми C++, алгоритм якої дозволяє користувачу самому обирати структуру мережі: кількість даних на вхід, кількість прихованих шарів та кількість нейронів в кожному з них. Виявилося, що програма алгоритм працює досить успішно, в процесі навчання помилка зменшується, та прогнозовані значення дуже схожі на реальні значення індексу Доу Джонса. Однак алгоритм не дуже чітко працює з даними, що не мають якоїсь тенденції . Також ми спробували розрахувати майбутнє значення індесу методом Рухомого(ковзного) середнього та методом Зваженого середнього.

ВИСНОВОК

У даній кваліфікаційній роботі була розглянута задача формування нейронної мережі. Перш за все, ми ознайомилися з фінансовою теорією, та визначили що таке індекс Доу Джонса. Дослідили алгоритми різних методів прогнозування числових рядів та пободували власну модель нейронної мережі. Вхідними даними був числовий ряд із значеннями індексу Доу Джонса за кожен день. На обраних інтервалах ми навчали нашу нейронну мережу. Були використані різні варіації змісту мережі, з різною кількістю прихованих шарів, нейронів в кожному з них, були подані різні розміри початкових даних, та були отримані та проаналізовані результати

У процесі досягнення поставлених раніше цілей, ми встановили залежність характеру поданих даних на вхід та отриманого результату, переконавшись в ефективності використання нейронних мереж.

Всі поставлені завдання були розкриті і виконані відповідно з обраної темою і поставленою метою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://alpari.com/ru/beginner/glossary/djia/
2. https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0
3. https://ru.investing.com/indices/us-30-historical-data
4. Каллан Р. Нейронні мережі короткий довідник. // Віл'ямс, 2017. – 288 с
5. Айвозян С.А., Методи економетрики. - М .: Економіст, 2010.-560с.Артюхов С.В., Базюкіна О.А., Корольов В.Ю., Кудрявцев А.А. Модель оптимального ціноутворення, заснована на процесах ризику з випадковими преміями. // Системи і засоби інформатики. Спеціальний випуск. - М .: ІПІРАН, 2005.-873с.
6. Пересада А.А. Інвестиційний процес в Україні. – К., «Видавництво Лібра» ТОВ, 1998р., – 392 с.
7. Покропивний В.М. Економіка підприємств: формування і регулювання фінансових інвестицій // http://library.if.ua/book/1/57.html
8. Савчук В.П. Оптимізація фондового портфелю // http://www.management.com.ua/finance/fin013.html
9. Управління портфелем інвестицій // http://vuzlib.net/invest_D/8.html
10. Чумаченко М.Г. Евристичні прийоми дослідження // http://ebk.net.ua/Book/Book.Ek.Analiz/part3.5.htm.
11. Берндт Е., Практика економетрики: класика і сучасність. -М .: Фінанси і статистика. 2009. - 414 с.
12. Бланк І.А. Стратегія і тактика управління фінансами. - Київ: ІТЕМлтд, АДЕФ-Україна, 2008. - 345 с.
13. Ван Хорн Дж.К. Основи управління фінансамі.- М .: Фінанси і статистика, 2009. - 240 с.
14. Доугерті К., Введення в економетрику. - М .: Справа, 2007. - 310 с.

15. Єфімова О.В. Фінансовий аналіз. - М .: Бухгалтерський облік. 2008. - 544 с.
16. Schmidhuber, J. Deep Learning in Neural Networks: An Overview // Neural Networks. – 2015. – Vol. 61. – P. 85–117.
17. Bengio, Y. Deep Learning / Y. Bengio, Y. LeCun, G. Hinton // Nature. – 2015. – Vol. 521. – P. 436–444.
18. Hochreiter, S. Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural Computation. – 1997. – Vol. 9, No. 8. – P. 1735-1780.
19. Gers, F. Learning precise timing with LSTM recurrent networks / F. Gers, N. Schraudolph, J. Schmidhuber // Journal of Machine Learning Research. – 2002. – Vol. 3. – P. 115—143. 182
20. Generative Adversarial Networks [Electronic resource] / [I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Sh. Ozair, A. Courville, Aaron, J. Bengio]. – Access mode: https://arxiv.org/pdf/1406.2661.pdf
21. Субботін, С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей : монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.
22. Ширяєв А.Н. Основи стохастичною фінансової математики. Теорія. - М .: Фазис, 1998.-506С.
23. Demuth, H. Neural Network Toolbox for use with Matlab: user’s guide / H. Demuth, M. Beale. – Natick: Mathworks Inc, 1997. – 700 p.
24. .Sumathi, S. Computational intelligence paradigms: theory and applications using Matlab / S. Sumathi, S. Paneerselvam. – Boca Raton: CRC Press, 2010. – 851 p.
25. How Does the Brain Works? [Електронний ресурс] / Sandra Blakeslee // The New York Times – Nov. 11, 2003 – Режим доступу до ресурсу: https://www.nytimes.com/2003/11/11/science/how-does-the-brain-work.html.
26. Artificial Neural Network (ANN) in Machine Learning [Електронний ресурс] /–Режим доступу до ресурсу:

https://www.datasciencecentral.com/profiles/blogs/artificial-neural-network-ann-inmachine-learning.

