Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Нейромережевий підхід до вирішення задач прогнозування цін цифрового активу
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-2
Троценко О.І.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

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

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 22 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Троценко Олексію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Нейромережевий підхід до вирішення задач прогнозування цін цифрового активу

затверджена наказом по університету від 24 березня 2022 р. № 414 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 11 травня 2022 р.

3. Вихідні дані до роботи джерела інформації, які описують основну частину теоретичних досліджень, опис існуючих архітектур нейронних мереж та способу створення власного набору даних, бібліотеки Tensorflow та Keras необхідні для написання програмної реалізації методу прогнозування цін цифрового активу та для проведення досліджень.

4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної галузі та постановка завдання, дослідження способів розробки нейронних мереж, розробка ефективного методу прогнозування цін цифрового активу, аналіз отриманих результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Рисунок 1 – Ноди які пов'язані один з одним, Рисунок 2 – Зворотне поширення для простої нейронної мережі, Рисунок 3 – Перцептрон, Рисунок 4 – Приклад рекурсивної мережи, Рисунок 5 – Приклад мережі з довгою короткостроковою пам'яттю, Рисунок 6 – Метод найшвидшого спуску, Рисунок 7 – Приклад стохастичного градієнтного спуску, Рисунок 8 – Графік залежності входів від виходів для функції активації ReLU, Рисунок 9 – Графік входів і виходів для функції активації сигмоїдальної форми, Рисунок 10 – Графік залежності входів від виходів для функції активації Tanh, Рисунок 11 – Як вибрати функцію активації прихованого шару, Рисунок 12 – Графік залежності входів від виходів для функції лінійної активації, Рисунок 13 – Як обрати функцію активації вихідного рівня, Рисунок 14 – Логотип TensorFlow, Рисунок 15 – CSV файл з даними Bitcoin, Рисунок 16 – Зображення даних у Yahoo Finance, Рисунок 17 – Графік loss для першої моделі класифікації, Рисунок 18 – Графік accuracy для першої моделі класифікації

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультантапро виконання розділу	
		підпис	дата
Основна частина	проф. Рябова Н. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	28.03	виконано
2	Аналіз предметної галузі та постановка завдання	01.04 – 04.04	виконано
3	Дослідження способів розробки	05.04 – 10.04	виконано
4	Створення власного набору даних	11.04 – 12.04	виконано
5	Розробка методу прогнозування цін	13.04 – 17.05	виконано
6	Експериментальна перевірка, обробка та аналіз	18.04 – 25.05	виконано
	отриманих результатів		
7	Оформлення пояснювальної записки	26.05 – 01.05	виконано
8	Попередній захист	02.05	виконано
9	Захист перед ЕК	11.05	виконано

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Рябова Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 69 с., 32 рис., 2 дод., 20 джерело.

НЕЙРОННА МЕРЕЖА, ПРОГНОЗУВАННЯ, ANACONDA, GOOGLE TENSORFLOW, KERAS, PYTHON

Об’єкт дослідження – задачі прогнозування цін цифрового активу.

Предмет дослідження – нейромережеві методи прогнозування цін цифрового активу.

Мета роботи – дослідження і розробка ефективного методу прогнозування цін цифрового активу на основі архітектури глибоких нейронних мереж.

Методи дослідження – глибинні нейронні мережі, класифікація, регресія.

В результаті проведених досліджень була побудована архітектура мережі та підібрані ваги таким чином, щоб мережа вирішувала поставлене завдання.

Отримані результати використовуються для порівняння точності прогнозування цін цифрового активу на тестовому наборі даних.

В якості програмної платформи використано Anaconda та мова програмування Python з бібліотеками машинного навчання Google Tensorflow та Keras. Пропонована розробка є корисною для пошуку найкращого часу для створення портфеля інвестицій у цифрових активах.

ABSTRACT

Explanatory note: 69 p., 32 fig., 2 ann., 20 sources.

ANACONDA, FORECASTING, GOOGLE TENSORFLOW, NEURAL NETWORK, KERAS, PYTHON

The object of research – digital asset price forecasting tasks.

The subject of research – neural network methods for forecasting digital asset prices.

The aim of the work – research and development of an effective method of forecasting a digital asset based on deep neural networks.

Methods of research – deep neural networks, classification, regression.

As a result of the research, the architecture of the network was constructed and the weights were chosen so that the network would solve the task.

The results obtained are used to compare the accuracy of the forecasting digital asset on the test dataset.

Anaconda and Python programming language with Google Tensorflow and Keras machine learning libraries were used as the programming platform. The proposed development is useful for finding the best time to build an investment portfolio in digital assets.

ЗМІСТ

ВСТУП

Нейромережеві технології, і особливо, глибинні нейронні мережі, дедалі ширше застосовують у різних галузях для вирішення важливих прикладних завдань. Одним із перспективних напрямів їх застосування є автоматизація процесів аналізу та прогнозування поведінки фінансових ринків, а також їх учасників. Основною перевагою глибоких нейронних мереж є їх здатність аналізувати великі обсяги даних і на основі навчання передбачати потрібні патерни поведінки фінансового інструменту. Як правило, вирішення такого роду завдань є складним і трудомістким процесом, результати якого мають наближений, імовірнісний характер. Незважаючи на свою розвиненість, класичні методи та моделі, що використовуються для прогнозування курсової динаміки цінних паперів та криптовалют, аналізу ризиків, управління портфелем, класифікації даних, все менш ефективно працюють в умовах глобалізації світового фінансового простору, розширення ринків, появи нових фінансових інструментів. У більшості випадків складність викликає виявлення безлічі зв'язків між факторами на фінансових ринках, які необхідно врахувати та провести їх моделювання.

На фоні збільшення інфляції долара та спроб великих компаній зберегти накопичені кошти, все більше інвестиційних портфелів збираються саме із цифрових активів, таких як криптовалюта. Одним з найбільш популярних, але в той же час волатильних представників криптовалюти є біткоін, ціна якого в даний час досягає понад 40 000 $ за одну монету. Такі гіганти як Tesla і MicroStrategy вклали мільярди доларів у біткоїн, називаючи його «Новим золотом». Але оскільки криптовалюта має тимчасові падіння в один період часу і стрімке зростання в інший, необхідно визначити патерн ціни і передбачити дату створення портфеля інвестицій [1].

Враховуючи вище згадане, темою магістерської кваліфікаційної роботи обрано – нейромережевий підхід до розв'язання задач прогнозування ціни цифрового активу. Такий підхід може використовувати дані про ціну активу за весь період його існування. Потім цей аналіз застосовується для прогнозування зміни ціни. Використання методів штучного інтелекту та машинного навчання дозволяє вийти за рамки аналізу та прорахунку даних, поєднати логічні закономірності цін для отримання повної картини. Також може допомогти у визначенні основних моделей купівлі та продажу активу, знаходженні вузьких місць, визначення аномалій та прогнозуванні фінансових трендів, визначенні ціни з більш високою точністю під час навчання на великих вибірках. Одним з важливих завдань, яке також вирішується за допомогою вибраних технологій за умови накопичення великих обсягів даних є довгострокове прогнозування того, коли актив може обрушитися в ціні задовго до цієї події.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз сучасного стану в області цифрової економіки

Цифрова економіка міцно входить у життя. Поряд із звичними матеріальними речами у цивільному обороті з'явилися цифрові, тобто, по суті, віртуальні активи. Найкращими представниками яких є криптовалюти та токени.

Цифровий актив – це віртуальний об'єкт цивільного обороту, який має реальну фінансову вартість і звертається у розподіленому реєстрі у вигляді унікального ідентифікатора.

За своєю правовою природою цифровий актив є спорідненим інформаційним ресурсам, оскільки представлений у цифровому вигляді об'єкт громадянського обороту має такі основні властивості інформаційного ресурсу, як:

– структурування за певними параметрами та категоріями;

– фіксується на цифровому носії;

– його можна зберігати, передавати, обмінювати та використовувати.

Також цифрові активи мають безперечну подібність із безготівковими коштами: цифрові активи можуть передаватися від однієї особи до іншої. Будучи нематеріальним майном, не втрачають своїх властивостей з часом, можуть використовуватись частинами, траншами, різні цифрові активи по-своєму виражені у обороті. Отже, цифрові активи формально можуть бути прирівняні до засобу платежу в економічному сенсі і, більше, найбільш схожі саме з безготівковими коштами через свою нематеріальну форму.

На сьогоднішній день існує значна кількість видів цифрових ресурсів. Найпоширеніших з них є криптовалюта. Являє собою різновид віртуальної валюти, облік внутрішніх розрахункових одиниць якої забезпечує децентралізована платіжна система. На сьогодні криптовалюта не регулюється жодним банківським наглядовим органом у світі та не має

емісійного центру. Створюють код, добувають умовний обмежений ресурс, самі користувачі з допомогою обчислювальної потужності комп'ютерів. Криптовалюту можна знайти самостійно або купити на криптовалютних біржах. Можливість використання в обороті з'явилася завдяки створенню та розповсюдженню технології блокчейн.

Блокчейн – революційна технологія XXI століття, тому кожен намагається отримати шматочок цього грошового пирога. Трейдери ловлять тренд, заробляючи неймовірні суми протягом кількох місяців чи навіть днів.

Інституційні інвестори, розуміючи величезний потенціал технології блокчейну, все частіше вкладають гроші у ризиковані проекти із проривними ідеями. Зараз уряди країн по всьому світу почали обговорювати питання про те, що робити з цими інструментами, що зароджуються.

Кожен учасник може запустити ноду з повною копією блокчейна. Повні ноди, які можуть записувати транзакції блокчейна, називаються вузлами консенсусу або майнерами. Повні ноди, які лише перевіряють правильність транзакцій, називаються вузлами аудиту. Легкі клієнти не зберігають повних копій блокчейну, а взаємодіють із мережею, використовуючи повні ноди.

Більшість користувачів для транзакцій використовують саме легких клієнтів або web гаманці. Усі ноди пов'язані один з одним. При такому наборі елементів архітектура мережі стає більш стійкою, як показано на рисунку 1.1.

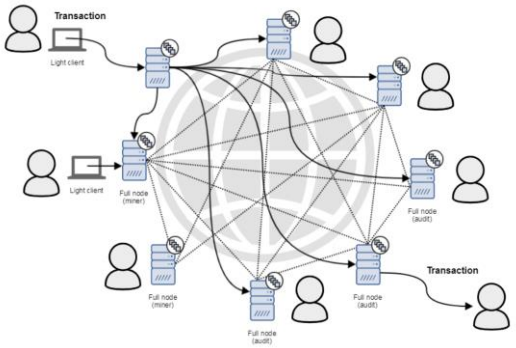


Рисунок 1.1 – Ноди які пов'язані один з одним

Кожна блокчейн транзакція має бути підписана цифровим підписом. Тому для транзакції кожен учасник повинен мати пару ключів. А саме private та public ключ. Іноді кілька ключів називають гаманець, т.к. ключі однозначно пов'язані з унікальною цифровою адресую та балансом учасника. Насправді ключі та адреси – це просто рядки цифр у різних системах числення.

Для створення цифрового підпису в блокчейнах використовується алгоритм, що базується на еліптичних кривих. Для його роботи приватний ключ, 256 бітне число, зазвичай береться випадково. Число варіантів ключів становить 2 у ступені 256, тому можна говорити про практичну неможливість збігу значень приватних ключів.