27. Yoon Kim Convolutional Neural Networks for Sentence Classification // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) – Doha, Qatar. – October 25-29, 2014. – P. 1746 – 1751. 72
28. Xiang Zhang Text Understanding from Scratch / Xiang Zhang, Yann LeCun // Computer Science Department, Courant Institute of Mathematical Sciences, New York University – New York, USA. – Apr 4, 2016.
29. Wen-tau Yih Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base / Wen-tau Yih, Ming-Wei Chang, Xiaodong He, Jianfeng Gao // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing – Beijing, China. – July 26-31, 2015. – P. 1321 – 1331.
30. Ф. Уоссермен. Нейрокомпьютерна техника

ДОДАТКИ

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Назва	Кількість	Примітки
1				
2		Документація		
3				
4	САУ.КР.22.17.ПЗ	Пояснювальна записка	85	Формат А4
5				
6	САУ.КР.22.17.ПЗ	Демонстраційні матеріали	10	Презентація на CD-R
7				
8	САУ.КР.22.17.ПЗ	Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.22.17.ДА.ПЗ			
Змін.	Аркуш	№ докум.	Підпис	Дата				
Розроб.		Водяницька			Матеріали кваліфікаційної роботи	Літ.	Аркуш	Аркушів
Керівн.		Ус						
Керівн. Сп. Р.		Ус				НТУ “ДП” 12; 124м-21		
Н.контр.		Ус						
Охорона праці								
Зав. каф.		Желдак						

ДОДАТОК Б

Відгук
на кваліфікаційну роботу магістра

студентки групи *124м – 21 – 1*
спеціальності *124 Системний аналіз*
Якуніної Анастасії Євгенівни

Тема кваліфікаційної роботи: «Застосування нейронних мереж в задачі прогнозування індексу Доу Джонса»

Обсяг кваліфікаційної роботи: пояснювальна записка 85 с., 23 рис., 2 табл., 7 додатків, 30 джерел.

Мета кваліфікаційної роботи: дослідження можливості застосування нейронної мережі до прогнозування Індекса Доу Джонса.

Актуальність теми обумовлюється реальною практичною задачею, розв'язанню якої присвячена робота та можливістю використання запропонованого підходу і програмного забезпечення для прогнозування значень інвестиційних показників.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 124 Системний аналіз, оскільки для своєї реалізації передбачає використання математичних методів та інформаційні технологій аналізу, моделювання, прогнозування, проектування та прийняття рішень.

Поставлені і виконані в кваліфікаційній роботі завдання відповідають вимогам, що висуваються до кваліфікаційних робіт ступеня магістра спеціальності 124 – системний аналіз.

Оригінальність наукових рішень полягає у використанні нейронних мереж до задачі пронозування інфестиційних показників.

Практичне значення результатів кваліфікаційної роботи полягає у можливості застосування розробленого програмного забезпечення до розв'язання задачі прогнозування індексу Доу Джонса на основі попередніх даних.

Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами, що висуваються до кваліфікаційних робіт бакалаврів спеціальності 124 – системний аналіз. Роботу виконано самостійно, відповідно до завдання та у повному обсязі. Під час виконання роботи Якуніна А. Є. проявила самостійність, ініціативу, відмінне знання предметної області та володіння методами аналізу і прогнозування.

Зауважень до роботи не маю.

Кваліфікаційна робота в цілому заслуговує оцінки: , а її автор Якуніна Анастасія Євгенівна заслуговує присвоєння освітньої кваліфікації «магістр з системного аналізу».

Керівник кваліфікаційної роботи магістра,
к.ф.-м.н, доцент,
професор кафедри системного аналізу
і управління С.А. Ус

ДОДАТОК В

Рецензія
на кваліфікаційну роботу магістра
студентки групи *124м – 21 – 1*
спеціальності *124 Системний аналіз*
Якуніної Анастасії Євгенівни

Тема кваліфікаційної роботи: «Застосування нейронних мереж в задачі прогнозування індексу Доу Джонса»

Обсяг кваліфікаційної роботи: пояснювальна записка 85 с., 23 рис., 2 табл., 7 додатків, 30 джерел.