Публічний ключ виходить з приватного шляхом множення його значення на координати точки, що знаходиться на еліптичній кривій, у результаті виходять координати нової точки цієї ж кривої. Ця дія гарантує отримання пари ключів, придатних для цифрових підписів транзакцій. І нарешті адреса гаманця однозначно обчислюється з громадського ключа.

Перевіривши валідність транзакцій, ноди формують їх блоки. Крім транзакцій у блок записується хеш попереднього блоку. Хеш повинен мати встановлені умови складності. Наприклад, у мережі Bitcoin складність хеша автоматично змінюється раз на 2 тижні залежно від потужності мережі так, щоб блок генерувався приблизно раз на 10 хвилин. Складність визначається такою умовою: знайдений хеш повинен бути меншим заздалегідь заданого числа. Якщо ця умова не виконується, то Nonce додається 1, і робота з обчислення хеша повторюється. Для підбору хешу використається поле Nonce, т.к. це єдині дані в блоці, які можна змінити, решта мають залишатися незмінними.

Процес знаходження хешей називається майнінгом, за аналогією зі здобиччю золота. Назва досить точно визначає суть процесу, т.к. відбувається простий перебір варіантів, і якщо хтось знайшов відповідний хеш, це справді удача.

Також існують Consortium або Public Permissioned blockchain. Це такі блокчейни, до яких кожен може підключитись для перегляду, але додавати інформацію або підключити свій вузол учасник може тільки з дозволу інших учасників. Такі блокчейни будують організації з підвищення довіри із боку замовників чи споживачів продукції чи суспільства загалом. Тут надійність також досягається присутністю довіри між учасниками та тими самими алгоритмами консенсусу PoS та BFT.

Якщо згадати інший вид цифрових ресурсів, маю на увазі токени. Токен сприймається як різновид цифрового активу який характеризується такими ознаками:

– посвідчення їм права об'єкта громадянських прав;

– існування у блокчейні чи інший розподіленої інформаційної системі. За своєю сутністю токен є цифровий спосіб фіксації майнових прав.

Залежно від виконуваних функцій токени блокчейна поділяють на програмні, платіжні, кредитні, інвестиційні токени і токени, які закріплюють права іншого об'єкта. Перші – це віртуальні одиниці, що надають власнику право на дії виключно всередині комп'ютерної системи. Другий різновид використовується для оплати товарів, робіт і послуг. Наприклад, токени платіжних систем Біткойн, Ефіріум. Третій вид використовується для закріплення прав вимоги повернення коштів чи пологових речей. Четвертий вид відображає операції з інвестування реальних коштів або віртуальних валют у бізнес-проекти. І п'ятий відображає права їх власника на роботи, послуги, речі, цінні папери, частку у статутному капіталі, інші об'єкти. У токені закодовано ціну цифрового об'єкта цивільного обороту. Використовуючи токен як товар, користувачі системи вправі здійснювати транзакції з його оплати у вигляді криптовалюти як цифрової одиниці розрахунку блокчейн-системі. У цьому випадку він розглядається як криптовалютний токен, тобто як засіб платежу, який можна обміняти на інші цифрові об'єкти, або сплатити їм через транзакцію. В останньому своєму значенні токен реалізує функцію

виконання грошового зобов'язання за вже надані товари, виконані роботи та надані послуги або дає право на їх отримання.

1.2 Обґрунтування використання нейронної мережі для прогнозування

У сучасному світі все з більшою гостротою виявляється інтерес до якісного прогнозування фінансових ринків. Це пов’язано з швидким розвитком високих технологій і, з появою нових інструментів аналізу даних. Проте той технічний аналіз, яким звикла користуватися більшість учасників ринку, не є ефективним. Прогнози з урахуванням експоненційних ковзних середніх, осциляторах та інших індикаторах не дають відчутний результат, так як економіка часто буває ірраціональною, тому що рухається ірраціональними мотиваціями людей.

Останніми роками, у фінансових аналітиків стали викликати великий інтерес так звані штучні нейронні мережі – це математичні моделі, і навіть їх програмні чи апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронів. Згодом ці моделі стали використовувати у практичних цілях, як правило, у завданнях прогнозування. Нейронні мережі не програмуються у звичному значенні цього слова, вони навчаються. Можливість навчання – одне з основних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними та вихідними, а також виконувати узагальнення. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення та виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення певної послідовності на основі кількох попередніх значень та факторів, що існують зараз.

1.3 Огляд завдання класифікації та регресії у межах прогнозування за допомогою нейронної мережі

Як правило, нейронні мережі навчаються в якості класифікатора для складання класифікацій. Але в машинному навчанні з вчителем зазвичай повинні робити регрес або прогнози, такі як прогнозування завтрашнього фондового індексу.

Різниця між класифікацією і регресією полягає в тому, що класифікація виводить вірогідність передбачення для класу / класів, а регресія дає значення. Ми можемо зробити нейронну мережу для виведення значення, просто змінивши функцію активації в останньому шарі для виведення значень.

Прогнозуюче моделювання – це проблема розробки моделі з використанням історичних даних для прогнозування нових даних, коли у нас немає відповіді. Прогнозуюче моделювання може бути описано як математична проблема наближення функції відображення f від вхідних змінних X до вихідних змінних y [6]. Це називається проблемою наближення функцій. Завдання алгоритму моделювання полягає в тому, щоб знайти кращу функцію відображення, яку ми можемо, з огляду на час і ресурси. Як правило, ми можемо розділити всі завдання наближення функцій на завдання класифікації і завдання регресії.

Класифікаційне прогнозуюче моделювання – це завдання наближення функції відображення f від вхідних змінних X до дискретних вихідних змінних y. Вихідні змінні часто називають мітками або категоріями. Функція відображення пророкує клас або категорію для даного спостереження.

Наприклад, повідомлення електронної пошти з текстом та / або картинкою може бути розцінена, як належить, одному з двох класів: «спам» і «не спам». Виходячи з цього прикладу маємо наступні особливості задачі класифікації:

– проблема класифікації вимагає, щоб приклади були класифіковані в один або два класи;

– класифікація може мати дійсні або дискретні вхідні змінні;

– проблема з двома класами часто називається проблемою двокласною або двійковій класифікації;

– проблема з більш ніж двома класами часто називається проблемою класифікації декількох класів;

– проблема, коли для прикладу призначається кілька класів, називається проблемою класифікації за кількома мітками.

Для класифікаційних моделей характерно передбачати безперервне значення як ймовірність даного прикладу, що належить кожному вихідному класу. Ймовірності можуть бути інтерпретовані як ймовірність або достовірність даного прикладу, що належить кожному класу. Прогнозована ймовірність може бути перетворена в значення класу шляхом вибору мітки класу, яка має найбільшу ймовірність.

Наприклад, конкретному текстовому електронному листу можуть бути присвоєні ймовірності 0,1 як спам і 0,9 як не спам. Ми можемо перетворити ці ймовірності в мітку класу, вибравши мітку «не спам», оскільки вона має найбільшу прогнозовану ймовірність.

Існує багато способів оцінити майстерність моделі прогнозування класифікації, але, можливо, найбільш поширеним є розрахунок точності класифікації. Точність класифікації – це відсоток правильно класифікованих прикладів від усіх зроблених прогнозів.

Наприклад, якщо модель прогнозування класифікації зробила 5 прогнозів і 3 з них були правильними, а 2 з них були неправильними, то точність класифікації моделі, заснованої тільки на цих прогнозах, була б порахована за формулою 1.1:

$$\text{точність} = \text{правильні прогнози} / \text{загальні прогнози} * 100\%. \quad (1.1)$$

Алгоритм, здатний вивчати модель прогнозування класифікації, називається алгоритмом класифікації.

Прогнозуюче регресійне моделювання – це завдання наближення функції відображення f від вхідних змінних X до безперервної вихідної змінної y. Безперервна вихідна змінна – це дійсне значення, таке як ціле число або значення з плаваючою комою. Це часто кількості, такі як суми і розміри.

Виходячи з цього прикладу маємо наступні особливості задачі регресії:

– проблема регресії вимагає прогнозування кількості;

– регресія може мати дійсні або дискретні вхідні змінні;

– проблема з кількома вхідними змінними часто називається проблемою багатовимірної регресії;

– проблема регресії, коли вхідні змінні впорядковані за часом, називається проблемою прогнозування часових рядів.

Оскільки модель прогнозування регресії пророкує кількість, навик моделі повинен бути вказаний як помилка в цих прогнозах. Існує багато способів оцінити майстерність моделі прогнозування регресії, але, мабуть, найбільш поширеним є обчислення середньоквадратичної помилки, скорочено позначається як RMSE.

Наприклад, якщо модель прогнозування регресії зробила 2 прогнозу: одне з 1,5, де очікуване значення дорівнює 1,0, а інше – 3,3 і очікуване значення – 3,0, тоді середньоквадратичне відхилення буде пораховане за формулою 1.2:

$$RMSE = sqrt(average(error^2)). \quad (1.2)$$

Перевага RMSE полягає в тому, що одиниці оцінки помилки знаходяться в тих же одиницях, що і прогнозоване значення.

Алгоритм, який здатний вивчати модель прогнозування регресії, називається алгоритмом регресії. Деякі алгоритми мають у своїй назві слово

«регресія», наприклад, лінійна регресія і логістична регресія, що може призвести до плутанини, оскільки лінійна регресія є алгоритмом регресії, тоді як логістична регресія є алгоритмом класифікації.

Класифікаційні задачі прогнозного моделювання відрізняються від завдань регресійного прогнозного моделювання наступним:

– класифікація – це завдання прогнозування мітки дискретного класу;

– регресія – це завдання прогнозування безперервного кількості.

Існує деякий збіг між алгоритмами класифікації і регресії; наприклад:

– алгоритм класифікації може передбачити безперервне значення, але безперервне значення має форму ймовірності для мітки класу;

– алгоритм регресії може прогнозувати дискретне значення, але дискретне значення в формі цілочисленної величини.

Деякі алгоритми можуть використовуватися як для класифікації, так і для регресії з невеликими модифікаціями, такими як дерева рішень і штучні нейронні мережі. Деякі алгоритми не можуть бути легко використані для обох типів завдань, таких як лінійна регресія для прогнозного моделювання з регресією і логістична регресія для прогнозного моделювання класифікації.

Важливо відзначити, що спосіб, яким ми оцінюємо прогнози класифікації і регресії, варіюється і не перекривається, наприклад:

– класифікаційні прогнози можуть оцінюватися з використанням точності, тоді як регресивні прогнози не можуть;

– прогнози регресії можуть бути оцінені з використанням середньоквадратичної помилки, тоді як прогнози класифікації не можуть.

У деяких випадках можна перетворити проблему регресії в задачу класифікації. Наприклад, прогнозована кількість може бути перетворено в окремі сегменти. Це часто називається дискретизацією, а результуюча вихідна змінна представляє собою класифікацію, в якій мітки мають впорядковане відношення зване порядковим номером.

У деяких випадках проблема класифікації може бути перетворена в проблему регресії. Наприклад, мітка може бути перетворена в безперервний

діапазон. Деякі алгоритми вже роблять це, пророкуючи ймовірність для кожного класу, яка, в свою чергу, може бути масштабувати до певного діапазону за формулою 1.3:

$$\text{кількість} = \min + \text{ймовірність} * \text{діапазон}. \quad (1.3)$$

Крім того, значення класів можуть бути впорядковані і зіставлені з безперервним діапазоном. Якщо мітки класів в завданні класифікації не мають природних порядкових відносин, перетворення з класифікації в регресію може привести до несподіваних або поганих характеристик, оскільки модель може дізнатися помилкове або неіснуюче відображення з вхідних даних в безперервний вихідний діапазон.