Висновок про відповідність кваліфікаційної роботи завданню та освітньо-професійній програмі спеціальності: зміст пояснювальної записки відповідає ОП «Системний аналіз», повнота і глибина задач достатня для кваліфікаційної роботи магістра

Загальна характеристика кваліфікаційної роботи, ступінь використання нормативно-методичної літератури та передового досвіду

У кваліфікаційні роботі проведено аналіз предметної області і методів прогнозування, які застосовуються при дослідження фінансового ринку, розроблено програмний продукт для побудови штучної нейронної мережі і розв'язання задачі прогнозування індексу Доу Джонса, досліджено вплив структури мережі на результат прогнозування.

Позитивні сторони кваліфікаційної роботи:

- Проведено аналіз методів прогнозування,
- Розроблено програмний продукт, який дозволяє будувати штучну нейронну мережу визначеної структури і здійснювати прогнозування індексу Доу Джонса.
- Проведено аналіз залежності якості прогнозу від структури мережі.

Основні недоліки кваліфікаційної роботи:

Опис предметної області і огляд літератури є дуже стислим.

Кваліфікаційна робота в цілому заслуговує оцінки:

З урахуванням висловлених зауважень її автор, Якуніна Анастасія Євгенівна, **заслуговує присвоєння освітньої кваліфікації** «магістр з системного аналізу».

Рецензент,

ДОДАТОК Г

Данні індексу Доу Джонса

Дата	Ціна	Откр.	Макс.	Мін.	Об’єм	Зм. %
02.12.2022	34.428,95	34.265,45	34.481,12	34.042,46	279,59M	**+0.09%**
01.12.2022	34.396,53	34.533,59	34.581,98	34.130,18	331,32M	**-0.55%**
30.11.2022	34.587,46	33.795,43	34.587,46	33.583,93	466,31M	**+2.18%**
29.11.2022	33.850,48	33.847,80	33.933,49	33.662,45	271,10M	**0.00%**
28.11.2022	33.849,66	34.185,42	34.303,88	33.799,41	286,59M	**-1.45%**
25.11.2022	34.346,83	34.213,04	34.385,89	34.203,36	123,65M	**+0.44%**
23.11.2022	34.195,11	34.091,57	34.262,07	34.004,64	225,79M	**+0.28%**
22.11.2022	34.098,03	33.810,05	34.112,84	33.810,05	268,49M	**+1.18%**
21.11.2022	33.700,87	33.770,81	33.862,64	33.559,60	339,60M	**-0.14%**
18.11.2022	33.747,14	33.606,59	33.827,83	33.541,07	296,37M	**+0.60%**
17.11.2022	33.547,37	33.277,00	33.615,82	33.245,78	301,53M	**-0.03%**
16.11.2022	33.556,60	33.554,93	33.682,39	33.521,23	288,59M	**-0.11%**
15.11.2022	33.594,17	33.804,97	33.984,90	33.321,26	380,64M	**+0.17%**
14.11.2022	33.537,16	33.645,97	33.963,62	33.534,27	339,05M	**-0.63%**
11.11.2022	33.749,18	33.797,75	33.815,82	33.394,49	418,43M	**+0.11%**
10.11.2022	33.712,21	33.263,91	33.727,73	33.167,82	465,42M	**+3.68%**
09.11.2022	32.514,27	33.004,47	33.062,52	32.479,06	344,56M	**-1.95%**
08.11.2022	33.162,28	32.934,56	33.354,34	32.831,46	324,65M	**+1.01%**
07.11.2022	32.829,18	32.484,25	32.895,34	32.426,08	320,13M	**+1.31%**

Щоденні:

"04.11.2022","32.404,80","32.265,01","32.610,60","31.938,92","409,62M","1,27%"
"03.11.2022","31.999,34","31.985,05","32.185,71","31.728,85","345,46M","-0,46%"
"02.11.2022","32.146,77","32.576,28","33.070,28","32.142,98","384,07M","-1,54%"
"01.11.2022","32.650,83","32.927,61","32.975,35","32.486,84","312,67M","-0,26%"
"31.10.2022","32.734,40","32.672,03","32.883,86","32.586,93","369,53M","-0,39%"
"28.10.2022","32.861,34","32.204,31","32.888,75","32.158,38","485,69M","2,57%"
"27.10.2022","32.037,83","32.062,14","32.387,35","31.994,43","356,82M","0,62%"
"26.10.2022","31.840,10","31.738,44","32.171,98","31.738,44","420,19M","0,01%"
"25.10.2022","31.837,20","31.463,65","31.875,34","31.431,09","327,06M","1,07%"
"24.10.2022","31.500,61","31.251,86","31.602,39","31.165,23","331,04M","1,34%"

……

"11.11.2019","27.691,49","27.580,66","27.714,39","27.517,67","204,62M","0,04%"