1.4 Аналіз основних підходів навчання нейронних мереж

Перш за все для створення нейронної мережі, необхідно чітко розуміти її завдання, для кого саме вона проектується та як змінить життя людей. Як тільки мережа була структурована для конкретного завдання, вона готова до навчання. Щоб почати цей процес, початкові ваги вибираються випадковим чином. Потім настає етап навчання.

Існує 3 підходи навчання нейронних мереж [4]:

– з вчителем;
– без вчителя;
– з підкріпленням.

Навчання з вчителем включає в себе механізм забезпечення мережі бажаним виходом, або вручну «оцінюючи» продуктивність мережі, або шляхом надання бажаних виходів разом з входами.

Навчання без вчителя – це коли мережа повинна розібратися у входах без сторонньої допомоги.

Базою для навчання з підкріпленням є прийняття відповідних заходів для максимізації винагороди в конкретній ситуації.

Переважна більшість мереж використовують навчання з учителем. Навчання без вчителя використовується для виконання деякої початкової характеристики вхідних даних.

При навчанні з вчителем надаються як входи, так і виходи. Це означає, що існує навчальний набір даних, який містить приклади з істинними значеннями: теги, класи, індикатори.

Нерозподілені набори також використовуються для навчання нейронних мереж, і для цього були розроблені відповідні неконтрольовані методи.

Штучна нейронна мережа складається з трьох компонентів: вхідний шар, прихований обчислювальний шар і вихідний шар.

Навчання складається з вибору коефіцієнтів для кожного нейрона в шарах, щоб при певних вхідних сигналах отримували необхідний набір вихідних сигналів. Нейронні мережі навчаються в два етапи:

– пряме поширення помилок;

– зворотне поширення помилок.

Під час прямого поширення помилки, робиться прогноз відповіді. Наприклад, для нейронних мереж з двома входами, трьома нейронами і одним виходом ми можемо встановити початкові ваги випадковим чином: w1, w2. Помножте введення на ваги, щоб сформувати прихований шар, як у формулі 1.4:

$$h_1 = (x_1 * w_1) + (x_2 * w_1),$$
$$h_2 = (x_1 * w_2) + (x_2 * w_2)$$
$$h_3 = (x_1 * w_3) + (x_2 * w_3) \quad (1.4)$$

де h – прихований шар;

x – введенні данні;

w – ваги.

Вихід з прихованого шару передається через нелінійну функцію активації, щоб отримати мережевий вихід, як у формулі 1.5:

$$y = f(h_1, h_2, h_3). \quad (1.5)$$

При зворотному поширенні помилка між фактичним і прогнозованим відгуком зводиться до мінімуму [5]. Це зображено на рисунку 1.2.

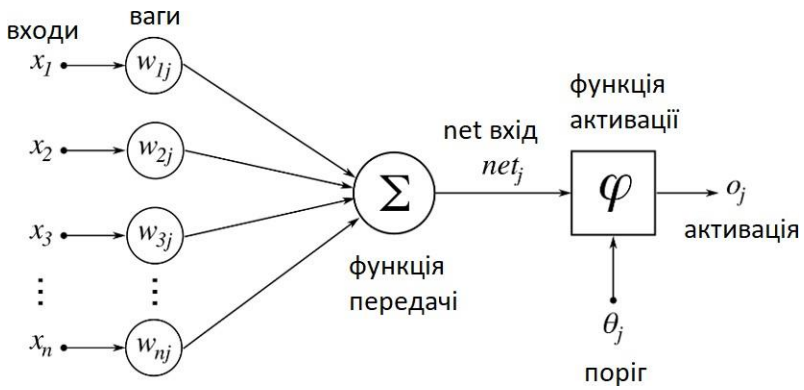


Рисунок 1.2 – Зворотне поширення для простої нейронної мережі

Зворотне поширення ефективно обчислює градієнт функції втрат по відношенню до ваги мережі для одного вводу – виводу. Це дозволяє використовувати градієнтні методи для навчання багаторівневих мереж, оновлюючи ваги, щоб мінімізувати втрати. Метод швидкого спуску або інші варіанти, такі як стохастичний градієнтний спуск, зазвичай використовуються для цього. Щоб мінімізувати значення помилку використовується формула 1.6.

$$\min (\text{loss\_value} = (y)^2 - (y')^2), \quad (1.6)$$

де y – очікуване значення;

y' – отримане значення.

Загальна помилка розраховується як різниця між очікуваним значенням y, з навчального набору і отриманим значенням y', розрахованим на етапі прямого поширення помилки, що проходить через функцію

вартості. Приватна похідна помилки розраховується для кожної ваги. Часткові диференціали відображають внесок кожного ваги в загальну втрату. Потім ці диференціали множаться на число, зване швидкістю навчання η. Результат потім віднімається з відповідних ваг. Пошук оновлених ваг показується у формулі 1.7.

$$w1 = w1 - (\eta * \partial(\text{розмір втрат}) / \partial(w1)), w2 = w2 - (\eta * \partial(\text{розмір втрат}) / \partial(w2))$$
$$w3 = w3 - (\eta * \partial(\text{розмір втрат}) / \partial(w3)) \quad (1.7)$$

де w – ваги;

∂ – розмір втрат;

η – швидкістю навчання.

Той факт, що ми приймаємо і ініціалізуємо ваги випадковим чином, і вони дають точні відповіді, не здається цілком розумним. Проте, це загальновідомий метод навчання, що працює добре.

Вибір відправної точки для складних архітектур нейронних мереж є досить складним завданням, але для більшості випадків існують перевірені технології для вибору початкового наближення.

Крім того, навчання мережі в даний час проводиться не по всьому набору даних, а по вибірках певного розміру, так званим партіям. Це означає, що ваги в нейронних мережах налаштовуються від епохи до епохи, щоб отримати кращі результати.

Таким чином, мережа обробляє вхідні дані і порівнює отримані результати з необхідними вихідними даними. Помилки потім поширюються назад через систему, змушуючи систему регулювати ваги, які контролюють мережу. Цей процес відбувається знову і знову, так як ваги постійно змінюються. Набір даних, який дозволяє проводити навчання, називається навчальним набором. Під час навчання мережі один і той же набір даних обробляється багато разів, оскільки ваги уточняються.

Поточні пакети розробки комерційних мереж надають інструменти для моніторингу того, наскільки добре штучна нейронна мережа має здатність передбачити правильну відповідь. Ці інструменти дозволяють навчальному процесу тривати протягом декількох днів, зупиняючись тільки тоді, коли система досягає деякої статистично бажаної точності. Однак деякі мережі ніколи не навчяться давати правильні відповіді. Це може бути пов'язано з тим, що вхідні дані не містять конкретної інформації, з якої є можливість отримати бажаний результат або даних недостатньо для повного навчання. В такому випадку розробник повинен переглянути вхідні і вихідні дані, кількість шарів, кількість елементів на шар, з'єднання між шарами, функції підсумовування, передачі і навчання, і навіть самі початкові ваги, щоб вирішити проблему не навчання.

Інша частина – правила навчання. Існує безліч алгоритмів, які використовуються для реалізації адаптивного зворотного зв'язку, необхідної для коригування ваг під час тренування. Найбільш поширеним методом є зворотне поширення помилок [6].

Коли, нарешті, система була належним чином навчена, і подальше навчання не потрібне, ваги можуть, при бажанні, бути «заморожені». У деяких системах цю завершену мережу з зафіксованими вагами потім перетворюють в апаратне забезпечення, так що вона може бути швидкою. Інші системи не блокуються та продовжують вчитися в процесі експлуатації.

Інший тип навчання називається навчанням без учителя. У цьому випадку мережа забезпечується входами, але не бажаними виходами. Потім система сама повинна вирішити, які функції вона буде використовувати для угрупування вхідних даних. Це часто називають самоорганізацією або адаптацією.

Життя сповнене ситуацій, коли точних тренувальних наборів не існує. Деякі з цих ситуацій пов'язані з військовими діями, в яких можуть зустрітися нові бойові прийоми і нову зброю. Через цього несподіваного аспекту життя і людського бажання бути готовим, тривають дослідження і

надія на цю область. Тим не менш, у даний час основна частина роботи нейронних мереж знаходиться в системах навчання з учителем. Навчання з вчителем наданий час – це про досягнення результатів.

Останній тип – навчання з підкріпленням. Йдеться про прийняття відповідних заходів для максимізації винагороди в конкретній ситуації. Він використовується різним програмним забезпеченням і машинами, щоб знайти найкращу можливу поведінку або шлях, по якому він повинен йти в конкретній ситуації. Навчання з підкріпленням відрізняється від навчання з учителем тим, що в навчанні з учителем навчальні дані мають ключ відповіді, тому модель навчається з правильною відповіддю, в той час як в навчанні з підкріпленням немає відповіді, але підкріплювальний агент вирішує, що робити. За відсутності навчального набору даних він повинен вчитися на своєму досвіді.

Основні моменти в навчанні з підкріпленням:

– вхідні дані повинні бути у початковому стані, з якого почнеться модель;

– є багато можливих висновків, так як існує безліч рішень для конкретної проблеми;

– навчання засноване на введенні, модель поверне стан, і користувач вирішить винагородити або покарати модель на основі її виведення;

– модель продовжує вчитися;

– краще рішення визначається виходячи з максимальної нагороди.

Є два різновиди навчання з підкріпленням:

– позитивне підкріплення, коли подія, що відбувається через певну поведінку, збільшує силу і частоту цієї позитивної поведінки. Перевагами цього різновиду є максимальне підвищення продуктивність мережі та стійка зміна протягом тривалого періоду часу. Недоліком є те, що занадто сильне посилення може призвести до перевантаження станів, що може зменшити результати;

– негативне підкріплення визначається як посилення поведінки, тому що негативний стан припиняється або його уникають.

Різновиди практичного застосування навчання з підкріпленням:

– в робототехніці для промислової автоматизації;

– в машинному навчанні та обробці даних;

– для створення систем навчання, які надають індивідуальні інструкції та матеріали відповідно до вимог учнів.

1.5 Постановка завдання

Мета роботи – дослідження і розробка ефективного методу прогнозування цін цифрового активу на основі архітектурі глибоких нейронних мереж.

Для досягнення даної мети необхідно виконати наступні кроки:

– проаналізувати літературу і підходи для розв'язання задачі;

– створити власний набір даних для мережі та правильним чином підготовити їх;

– розробити архітектуру мережі та підібрати ваги таки чином, щоб мережа вирішувала поставлене завдання;

– навчити нейронну мережу на підготовленому наборі даних;

– порівняти отримані результати на тестовому наборі даних та провести аналіз точності роботи нейронної мережі.

2 ДОСЛІДЖЕННЯ СПОСОБІВ РОЗРОБКИ НЕЙРОННИХ МЕРЕЖ

2.1 Архітектури нейронних мереж

Є багато різних архітектур нейронних мереж, деякі з них дуже популярні бо показали себе добре у домені обробки природної мови та є ефектні у розв’язані завдань. Найпростішим прикладом є усім відомий перцептрон, познайомитися з ним ви можете на рисунку 2.1.

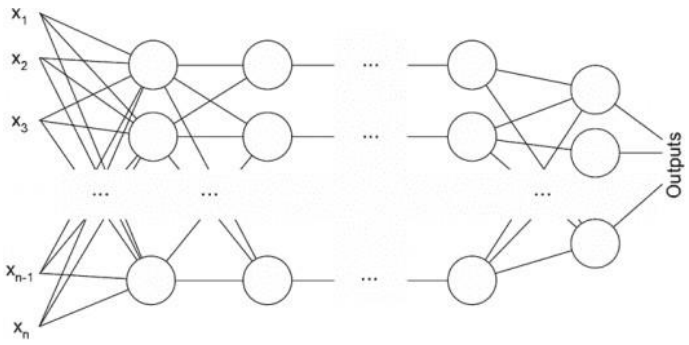


Рисунок 2.1 – Перцептрон

Перцептрон найчастіше зустрічаєтеся з декількома шарами, наприклад з трьома. При цьому усі вузли з'єднані між собою і створюють пов’язану та єдину мережу. Якщо брати до уваги функції активації то це часто тангенціальна, логістична або нелінійна, які в свою чергу дозволяють класифікувати лінійно нероздільні дані. Згадана архітектура крута тим, що показує хороші результати для завдань машинного перекладу.

Тепер обов’язково потрібно згадати згорткову та рекурсивну мережу. У першому випадку нейронна мережа містить один або більше об'єднаних шарів, які є варіацією багатошарового перцептрона, який ми згадали вище. Ці мережеві шари використовують операцію згортки для вхідних даних та відправляють результат в наступний шар. Цей спосіб дозволяє забезпечити меншу кількістю параметрів при той же глибинні.

У другому випадку нейронна мережа сформована при застосуванні тих самих ваг – рекурсивна над структурою. Можна перевірити як це все виглядає на рисунку 2.2. У простому випадку матриця ваг, нелінійність, та функція активації розподілені між всією мережею та об’єднує вузли в спільні об'єкти.

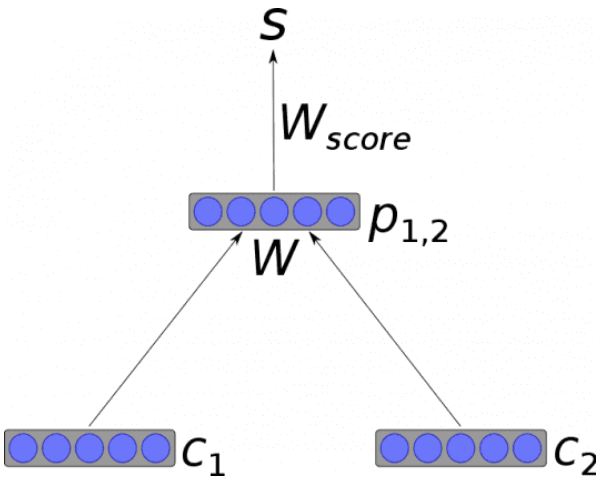


Рисунок 2.2 – Приклад рекурсивної мережи

Є ще цікавих вид, а саме рекурентна нейронна мережа. Її відмінність у тому що вона спрямовано та циклічно працює з в'язками між нейронами. Вихідна інформація залежить від станів нейрона на попередніх кроках, що в свою чергу дозволяє розв'язувати задачі у домені мови та тексту.

Для більш точного моделювання часових послідовностей і їх довгострокових залежностей вигадали мережу з короткостроковою пам’яттю. А різниця цієї мережи між мережою з довго короткостроковою пам’яттю у тому що остання не використовує функцію активації в рекурентних компонентах та збережені значення не модифікує. Отже, пропоную поглянути на рисунку 2.3 з описом вхідних та вихідних блоків та гейту забування, у рамках мережі з довгою короткостровою пам’яттю.

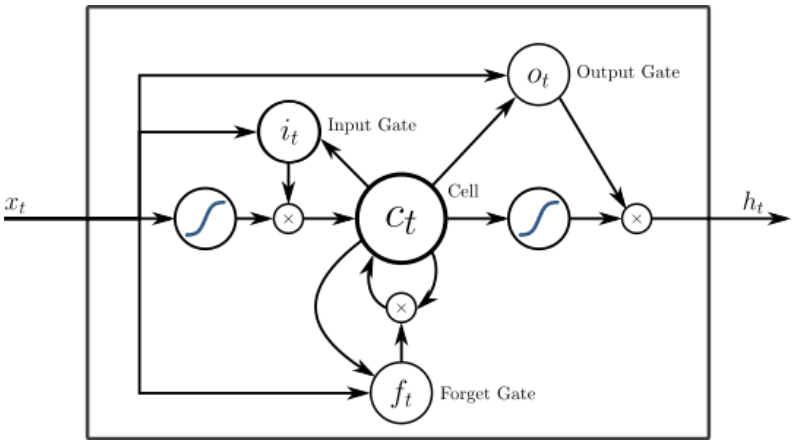


Рисунок 2.3 – Приклад мережі з довгою короткостроковою пам’яттю

Моделі що складаються з більше ніж 2 рекурентних мереж називають послівними. Вони обов’язково включать кодувальник, який обробляє вхідні дані та декодер, який видає фінальний висновок. Ці моделі частіше за все зустрічаються у чат-ботах, FAQ системах та у програмах з перекладом за допомогою машин.

З роблячи висновки можна сформувати бачення того як функціонують нейронні мережі, які різні архітектури існують та як підходять до різних завдань. Для задач класифікації тексту більше за все підходять згорткові нейронні мережі, в той час як рекурентні мережі ефективно працюють з машинним перекладом.

2.2 Математичні методи, що використовуються для навчання

Існує багато методів, що використовуються для навчання. Найбільш цікавий для нас є метод найшвидшого спуску бо його функція втрат наближається до мінімум. Приклад як це виглядає, зображено на рисунку 2.4.

Для знаходження локального мін. функції з використанням градієнтного спуску, необхідно виконати кроки, пропорційні від'ємному значенню градієнта функції в поточній точці [12]. В зворотному випадку

кроки пропорційні позитивному значенню градієнта наближають до локального максимуму функції, і процедура тоді називається підйомом градієнта.

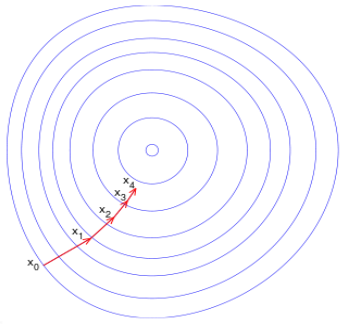


Рисунок 2.4 – Метод найшвидшого спуску

Одним з основних методів навчання нейронних мереж є метод стохастичного градієнтного спуску, який зображено на рисунку 2.5. Пояснення цього методу звучить наступним образом, траєкторія не є прямою, але дозволяє навчати нейронні мережі на частинах даних. Основна проблема градієнтного спуску є в тому що для визначення вагового вектору необхідно провести розрахунки з градієнтом для кожного елемента вибірки. Найчастіше саме це значно уповільнює потужність мережи та коду який творить ці чудеса. Але все ж таки можна збільшити швидкість алгоритму, якщо використати тільки один елемент підвибірки для обчислення нового наближення ваг, але не кожен раз це може призвести до ефективному обчислюванню.

Хочу сказати пару слів про градієнтний метод першого порядку який в свою чергу проводить навчання нейронних мереж. Його використовують бо немає альтернатив для дійсно великої кількість параметрів в нейронної мережі.

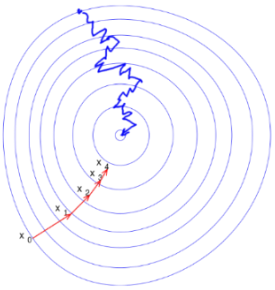


Рисунок 2.5 – Приклад стохастичного градієнтного спуску, який зображений синім кольором

У випадку коли крок навчання підібрано не точно при використовувані градієнтного спуску, він може розходитися або сходитися досить повільно. Саме тому є безліч альтернатив, що дозволяють покращити навчання і позбавити розробника від необхідності затратити багато часу на підбір гіперпараметрів. Бо вони часто розраховують градієнти більш ефективно і динамічна змінюють сам крок ітерації.

З боку альтернатив краще за все розглядати випадковий метод пошуку або метод пошуку перебором. У деяких випадках також є сенс розглянути метод стрибка в басейн та метод Адама.

Для цільових функцій краще використовувати щось інше та звичайне. Якщо ж цільова функція є замудрованою то знайти властивості дуже складно, а отже і напрямок визначити неможливо. Побудувати щось непрацююче дуже легко, навіть у випадку розглянутого вище. Отже навіщо нам цей випадковий пошук, бо він ефективно підбирає способи оптимізації які здатні так чи інакше виконати поставлене завдання. Якщо є завдання поєднати декілька методів інших типів з методом пошуку то це можливо зробити і немає ніяких перешкод у цьому. Мерщій розглянути більш детально ті методи які були згадані.

Перший з них це метод пошуку перебір – він є простим для знаходження значень функцій за допомогою одного з критеріїв порівняння.

Другий це метод стрибка в басейн – він є двофазним з поєднанням в собі покрокового алгоритму з мінімізацією. Цей метод чудово імітує природний процес мінімізації енергії кластерів атомів.

Ще один, це методи підпростору Крилова. Вони працюють формуючи основу послідовності ступенів матриці помноженої на початковий залишок.

Метод Адама став відомий 7 років тому, він показує високу спроможність для малої кількості пам'яті. Також цей метод можна використовувати для завдань с гігантською кількістю обчислювань. Метод також підходить для цілей і завдань з різними типами градієнтів.

У сучасному підході не потрібно створювати нейроні мережі з чистого листа бо є дуже багато відомих бібліотек від Google та інших талановитих фахівців, які в свою чергу значно спрощують цей процес. Найчастіше ці бібліотеки надають високоякісні та перевірені кусочки коду для навчання мереж. У рамках моїх досліджень я використовую готові функції у Tensorflow для обчислювання та Keras для того щоб точно настроювати параметри для дослідженого методу прогнозування цін цифрового активу. Написав весь цей код я вказав зі скількох шарів складається нейронна мережа, які використовуються функції активації, оптимізації для зменшення помилки. Також було побудовано багато графіків за допомогою цих інструментів.

2.3 Функції активації нейронної мережі

Гарною практикою є приділення достатньої уваги до функцій активації при розробці дизайну нейронної мережі. Це обумовлено тим, що вибір функції активації має великий вплив на можливості та продуктивність нейронної мережі, і різні функції активації можуть використовуватися в різних частинах моделі.

Типи шарів також грають роль при виборі функції активації, а саме чи є шар вхідним вихідним, який отримує необроблені вхідні дані з домену, чи є шар прихованим, який бере вхідні дані з іншого шару і передають вихід на

інший шар, та чи є шар вихідним, який робить прогноз. Незважаючи на це є сучасна функція активації за замовчуванням для прихованих шарів - функція ReLU.

Вибір функції активації в прихованому шарі буде керувати тим, наскільки добре модель засвоює навчальний набір даних. Вибір функції активації на вихідному шарі визначить тип прогнозів, які може зробити модель.

Функція активації в нейронній мережі визначає, як зважена сума вхідних даних перетворюється на вихід з вузла або вузлів у шарі мережі.

Технічно функція активації використовується в межах або після внутрішньої обробки кожного вузла в мережі, хоча мережі призначені для використання однієї і тієї ж функції активації для всіх вузлів у шарі.

Усі приховані шари зазвичай використовують ту саму функцію активації. Вихідний рівень зазвичай використовує функцію активації, відмінну від прихованих шарів, і залежить від типу передбачення, якого вимагає модель.