"08.11.2019","27.681,24","27.686,20","27.694,95","27.578,97","222,79M","0,02%"

"07.11.2019","27.674,80","27.590,16","27.774,67","27.590,16","263,58M","0,66%"

"06.11.2019","27.492,56","27.502,74","27.526,05","27.407,81","243,19M","0,00%"

"05.11.2019","27.492,63","27.500,23","27.560,36","27.453,55","293,96M","0,11%"

"04.11.2019","27.462,11","27.402,06","27.517,58","27.402,06","273,03M","0,42%"

ДОДАТОК Д

Лістинг програми на Python для приведення початкових даних до належного виду

```
def PrepareDataForEconomic(InputFilePath, OutputFilePath):
    InputFile=open(InputFilePath,'r')
    OutputFile=open(OutputFilePath,'w')
    if InputFile==None or OutputFile==None:
        return False
    LinesList=[]

    for Line in InputFile:
        LinesList.append(Line)
    for LineNumber in range(len(LinesList)-1, -1, -1):
        WordsList=LinesList[LineNumber].split(',')
        Word=WordsList[1]
        Word=Word.replace("","").replace(".","")
        OutputFile.write(Word+'\t')
    InputFile.close()
    OutputFile.close()



InputFilePath=r'D:\Stocks.txt'
OutputFilePath=r'D:\Source2.txt'
PrepareDataForEconomic(InputFilePath, OutputFilePath)
```

ДОДАТОК Ж

Лістинг програми C++ Нейронна мережа для прогнозування майбутніх значень індексу Доу Джонсона