Існує багато різних типів функцій активації, які використовуються в нейронних мережах, хоча, можливо, лише невелика кількість функцій використовується на практиці для прихованих і вихідних шарів.

Нейронна мережа може мати нуль або більше прихованих шарів.

Як правило, диференційована нелінійна функція активації використовується в прихованих шарах нейронної мережі. Це дозволяє моделі вивчати більш складні функції, ніж мережа, навчена за допомогою функції лінійної активації.

Найбільш часто використовуваними є три функції активації для застосування в прихованих шарах: виправлена лінійна активація (ReLU), логістика (сигмоїдальна), гіперболічний тангенс (Tanh).

Випрямлена функція лінійної активації, або функція активації ReLU, є, мабуть, найпоширенішою функцією, яка використовується для прихованих шарів, оскільки є простою у реалізації та ефективною у подоланні обмежень інших раніше популярних функцій активації, таких як

Sigmoid і Tanh. Зокрема, ця функція є менш чутливою до зникаючих градієнтів, які заважають навчанню глибоких моделей, хоча має свій список потенційних перепон, наприклад, насичених або «мертвих» одиниць.

Якщо вхідне значення (x) є негативним, то повертається значення 0,0, інакше повертається значення. Це зображено на рисунку 2.6.

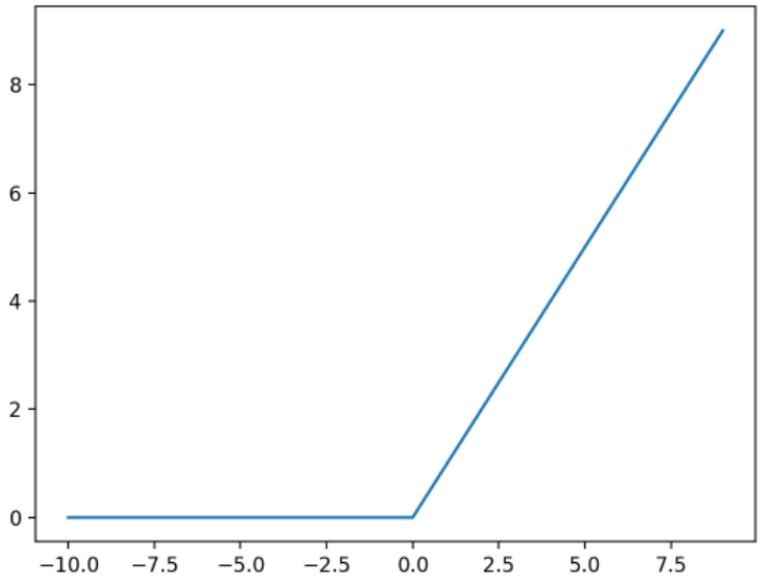


Рисунок 2.6 – Графік залежності входів від виходів для функції активації ReLU

Симоїдальну функцію активації також називають логістичною функцією. Вона також використовується в алгоритмі класифікації логістичної регресії.

Функція приймає будь-яке реальне значення в якості вхідних даних і виводить значення в діапазоні від 0 до 1. Чим більше вхідний сигнал (більш позитивний), тим ближче вихідне значення буде до 1,0, тоді як чим менше вхід (більш негативний), тим ближче вихід буде 0,0. Це зображено на рисунку 2.7.

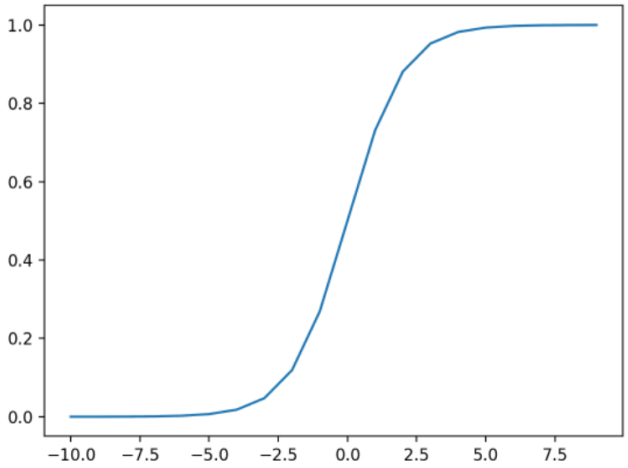


Рисунок 2.7 – Графік входів і виходів для функції активації сигмоїдальної форми

Функція активації гіперболічного тангенса. Графіком є знайома S-подібна форма функції активації Tanh, схожа до сигмоїдальної.

Функція приймає будь-яке дійсне значення в якості вхідних даних і виводить значення в діапазоні від -1 до 1. Чим більше вхід (більш позитивний), тим ближче вихідне значення буде до 1,0, тоді як чим менше вхід (більш негативний), тим ближче вихід буде до -1,0. Це зображено на рисунку 2.8.

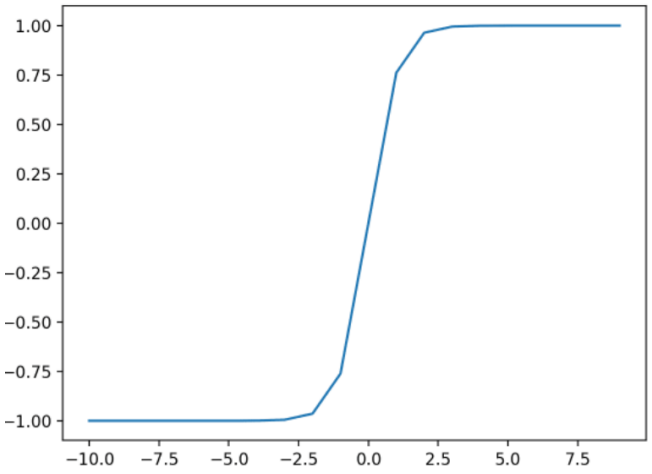


Рисунок 2.8 – Графік залежності входів від виходів для функції активації Tanh

Нейронна мережа майже завжди матиме однакову функцію активації у всіх прихованих шарах. Традиційно, сигмоїдальна функція активації була функцією активації за замовчуванням у 1990-х роках. З середини до кінця 1990-х до 2010-х років функція Tanh була функцією активації за замовчуванням для прихованих шарів.

Функція активації, яка використовується в прихованих шарах, зазвичай вибирається на основі типу архітектури нейронної мережі.

На рисунку 2.9 показано, як вибрати функцію активації для прихованих шарів моделі нейронної мережі.

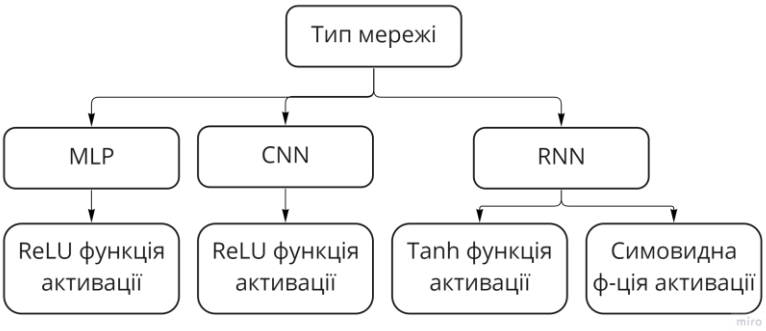


Рисунок 2.9 – Як вибрати функцію активації прихованого шару

Вихідний шар – це шар моделі нейронної мережі, який безпосередньо виводить прогноз. Усі моделі нейронних мереж із прямим зв’язком мають вихідний рівень.

Найбільш часто використовуваними є три функції активації для застосування на вихідному рівні: лінійний, логістика (сигмоїдальна), softmax.

Лінійна функція активації також називається «ідентичність» помножена на 1,0 або «без активації». Це пов’язано з тим, що функція лінійної активації жодним чином не змінює зважену суму вхідних даних і натомість повертає значення безпосередньо. Це зображено на рисунку 2.10.

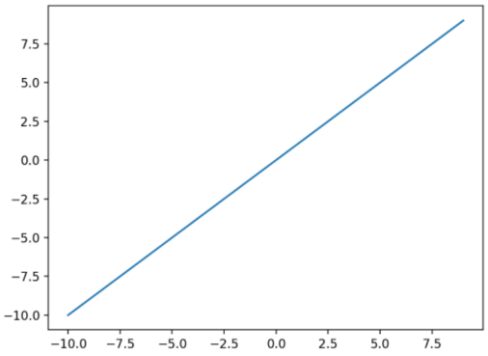


Рисунок 2.10 – Графік залежності входів від виходів для функції лінійної активації

Цільові значення, які використовуються для навчання моделі з лінійною функцією активації на вихідному шарі, зазвичай масштабуються перед моделюванням за допомогою перетворення нормалізації або стандартизації. Типи проблем зображено на рисунку 2.11.

Функція softmax виводить вектор значень, які досягають суми 1,0, які можна інтерпретувати як ймовірності приналежності до класу.

Це пов’язано з функцією argmax, яка виводить 0 для всіх параметрів і 1 для вибраного параметра. Softmax – це м’яка версія argmax, яка дозволяє отримати схожий на ймовірність вихід функції переможець отримує все.

Таким чином, вхід до функції є вектором дійсних значень, а вихідним є вектор такої ж довжини зі значеннями, які сумують до 1,0, як ймовірності.

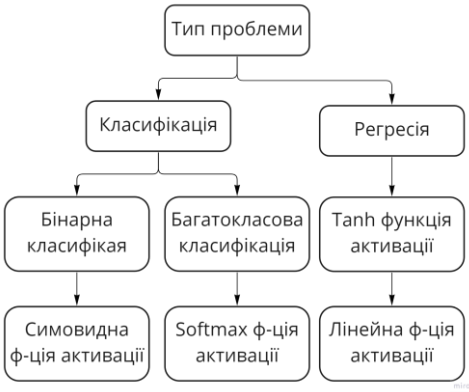


Рисунок 2.11 – Як обрати функцію активації вихідного рівня

2.4 Бібліотеки для побудови нейронної мережі

TensorFlow використовується як платформа для розробки алгоритмів глибокого навчання, зокрема для:

– рекомендації: вибір найкращих шаблонів, які можна рекомендувати користувачам електронною поштою в різних країнах, в яких компанія працює, підтримується понад 100 мов;

– класифікація відгуків користувачів: коли користувачі надають відгуки, алгоритми обробки природної мови, реалізовані в TensorFlow і Keras, використовуються для класифікації проблем, щоб зацікавлені сторони могли визначити основні проблеми з випуском продукту/продукту;

– навчання для визначення рейтингу для пошуку: є певний розвиток щодо покращення результатів пошуку шляхом переходу на алгоритми глибокого навчання з алгоритмів, що підвищують градієнт, і TensorFlow надає цю можливість;

– комп'ютерний зір: деякі експерименти, проведені з виявленням об'єктів та класифікацією зображень.

На рисунку 2.12 наведено логотип TensorFlow.

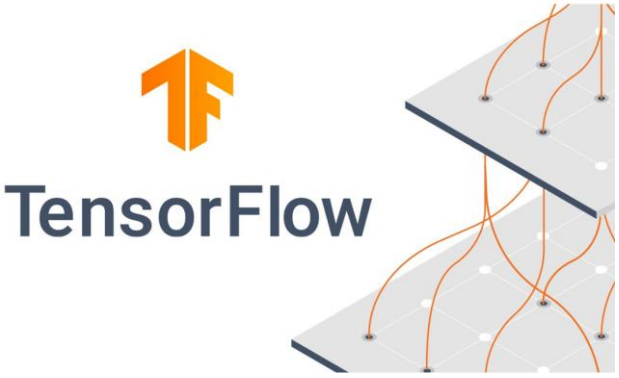


Рисунок 2.12 – Логотип TensorFlow

Для того щоб впевнитися що ця бібліотека нам підходить, пропоную розглянути її плюси та мінуси.

TensorFlow досить простий у використанні, з достатніми підручниками, щоб будь-який користувач швидко розпочав роботу.

Інструменти навколо TensorFlow, такі як TensorBoard, є золотим стандартом. Він значно полегшив процес навчання та налагодження порівняно з більшістю інших платформ глибокого навчання.

Спільнота підтримка TensorFlow дуже добра. Якщо є проблема, зазвичай можна знайти відповідь, просто погугливши. Також документація для TensorFlow часто є на вищому рівні.

До TensorFlow 2.0 налаштувати прийом даних для TensorFlow може бути величезною проблемою. Настільки, що TensorFlow Lite та альтернативи, такі як Keras, роблять його більш привабливим. Проте з TensorFlow 2.0 все змінюється.

Деякі повідомлення про помилки від TensorFlow можуть бути досить складними для розуміння. Наприклад, нещодавня помилка використання шару крапкового добутку в TensorFlow 2.0 зробила враження, що виникла проблема з надходженням даних, але після зниження до TensorFlow 1.14.0 проблема зникає.

Інструменти з Bazel, як наш вибір для інструменту збірки, у нашому монорепо – це трохи кошмар, частково тому, що Bazel погано підтримує Python. Проте ми змогли легко інтегрувати PyTorch з Bazel, але не з TensorFlow.

Хотілося б мати кращі зв’язки з JVM, а не тільки з Python, враховуючи, що багато компаній мають стек на основі JVM, що полегшує його інтеграцію.

TensorFlow чудово підходить для більшості цілей глибокого навчання. Особливо це стосується двох областей:

– комп'ютерний зір: класифікація зображень, виявлення об'єктів та генерація зображень через генеративні змагальні мережі;

– обробка природної мови: класифікація та генерація тексту.

Хороша підтримка спільноти часто означає, що багато готових моделей можна використовувати для швидкого підтвердження концепції або перевірки ідеї. Це, а також просування Google Colab означає, що ідеями можна ділитися досить вільно. Навчання, візуалізація та налагодження моделей дуже легко в TensorFlow, порівняно з іншими платформами.

З точки зору виробництва, це трохи змішане. Велика частина нашого створення функцій виконується через Apache Spark. Це означає необхідність конвертувати файли Parquet (оптимізовані для стовпців) у формат, зручний для TensorFlow, тобто protobufs. Відсутність хороших прив'язок JVM означає, що наші проекти в кінцевому підсумку є сумішшю Python і Scala. Це ускладнює повторне використання деяких інструментів і підтримки. Ось де MXNet сяє краще.

Підтримка TensorFlow спільнотою є чудовою. Існує величезна спільнота, яка справді любить платформу, і є багато прикладів розвитку в TensorFlow. Часто, коли публікується нова гарна техніка, незабаром з'являється реалізація TensorFlow. Це дає змогу швидко об'єднати новітні технології від наукових кіл до систем виробничого рівня. Інструменти навколо TensorFlow також добре. TensorBoard був настільки корисним інструментом, що я не можу уявити, як важко було б налагодити глибоку нейронну мережу, яка вийшла з ладу без TensorBoard.

Я думаю, що TensorFlow дуже добре підходить для людей, які хочуть зануритися глибше і мати повний контроль над рівнем деталей NN. Це готовий до виробництва дизайн і підтримує розподілене середовище, тому він дуже хороший для зрілого та корпоративного виробництва.

2.5 Методи для запобігання перенавчання в нейронних мережах

Пригадаємо що ж таке перенавчання – це явище, при якому побудована мереже добре орієнтується у прикладах з навчальної множини та при цьому зовсім погано працює з тими вхідними даними які були подані в мережу зі сторони. Це відбувається через те, що мережа дуже довгий час

бачила лише дані з навчання і як тільки вона бачить щось інше вона не впізнає жодних знайомих ознак та не шукає відповідність закономірностям. Тобто модель зовсім втрачає здатність до аналізу та узагальненню, а отже на практиці вона виступає у ролі музейного експонату на який можна тільки дивитись. Особливо цікаво дивитися на маленький процент помилки.

Існує декілька способів натрапити на перенавчання, перший з них це надмірно підігнати параметри моделі до специфічних залежностей, що є в навчанні. Добре відомо що описана вище проблема зустрічаєтеся у багатьох видів моделей навчання, але найбільш небезпечно саме у нейронних мережах.

Якщо більш детально зазирнути у причину, то можна зрозуміти що досить часто у процесі навчання трапляється підгонка ваг нейронної мережі, для того щоб вона перетворювала входи до бажаних виходів відповідно до ознак які були виявлені у даних.

Других спосіб є в тому, щоб навчати мережу на занадто великої кількості ваг бо з певного моменту мережа починає підлаштовуватися під особливості окремих прикладів або краще сказати даних, які можуть містити аномальні значення або якісь помилки. У підсумку модель зможе розпізнати новеспостереження тільки в тому випадку, якщо воно співпаде з одним з навчальних прикладів.

Для уникнення перенавчання, можна використовувати такі круті правила:

– ведення перевірочного набору даних, який формується з певного проценту навчального набору даних. Найчастіше цей процент дорівнює 15%. Ці дані подаються на вхід мережі між навчальними прикладами, причому коригування ваг непотрібне. Помилка зменшується на перевірочному та на навчальному наборі даних. Треба бути уважним, що зі збільшенням кількості епох помилка буде рости. Це сигналізує про початок перенавчання та необхідності примусово зупинити мережу та зменшити кількості епох та трохи погратися с параметрами навчання;

– ведення перехресної перевірки, в якому дані поділяються на k блоків рівного розміру. Навчання запускається на k-1 блоках, тестування на k-м. Процедура повторюється визначену кількість разів. При цьому в межах тестування беруться блоки які були заздалегідь залишені для нього;

– змінювати конфігурацію моделі. Рекомендується обирати конфігурацію мережі, в якій кількість параметрів моделі була в 3 рази менше числа прикладів навчальної вибірки. Якщо недотримуватися цього то число параметрів моделі і навчальних прикладів буде однаковим, та модель запам'ятає усі закономірності для вхідних та вихідних множини. Як у результаті буде намагатися постійно їх відтворювати.

У проекті стосовно прогнозування цін цифрового активу використовуються функція дропаут, як метод запобігання перенавчання. Побачити графічне відтворювання цієї функції можна на рисунку 2.13. Ліворуч зображена мережа до, праворуч зображена мережа після використання дропаун функції.

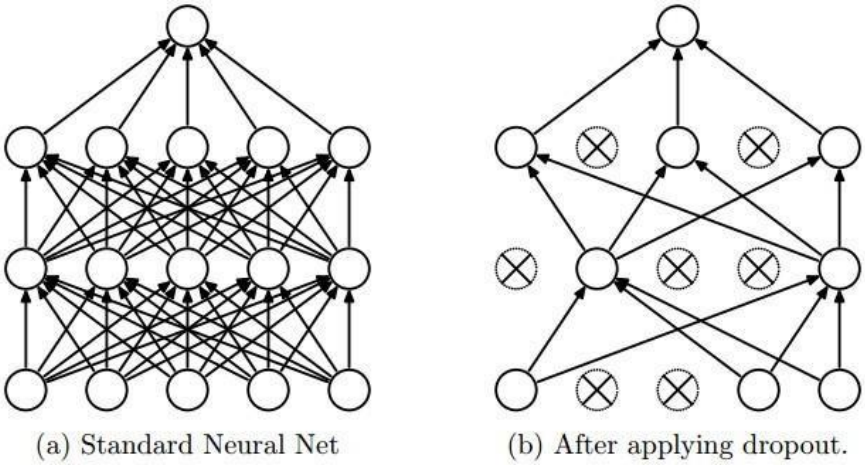


Рисунок 2.13 – Приклад функції дропаут

Головна ідея дропаут – навчити групу глибоких нейронних мереж, а потім підвести результати як усереднене значення.

Мережі для навчання створюються за допомогою виключення з мережі нейронів з ймовірністю p, таким чином, вірогідність того, що нейрон залишиться в мережі становить за формулою 2.1:

$$q = 1 - p, \tag{2.1}$$

де q – вірогідність того, що нейрон залишиться в мережі;

p – ймовірність [17].

Виняток нейрона означає, що при будь-яких вхідних даних або параметрах він повертає 0.

Виключені нейрони не вносять свій внесок в процес навчання на жодному з етапів алгоритму зворотного поширення помилки, тому виключення хоча б одного з нейронів рівносильно навчанню нової нейронної мережі [18].

Дропаут зустрічається у 2 видах, а саме прямий та зворотній. Шар на окремому нейроні може бути представлений як випадкова величина з розподілом Бернуллі. На безлічі нейронів цей шар може виглядати як випадкова величина з біноміальним розподілом. Незважаючи на те, що ймовірність того, що з мережі буде вимкнено рівно np нейронів. Зворотній Dropout збільшує швидкість навчання [19].

Дропаут корисний та дієвий разом з іншими методами нормалізації, що обмежують значення параметрів. Це є важливим для спрощення процесу вибору швидкості навчання, бо дропаут допомагає запобігти проблемі навчання в глибоких нейронних мережах.

Дропаунт часто використовується на практиці бо э ефективним при адаптаціїнейронів на етапі навчання.

3 ДОСЛІДЖЕННЯ І РОЗРОБКА ЕФЕКТИВНОГО МЕТОДУ ПРОГНОЗУВАННЯ ЦІН ЦИФРОВОГО АКТИВУ

3.1 Підготовка даних

Перш за все, необхідно створити дані для навчання нейронної мережі. Для цього візьмемо ціни такої скромної криптовалюти як Bitcoin з 2014 до сьогодні. Ці дані розповсюджуються по підписці на такому відомому сервісі як Yahoo Finance, приклад того як виглядає сервіс можна побачити на рисунку 3.1.