Neiron.h

```
#pragma once
#include<vector>
#include<memory>
#include<random>
#include"I threshold function.h"
class INeiron {
public:
	//typedef std::pair<std::shared_ptr<INeiron>, double> Akson;
	typedef std::pair<std::shared_ptr<INeiron>, double> Dendrite;
	virtual ~INeiron() {}
	virtual void AddDendrite(std::shared_ptr<INeiron>) = 0;
	virtual void ResetNeiron() = 0;
	virtual int GetLayNumber()const = 0;
	virtual void CalculateOutput() = 0;
	virtual double GetOutput()const = 0;
	virtual void CorrectWeights() = 0;
	virtual void SetZeroWeight(double) = 0;
	virtual double GetZeroWeight()const = 0;
	virtual void SetOutput(double output) = 0;
	virtual double GetCorrection()const { return 0; }
	virtual void CalculateCorrection() {;}
	virtual void SendCorrection()const { ; }
	virtual void GiveCorrectedElementFromNextLay(double dCorrElement) { ; }
	virtual bool IsInLastLayer() { return false; }
	virtual void RefreshOutput() { ; }

	virtual void RefreshCorrection() { ; }

};
class FirstLayNeiron :public INeiron {
private:
	//int m_nLayNumber;
	double m_dOutput;
public:
	FirstLayNeiron(): m_dOutput(0){}
	void AddDendrite(std::shared_ptr<INeiron>)override {;}
```

```
        void ResetNeiron()override { ; }
        int GetLayNumber()const override { return 1; }
        void CalculateOutput()override { ; }
        double GetOutput()const override { return m_dOutput; }
        void CorrectWeights()override { ; }
        void SetZeroWeight(double)override { ; }
        double GetZeroWeight()const override { return 0; }
        void SetOutput(double output)override { m_dOutput = output; }
        virtual void RefreshOutput()override { m_dOutput = 0; }
};


class CNeiron :public INeiron {
public:

protected:
        std::shared_ptr<IThresholdFunction> m_pThresholdFunction;
        int m_nLayNumber;
        int m_nNumberInLay;
        std::vector<std::shared_ptr<Dendrite>> m_aDendriteSet;
        //std::vector<std::shared_ptr<Akson>> m_aAksonSet;
        double m_dZeroWeight;
        double m_dWeightSum;
        double m_dOutput;
        double m_dCorrection;
        double m_dStudySpeed;

public:
        void AddDendrite(std::shared_ptr<INeiron> pNeiron)override
        {
                m_aDendriteSet.push_back(std::make_shared<Dendrite>(pNeiron, 0));
        }
        CNeiron(int nLayNumber, int nNumberInLay, double dStudySpeed, std::shared_ptr< IThresholdFunction>
pThresholdFunction) :m_nLayNumber(nLayNumber),
                m_nNumberInLay(nNumberInLay),
                m_dStudySpeed(dStudySpeed),
                m_dOutput(0),
                m_dZeroWeight(0),
                m_pThresholdFunction(pThresholdFunction),
                m_dWeightSum(0),
                m_dCorrection(0)  {          }
        CNeiron(const CNeiron& rNeiron):
                m_nLayNumber(rNeiron.m_nLayNumber),
                m_nNumberInLay(rNeiron.m_nNumberInLay),
                m_dStudySpeed(rNeiron.m_dStudySpeed),
                m_dOutput(rNeiron.m_dOutput),
                m_dZeroWeight(rNeiron.m_dZeroWeight),
```

```
            m_pThresholdFunction(rNeiron.m_pThresholdFunction),
            m_dWeightSum(rNeiron.m_dWeightSum),
            m_dCorrection(rNeiron.m_dCorrection) {          }
 CNeiron(CNeiron&& rNeiron) noexcept:
            m_nLayNumber(std::move(rNeiron.m_nLayNumber)),
            m_nNumberInLay(std::move(rNeiron.m_nNumberInLay)),
            m_dStudySpeed(std::move(rNeiron.m_dStudySpeed)),
            m_dOutput(std::move(rNeiron.m_dOutput)),
            m_dZeroWeight(std::move(rNeiron.m_dZeroWeight)),
            m_pThresholdFunction(std::move(rNeiron.m_pThresholdFunction)),
            m_dWeightSum(std::move(rNeiron.m_dWeightSum)),
            m_dCorrection(std::move(rNeiron.m_dCorrection)) {       }
double GetOutput()const
{
            return m_dOutput;
}
int GetLayNumber()const override
{
            return m_nLayNumber;
}
void CalculateOutput()override
{
            m_dWeightSum = 0;
            for (const auto & element : m_aDendriteSet)
                        m_dWeightSum += (*element).first->GetOutput() * (*element).second;
            m_dWeightSum -= m_dZeroWeight;
            m_dOutput = (*m_pThresholdFunction)(m_dWeightSum);
}
void ResetNeiron()override
{
            std::random_device rd;
            std::mt19937 gen(rd());
            std::uniform_real_distribution<> dist(-0.1, 0.1);

            for(auto & element:m_aDendriteSet)
                        (*element).second=dist(gen);
}
virtual void RefreshOutput()override
{
            m_dOutput = 0;
            m_dWeightSum = 0;
}
virtual void RefreshCorrection()override
{
            m_dCorrection = 0;
}
void SetZeroWeight(double zw)override
```

```
	{
		m_dZeroWeight = zw;
	}
	double GetZeroWeight()const override
	{
		return m_dZeroWeight;
	}
	void CorrectWeights()override
	{
		for (auto& rElement : m_aDendriteSet)
		{
			rElement->second += m_dStudySpeed * rElement->first->GetOutput() * m_dCorrection;
		}
	}
	void SetOutput(double output)override {;}
	virtual void SendCorrection()const override
	{
		for (auto& rElement : m_aDendriteSet)
			(*rElement).first->GiveCorrectedElementFromNextLay((*rElement).second * m_dCorrection);
	}

	virtual void GiveCorrectedElementFromNextLay(double dCorrectElement) override
	{
		m_dCorrection += dCorrectElement;
	}

	virtual void CalculateCorrection()override
	{
		m_dCorrection *= m_pThresholdFunction->Derivative(m_dWeightSum);
	}
};

class CLastLayNeiron : public CNeiron {
public:
	CLastLayNeiron(int nLayNumber, int nNumberInLay, double dStudySpeed, std::shared_ptr< IThresholdFunction>
pThresholdFunction):CNeiron(nLayNumber, nNumberInLay,  dStudySpeed, pThresholdFunction){}
	virtual void GiveCorrectedElementFromNextLay(double dCorrectElement) override
	{
		m_dCorrection = dCorrectElement-m_dOutput;
	}

	virtual bool IsInLastLayer() { return true; }
};
```

I threshold function.h

```
#pragma once
class IThresholdFunction {
```

```
public:
	virtual double operator()(double) = 0; // значение пороговой функции
	virtual double Derivative(double) = 0; // производная этой функции
};

class CTresholdSigmoid :public IThresholdFunction {
private:
	double m_dKoef;
public:
	CTresholdSigmoid(double dKoef = 0.5);
	virtual double operator()(double )override;
	virtual double Derivative(double)override;
};
```

Neiron.cpp

```
#include "pch.h"
#include <iostream>
#include<map>
#include"I threshold function.h"
#include"Neiron.h"
#include <fstream>
#include<assert.h>
#include<math.h>

const double dStudySpeed = 0.4;

std::vector<double> MakeNormalizedVector(const std::vector<double>& aInitialVector, double dMashtab = 1)
{
	std::vector<double> aResult(aInitialVector.size(), 0);

	double dMax =aInitialVector.empty()? 0 : aInitialVector[0];
	double dMin = dMax;
	for (const double& value : aInitialVector)
	{
		if (dMax < value)
			dMax = value;
		if (dMin > value)
			dMin = value;
	}
	for (int i = 0; i < aInitialVector.size(); i++)
		aResult[i] = (aInitialVector[i] - dMin) * dMashtab / (dMax - dMin);
	return aResult;

}
```

```
std::vector<double> MakeNormalizedVector(const std::vector<double>& aInitialVector, int nFirstIndex, int nLastIndex, double
dMashtab = 1)
{
	assert(aInitialVector.size() > nFirstIndex && aInitialVector.size() > nLastIndex);
	std::vector<double> aResult(0, 0);
	if (aInitialVector.size() < nFirstIndex || aInitialVector.size() < nLastIndex)
		return aResult;

	double dMax = aInitialVector[nFirstIndex];
	double dMin = dMax;
	for (int i=nFirstIndex; i<nLastIndex; i++)
	{
		if (dMax < aInitialVector[i])
			dMax = aInitialVector[i];
		if (dMin > aInitialVector[i])
			dMin = aInitialVector[i];
	}
	for (int i = nFirstIndex; i <= nLastIndex; i++)
		aResult.push_back((aInitialVector[i] - dMin) * dMashtab / (dMax - dMin));
	return aResult;
}

class NeironNet {
public:
	typedef std::vector<int> ConnectSet;
	typedef std::pair<std::shared_ptr<INeiron>, ConnectSet> NeironWithConnect;
	typedef std::map<int, NeironWithConnect> Lay;
	typedef std::map<int, Lay> NetMap;
private:
	NetMap m_NetMap;
	std::shared_ptr<IThresholdFunction> m_TresholdFunction;
	std::vector<double> m_aInputData;
	bool m_bNormalized;
	double m_dMaxForNormalize;
	double m_dMinForNormalize;
public:

	static NetMap ConstructSimmeticMap(int nLayNumber, int nNumberNeironInLay, std::shared_ptr<IThresholdFunction>
pThresholdFunction)
	{
		NetMap oNetMap;
		ConnectSet aStandartSet(nNumberNeironInLay, 0);
		for (int i = 0; i < nNumberNeironInLay; i++)
			aStandartSet[i] = i;
		Lay aFirstLay;
		for (int j = 0; j < nNumberNeironInLay; j++)
			aFirstLay[j] = std::make_pair(std::make_shared<FirstLayNeiron>(), ConnectSet(0));
```

```
		oNetMap[0] = aFirstLay;



		for (int i = 1; i < nLayNumber-1; i++)
		{
			Lay aInnerLay;
			for (int j = 0; j < nNumberNeironInLay; j++)
				aInnerLay[j] = std::make_pair(std::make_shared<CNeiron>(i,j,
dStudySpeed,pThresholdFunction), aStandartSet);
			oNetMap[i] = aInnerLay;
		}

		Lay aLastLay;
		for (int j = 0; j < nNumberNeironInLay; j++)
			aLastLay[j] = std::pair<std::shared_ptr<INeiron>, ConnectSet>
				(std::make_shared<CLastLayNeiron>(nLayNumber - 1,j,dStudySpeed,pThresholdFunction),
aStandartSet);

		oNetMap[nLayNumber - 1] = aLastLay;

		return oNetMap;

	}

	static NetMap ConstructSimmeticMap(const std::vector<int>& annNumberNeironInLays,
std::shared_ptr<IThresholdFunction> pThresholdFunction)
	{
		int nNumberOfLays = annNumberNeironInLays.size();
		NetMap oNetMap;

		Lay aFirstLay;
		for (int j = 0; j < annNumberNeironInLays[0]; j++)
			aFirstLay[j] = std::make_pair(std::make_shared<FirstLayNeiron>(), ConnectSet(0));
		oNetMap[0] = aFirstLay;



		for (int i = 1; i < nNumberOfLays - 1; i++)
		{
			Lay aInnerLay;
			for (int j = 0; j < annNumberNeironInLays[i]; j++)
			{
				ConnectSet aStandartSet(annNumberNeironInLays[i-1], 0);
				for (int k = 0; k < annNumberNeironInLays[i - 1]; k++)
					aStandartSet[k] = k;
```

```
                                aInnerLay[j] = std::make_pair(std::make_shared<CNeiron>(i, j, dStudySpeed,
pThresholdFunction), aStandartSet);
                        }

                        oNetMap[i] = aInnerLay;
                }

                Lay aLastLay;
                for (int j = 0; j < anNumberNeironInLays[nNumberOfLays - 1]; j++)
                {
                        ConnectSet aStandartSet(anNumberNeironInLays[nNumberOfLays - 2], 0);
                        for (int i = 0; i < anNumberNeironInLays[nNumberOfLays - 2]; i++)
                                aStandartSet[i] = i;