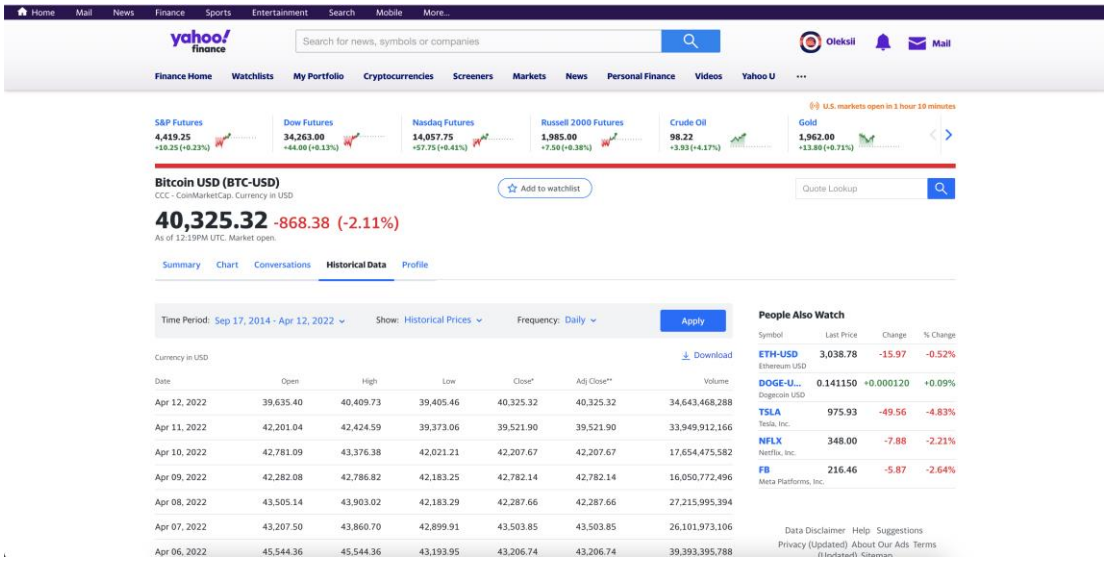
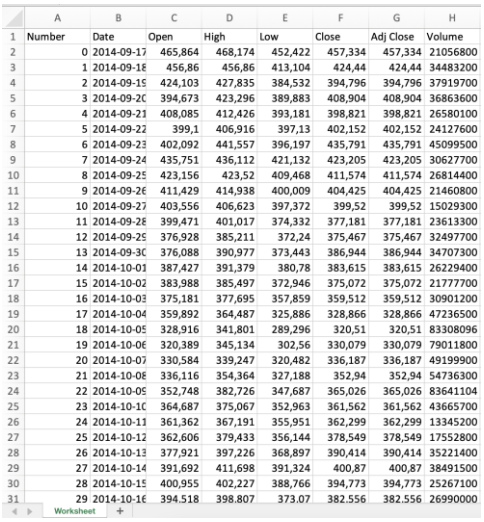


Рисунок 3.1 – Зображення даних у Yahoo Finance

Дані представлені у форматі .csv, що є дуже зручно для нас. Отже, завантажимо дані і подивимося, як вся ця краса виглядає на рисунку 3.2.



	A	B	C	D	E	F	G	H
1	Number	Date	Open	High	Low	Close	Adj Close	Volume
2	0	2014-09-17	465,864	468,174	452,422	457,334	457,334	21056800
3	1	2014-09-18	456,86	456,86	413,104	424,44	424,44	34483200
4	2	2014-09-19	424,103	427,835	384,532	394,796	394,796	37919700
5	3	2014-09-20	394,673	423,296	389,883	408,904	408,904	36863600
6	4	2014-09-21	408,085	412,426	393,181	398,821	398,821	26580100
7	5	2014-09-22	399,1	406,916	397,13	402,152	402,152	24127600
8	6	2014-09-23	402,092	441,557	396,197	435,791	435,791	45099500
9	7	2014-09-24	435,751	436,112	421,132	423,205	423,205	30627700
10	8	2014-09-25	423,156	423,52	409,468	411,574	411,574	26814400
11	9	2014-09-26	411,429	414,938	400,009	404,425	404,425	21460800
12	10	2014-09-27	403,556	406,623	397,372	399,52	399,52	15029300
13	11	2014-09-28	399,471	401,017	374,332	377,181	377,181	23613300
14	12	2014-09-29	376,928	385,211	372,24	375,467	375,467	32497700
15	13	2014-09-30	376,088	390,977	373,443	386,944	386,944	34707300
16	14	2014-10-01	387,427	391,379	380,78	383,615	383,615	26229400
17	15	2014-10-02	383,988	385,497	372,946	375,072	375,072	21777700
18	16	2014-10-03	375,181	377,695	357,859	359,512	359,512	30901200
19	17	2014-10-04	359,892	364,487	325,886	328,866	328,866	47236500
20	18	2014-10-05	328,916	341,801	289,296	320,51	320,51	83308096
21	19	2014-10-06	320,389	345,134	302,56	330,079	330,079	79011800
22	20	2014-10-07	330,584	339,247	320,482	336,187	336,187	49199900
23	21	2014-10-08	336,116	354,364	327,188	352,94	352,94	54736300
24	22	2014-10-09	352,748	382,726	347,687	365,026	365,026	83641104
25	23	2014-10-10	364,687	375,067	352,963	361,562	361,562	43665700
26	24	2014-10-11	361,362	367,191	355,951	362,299	362,299	13345200
27	25	2014-10-12	362,606	379,433	356,144	378,549	378,549	17552800
28	26	2014-10-13	377,921	397,226	368,897	390,414	390,414	35221400
29	27	2014-10-14	391,692	411,698	391,324	400,87	400,87	38491500
30	28	2014-10-15	400,955	402,227	388,766	394,773	394,773	25267100
31	29	2014-10-16	394,518	398,807	373,07	382,556	382,556	26990000

Worksheet +

Рисунок 3.2 – CSV файл с даними Bitcoin

Вставимо потрібні нам для завантаження бібліотеки у код, як це наведено у лістингу 3.1.

Лістинг 3.1 – Необхідні пакети для роботи скрипта

```
import matplotlib.pylab as plt
import numpy as np
import pandas as pd
```

Спробуємо прочитати дані, озираючись на те що дані представлені у у зворотному порядку – від 2022 до 2014, нам потрібно їх «перевернути» за допомогою кода, як в лістингу 3.2.

Лістинг 3.2 – Код для відображення даних

```
data = pd.read_csv('./data/BTC.csv')[::-1]
close_price = data.ix[:, 'Adj Close'].tolist()
plt.plot(close_price)
```

Вигляд самих даних можна побачити на рисунку 3.3, який зображених трохи нижче.

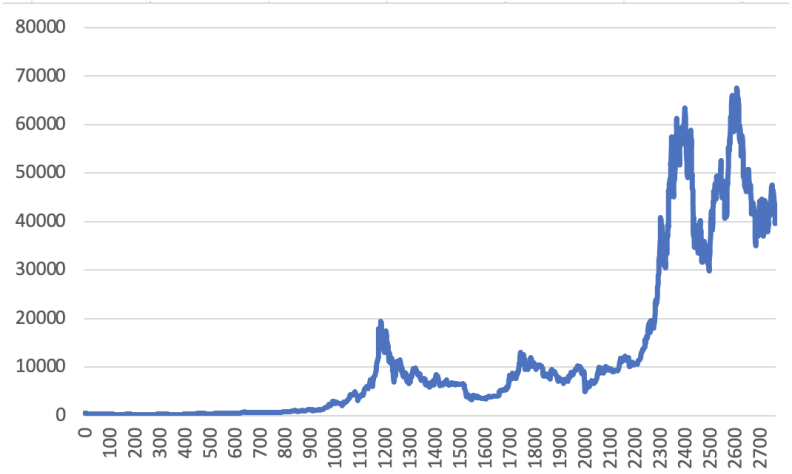


Рисунок 3.3 – Зображення даних з дата сету

Проаналізувавши цей рисунок можна уявити з яким випадковим процесом ми будемо мати справу. Перш ніж вирішувати задачу прогнозування на день або декілька днів вперед необхідно завдання «прогнозування» описати ближче до завдань машинного навчання. З одного боку це завдання можна розглянути як рух ціни на ринку – більше чи менше. Це буде завдання бінарної класифікації. З іншого можна розглянути як передбачення зміни ціни в наступний день порівняно з останнім днем або ж логарифм від цієї різниці. Тобто як завдання регресії. Той чи інший бік завдання має плюси та мінуси які ми розглянемо трохи нижче. Разом з такими важливими моментами як нормалізація даних.

Що у випадку класифікації, що у разі регресії, на вхід ми візьмемо якесь вікно часового ряду наприклад, 30 днів і зосередимося передбачити або рух ціни наступного дня як класифікація, або значення зміни як регресія.

Основна проблема фінансових часових рядів – вони взагалі не стаціонарні. Це можна перевірити самому за допомогою, скажімо, тесту Дікі-Фуллера. Їх характеристики, як мат. очікування, дисперсія, середнє максимальне та мінімальне значення у вікні змінюються з часом, що означає, що по-хорошому ми не можемо використовувати ці значення для MinMax або Z-score нормалізації по наших вікнах. Тому що характеристики можуть змінитися вже наступного дня або змінитись посередині обраного

вікна.

Але все ж таки якщо уважно подивитися на завдання класифікації, нас не дуже цікавить мат. очікування або дисперсія наступного дня, нас цікавить виключно рух вгору або вниз. Тому ми ризикнемо, і нормалізуватимемо наші 30-денні вікна за допомогою Z-score, але тільки їх, не торкаючись нічого з «майбутнього», для цього напишемо код, як у лістингу 3.3.

Лістинг 3.3 – Код для нормалізаціх вікна даних

```
X = [(np.array(x) - np.mean(x)) / np.std(x) for x in X]
```

Для завдання регресії так уже зробити не вийде, адже якщо ми також відніматимемо середнє і ділитимемо на відхилення, нам доведеться відновлювати це значення для значення ціни наступного дня, а там вже ці параметри можуть бути зовсім іншими. Тому ми спробуємо два варіанти: навчити на необроблених даних і спробуємо обдурити систему, взявши відсоткову зміну ціни наступного дня – з цим нам допоможе pandas, як у лістингу 3.4.

Лістинг 3.4 – Використання pandas

```
close_price_diffs = close.price.pct_change()
```

Тепер дані, отримані без маніпуляцій зі статистичними характеристиками, вже лежать у межі від -0.5 до 0.5.

3.2 Створення різних наборів даних для мережі

Тепер потрібно створити 2 типу даних, а саме навчальну та тренувальну вибірку. Для першої візьмемо 85% вікон у часі для навчання та останні 15% для другої для того щоб, перевірити роботу нейронної мережі.

Тому для навчання нейронної мережі ми отримаємо наступні пари X, Y: ціни в момент закриття ринку за 30 днів і [1, 0] або [0, 1] залежно від того,

чи зросло або впало значення ціни для бінарної класифікації, процентна зміна цін за 30 днів та зміна наступного дня для регресії.

3.3 Створення архітектури мережі та підбір ваг таким чином, щоб мережа вирішувала поставлене завдання

Найголовніше зараз для нас це побудувати правильну архітектуру нейронної мережі. Для цього будемо використовувати багатошаровий перцептрон та фреймворк для імплементації Keras. Він дуже простий, інтуїтивно зрозумілий і з ним можна реалізовувати досить складні обчислювальні графи майже немаючі часу. Почнемо наш експеримент з реалізації простенької мережи з вхідного шара з 30 нейронами як довжина нашого вікна, першим прихованим шаром з 64 нейронами, після нього BatchNormalization бо його рекомендується використовувати практично для будь-яких багатошарових мереж, потім активаційна функція LeakyReLU, бо вона значна виграє у ефективності в порівняно з архаїчною функцією ReLU. На виході розмістимо один нейрон або два для класифікації, який в залежності від задачі класифікація або регресія буде мати softmax на виході, або залишимо його без нелінійності, щоб мати можливість прогнозувати будь-яке значення, та напишімо код, як у лістингу 3.5.