                        aLastLay[j] = std::pair<std::shared_ptr<INeiron>, ConnectSet>
                                (std::make_shared<CLastLayNeiron>(nNumberOfLays - 1, j, dStudySpeed,
pThresholdFunction), aStandartSet);
                }


                oNetMap[nNumberOfLays - 1] = aLastLay;

                return oNetMap;

        }

        NeironNet(NetMap argNetMap, std::shared_ptr<IThresholdFunction> argTresholdFunction):m_NetMap(argNetMap),
                m_TresholdFunction(argTresholdFunction),
                m_aInputData(std::vector<double>(0,0)),
                m_bNormalized(false),
                m_dMaxForNormalize(0),
                m_dMinForNormalize(0){}
        /*
        void InitNet()
        {
                for (auto & n : m_NetMap[0])
                {
                        //n.second.first = std::make_shared<CNeiron>(0, n.first, m_TresholdFunction);
                        n.second.first = std::make_shared<FirstLayNeiron>();
                }
                for (size_t LayNumber = 1; LayNumber< m_NetMap.size() ; LayNumber++)
                {
                        for (auto& n : m_NetMap[LayNumber])
                        {
                                n.second.first = std::make_shared<CNeiron>(0, n.first, m_TresholdFunction);
                                for (auto i : n.second.second)
                                {
                                        n.second.first->AddDendrite(m_NetMap[LayNumber - 1][i].first);
```

```
                                }
                                n.second.first->ResetNeiron();

                        }
                }
        }
        */

        bool NormalizedInputDataVector()
        {
                if (m_aInputData.empty())
                        return false;
                double dMax = m_aInputData[0];
                double dMin = dMax;
                for (const double& value : m_aInputData)
                {
                        if (dMax < value)
                                dMax = value;
                        if (dMin > value)
                                dMin = value;
                }
                for (double& value : m_aInputData)
                        value=((value - dMin)  / (dMax - dMin));
                m_bNormalized = true;
                m_dMaxForNormalize = dMax;
                m_dMinForNormalize = dMin;
                return true;
        }

        double DeNormalize(double value)
        {
                return value * (m_dMaxForNormalize - m_dMinForNormalize) + m_dMinForNormalize;
        }
        void DeNormalize(std::vector<double>& aValues)
        {
                for (auto& value : aValues)
                {
                        value *= (m_dMaxForNormalize - m_dMinForNormalize);
                        value += m_dMinForNormalize;
                }
        }

        void InitNet()
        {

                for (size_t LayNumber = 1; LayNumber < m_NetMap.size(); LayNumber++)
```

```
		{
			for (auto& n : m_NetMap[LayNumber])
			{
				for (auto i : n.second.second)
				{
					n.second.first->AddDendrite(m_NetMap[LayNumber - 1][i].first);

				}
				n.second.first->ResetNeiron();

			}
		}
	}

	bool SetInput(int nFirstIndex)
	{
		if (m_NetMap[0].size() > m_aInputData.size() - nFirstIndex - 2)
			return false;
		if (!m_bNormalized)
			NormalizedInputDataVector();
		for (size_t i = 0;i < m_NetMap[0].size();i++)
			m_NetMap[0][i].first->SetOutput(m_aInputData[nFirstIndex+i]);
	}

	bool SetCorrectionInLastLay(int nFirstIndexForInput)
	{
		int nLayNumber = m_NetMap.size();

		if (m_aInputData.size() < nFirstIndexForInput + m_NetMap[0].size() + m_NetMap[nLayNumber - 1].size())
			return false;
		int nNeironNumberInLastLay = m_NetMap[nLayNumber-1].size();
		int nFirstIndexForCorrection = nFirstIndexForInput + m_NetMap[0].size();
		for (int i = 0; i < nNeironNumberInLastLay; i++)
			m_NetMap[nLayNumber-1][i].first-
>GiveCorrectedElementFromNextLay(m_aInputData[nFirstIndexForCorrection + i]);

		// For debug
		double Error = 0;
		for (int i = 0; i < nNeironNumberInLastLay; i++)
			Error+= 0.5*pow(DeNormalize(m_NetMap[nLayNumber - 1][i].first->GetOutput() -
m_aInputData[nFirstIndexForCorrection + i])/DeNormalize( m_aInputData[nFirstIndexForCorrection + i]), 2);
		std::cout << std::endl << " Error = " << Error;
	}

	void RefreshOutput()
	{
		for (auto& rLay : m_NetMap)
```

```
                for (auto& rNeiroAndConnection : rLay.second)
                        rNeiroAndConnection.second.first->RefreshOutput();
}

void RefreshCorrection()
{
        for (auto& rLay : m_NetMap)
                for (auto& rNeiroAndConnection : rLay.second)
                        rNeiroAndConnection.second.first->RefreshCorrection();
}

bool ForvardGo(int nFirstIndex)
{
        RefreshOutput();
        bool bResult=SetInput(nFirstIndex);
        if (bResult == false)
                return false;

        for (size_t j = 1; j < m_NetMap.size(); j++)
        {
                for (auto& NeirWCon : m_NetMap[j])
                {
                        NeirWCon.second.first->CalculateOutput();
                }
        }
        return bResult;
}

bool BackGo(int nFirstIndex)
{
        RefreshCorrection();
        bool bResult = SetCorrectionInLastLay(nFirstIndex);
        if (bResult == false)
                return false;
        int nNumberOfLay = m_NetMap.size();
        for (int i = nNumberOfLay - 1; i > 0; i--)
        {
                for (auto& NeironAndConnection : m_NetMap[i])
                {
                        NeironAndConnection.second.first->CalculateCorrection();
                        NeironAndConnection.second.first->SendCorrection();
                        NeironAndConnection.second.first->CorrectWeights();
                }

        }

}
```

```
	bool ReadDataFromFile(const std::string& sPathToDataFile)
	{
		std::fstream File;

		File.open(sPathToDataFile);
		if (!File.is_open())
			return false;
		while (!File.eof())
		{
			double value;
			File >> value;
			m_aInputData.push_back(value);
		}
		File.close();

	}

	void RunNet()
	{

	}
	void GetResult(std::vector<double>& aResult, bool bGetDeNormalized=true)
	{
		aResult.clear();
		aResult.resize(m_NetMap[m_NetMap.size() - 1].size(), 0);
		for (int i = 0; i < aResult.size(); i++)
			aResult[i] = m_NetMap[m_NetMap.size() - 1][i].first->GetOutput();
		if (bGetDeNormalized)
			DeNormalize(aResult);
	}

	void GetRightResult(std::vector<double>& aResult, int nShift, bool bGetDeNormalized = true)
	{
		aResult.clear();
		aResult.resize(m_NetMap[m_NetMap.size() - 1].size(), 0);
		for (int i = 0; i < aResult.size(); i++)
			aResult[i] = m_aInputData[m_NetMap[0].size() + nShift+i];
		if (bGetDeNormalized)
			DeNormalize(aResult);
	}
};


int main()
{
	std::shared_ptr<IThresholdFunction> spTresholFunction = std::make_shared<CTresholdSigmoid>();
```

```
	std::vector<int> anNetStructure = {6,6,3,3};
	NeironNet oNN(NeironNet::ConstructSimmeticMap(anNetStructure, spTresholFunction), spTresholFunction);
	oNN.ReadDataFromFile("Source.txt");
	oNN.InitNet();
	int StudyLen = 10;
	for (int k = 0; k < 4; k++)
	{
		for (int i = 0; i < StudyLen; i++)
		{
			oNN.ForvardGo(i);
			oNN.BackGo(i);
		}
		std::cout << std::endl;
	}

	oNN.ForvardGo(StudyLen);
	std::vector<double> aPrognoz;

	oNN.GetResult(aPrognoz);
	std::vector<double> aRightValues;
	oNN.GetRightResult(aRightValues, StudyLen);
	int r = 8;
	/*
	oNN.SetInput({ 6,3,2,1,7,4 });
	std::shared_ptr<INeiron> n0 = std::make_shared<FirstLayNeiron>(2.5);
	std::shared_ptr<INeiron> n1=std::make_shared<CNeiron>(1, 1,spTresholFunction);
	std::shared_ptr<INeiron> n2 = std::make_shared<CNeiron>(2,1, spTresholFunction);

	n1->AddDendrite(n0);
	n2->AddDendrite(n1);

	n1->ResetNeiron();
	n2->ResetNeiron();

	n1->CalculateOutput();
	n2->CalculateOutput();
  std::cout << "Hello World!\n"<<n0->GetOutput()<<"\n"<< n1->GetOutput() << "\n"<< n2->GetOutput() << "\n";
	*/
}
```

ThresholdFunction.cpp

```
#include"pch.h"
#include"I threshold function.h"
#include<math.h>
CTresholdSigmoid::CTresholdSigmoid(double dKoef /*= 0.5*/): m_dKoef(dKoef){}
```

```
double CTresholdSigmoid::operator()(double arg)
{
	return 1. / (1. + exp(-2*m_dKoef*arg));
}
double CTresholdSigmoid::Derivative(double arg)
{
	return 2*m_dKoef*(1. - operator()(arg))*operator()(arg); // производная сигмоидальной функции легко выражается
                                                                 // через значение самой
функ
```