Лістинг 3.5 – Код простої архітектури для класифікації

```
model = Sequential()
model.add(Dense(64, input_dim=30))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Для завдання регресії в кінці параметр активації має бути linear. Далі нам потрібно визначити функції помилки та алгоритм оптимізації. Не

вдаючись до деталей варіацій градієнтного спуску візьмемо Adam з довжиною кроку 0.001. Параметр loss для класифікації необхідно поставити крос-ентропію – categorical_crossentropy, а регресії – середню квадратичну помилку таку як mse. Також Keras дозволяє нам досить гнучко контролювати процес навчання, наприклад, хороша практика зменшувати значення кроку градієнтного спуску, якщо наші результати не покращуються. Саме цим і займається ReduceLROnPlateau, який ми додали як колбек у навчання моделі за допомогою коду, як в лістингу 3.6.

Лістинг 3.6 – Код для регресії

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
factor=0.9, patience=5, min_lr=0.000001, verbose=1)
model.compile(optimizer=opt,
loss='categorical_crossentropy',metrics=['accuracy'])
```

3.4 Навчання нейронної мережі та експерименти

Тепер спробуємо підібрати правильну кількість епох для навчання та вибірку. Наприклад найкращі результати зустрічаються при 50 епох и при вибірці в 128, тому напишемо код, як у лістингу 3.7 та почнемо навчання.

Лістинг 3.7 – Код для навчання нейронної мережі

```
history = model.fit(X_train, Y_train,
nb_epoch = 50,
batch_size = 128,
verbose=1,
validation_data=(X_test, Y_test),
shuffle=True,
callbacks=[reduce_lr])
```

Після завершення процесу навчання буде непогано вивести на екран графіки динаміки значення помилки і точності за допомогою коду, як у лістингу 3.8.

Лістинг 3.8 – Код для графіків динаміки значення помилки і точності

```
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
plt.figure()
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel(‘acc’)
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```

Перед запуском навчання хочу звернути увагу на важливий момент, вивчати алгоритми таких даних треба довше, хоча б 50–100 епох. Це пов'язано з тим, що якщо обрати, скажімо, 5–10 епох і побачите 55% точності, це швидше за все не буде означати, що мережа навчилися знаходити патерни. Наприклад якщо провести аналіз тренувальних даних, буде видно, що 55% вікон були для одного патерна як підвищення, а решта 45% – для іншого як зниження. У нашому випадку 53% вікон класу зниження, а 47% як підвищення, тому треба намагатися отримати точність вище 53%, яка і говоритиме про те, що мережа навчилися знаходити ознаки.

Занадто висока точність може означати перенавчання або «заглядання» у майбутнє під час підготовки навчальної вибірки. З цим треба буди дуже обережним и запобігати це за допомогою методів описаних трішки нижче.

З точки зору задача класифікації проведемо навчання нашої першої моделі та подивимося на графіки. Перший графік з моделлю loss, як зображено на рисунку 3.4.

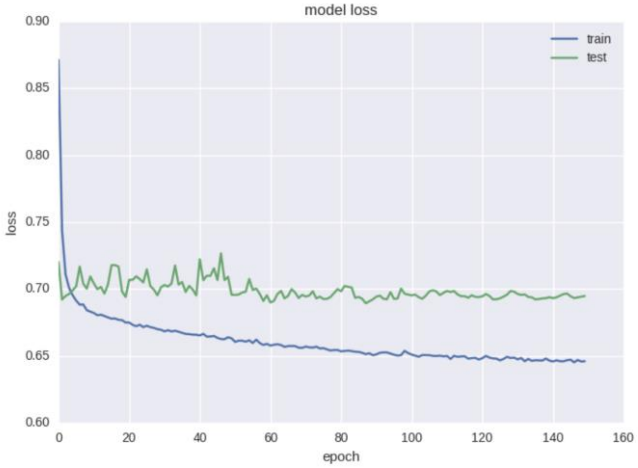


Рисунок 3.4 – Графік loss для першої моделі класифікації

Другий графік з моделлю accuracy, який зображений у рисунку 3.5.

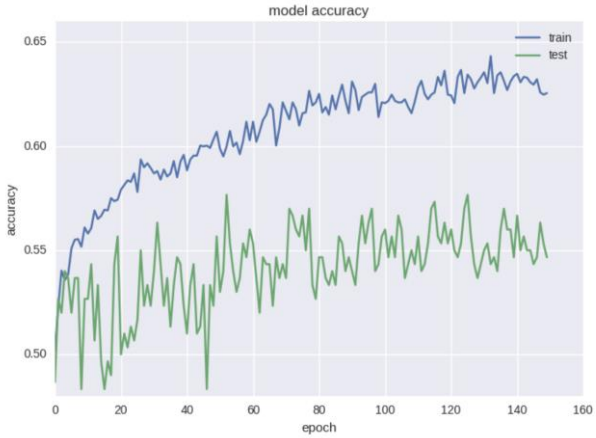


Рисунок 3.5 – Графік accuracy для першої моделі класифікації

Проаналізувавши можна побачити, що точність для тестової вибірки весь час залишається на плюс-мінус одному значенні, а помилка для тренувальної падає, а точність зростає, що говорить нам про перенавчання. Якщо використати більше шарів, як у лістингу 3.9 то це значно покращить нашу модель.

Лістинг 3.9 – Архітектура мережі зі збільшеною кількістю шарів

```
model = Sequential()
model.add(Dense(64, input_dim=30))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax')
```

Результати роботи нового підходу можна перевірити на графіках, які зображені на рисунку 3.6 та на рисунку 3.7.

Рисунок 3.6 – Зображення графіку loss для моделі з двома шарами у випадку класифікації

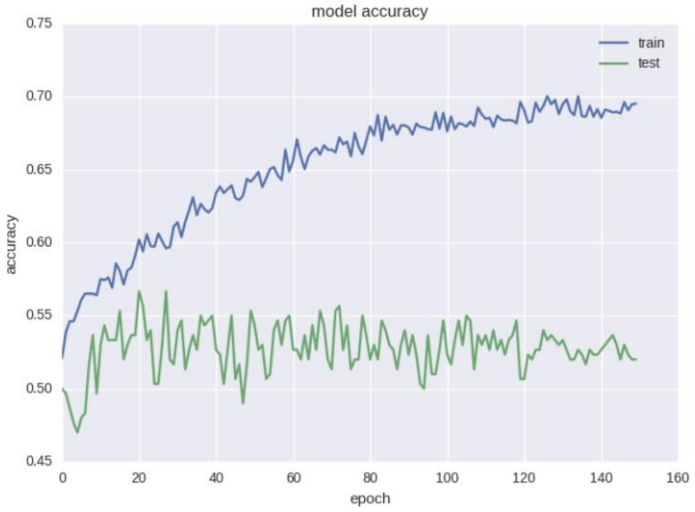


Рисунок 3.7 – Accuracy для моделі з 2 шарами у випадку класифікації

Приблизно така сама картина, якщо ми стикаємося з ефектом перенавчання. Тому нам потрібно додавати регуляризацію до нашої моделі. Під час перенавчання ми будуємо модель, яка просто запам'ятовує тренувальні дані і не дозволяє узагальнити знання. У процесі регуляризації ми накладаємо певні обмеження на ваги нейронної мережі, щоб не було великого розкиду в значеннях і не дивлячись на велику кількість параметрів, тобто ваги мережі, частину з них звернути в нуль для спрощення. Для цього використаємо найпоширеніший спосіб. Додамо до функції помилки регулятор, такий як keras.regularizers.activity_regularizer. Його використання можна побачити у лістингу 3.10.

Лістинг 3.10 – Код для моделі з додатковою L2 нормою

```
model = Sequential()
model.add(Dense(64, input_dim=30,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
```

Продовження лістингу 3.10

```
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Пропоную ще раз переглянути графіки помилки та точності на рисунку 3.8 та рисунку 3.9. Така нейронна мережа вчиться вже трохи краще з точки зору функції помилки, але точність все ще страждає.

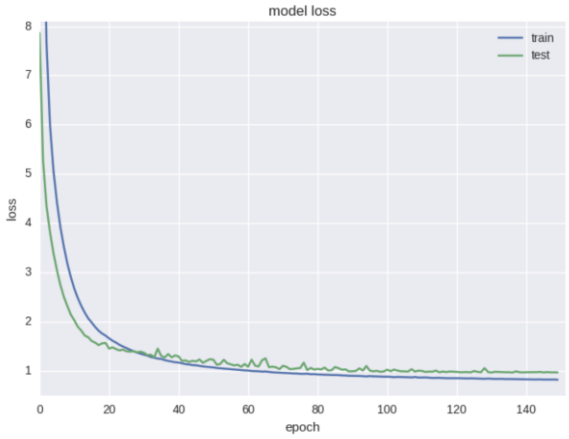


Рисунок 3.8 – Model loss з додатковою L2 нормою

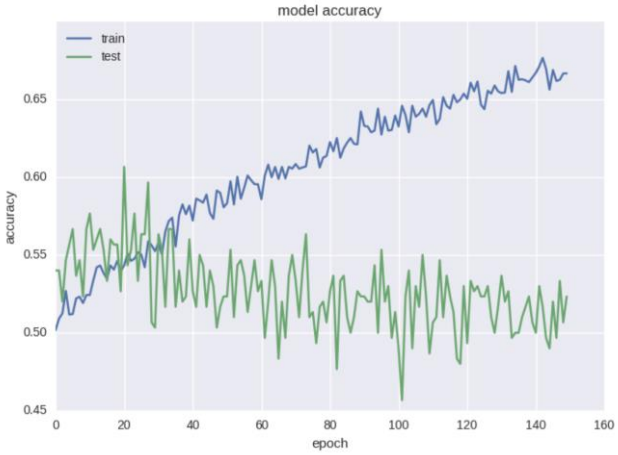


Рисунок 3.9 – Model accuracy з додатковою L2 нормою

Такий дивний ефект як зменшення помилки, але не зменшення точності нерідко зустрічається при роботі з даними великої зашумленості або випадкової природи.

Тому варто додати ще більше регуляризації в нашу модель за допомогою популярної в останні роки техніки Dropout. Який представляє собою випадкове ігнорування деяких ваг у процесі навчання, щоб уникнути коадаптації нейронів. Щоб вони не вивчали однакові ознаки. Змінена модель описана у лістингу 3.11.

Лістинг 3.11 – Приклад моделі з dropout

```
model = Sequential()
model.add(Dense(64, input_dim=30,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dropout(0.5))
model.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Між двома прихованими шарами ми будемо використовувати dropout під час навчання з ймовірністю 50% для кожної ваги. Дропаут зазвичай не додають між вхідним шаром і першим прихованим, так як в цьому випадку ми будемо вчити на зашумлених даних, і також не додається прямо перед виходом. Під час тестування мережі жодного дропауту, зрозуміло, не відбувається. Перевіримо результати на графіках, які зображені на рисунку 3.10 та рисунку 3.11.

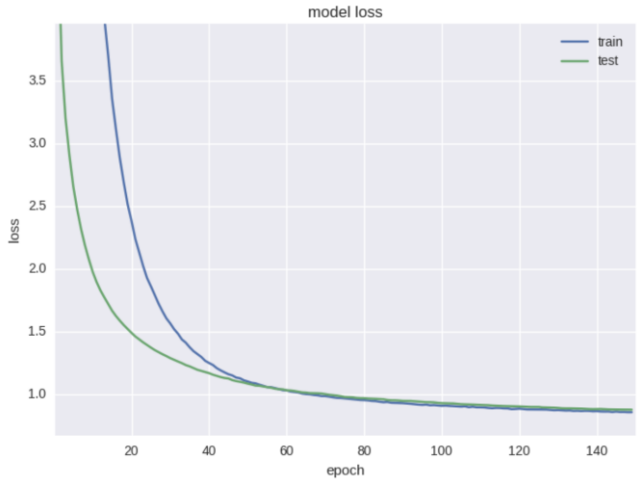


Рисунок 3.10 – Зображення model loss з dropout

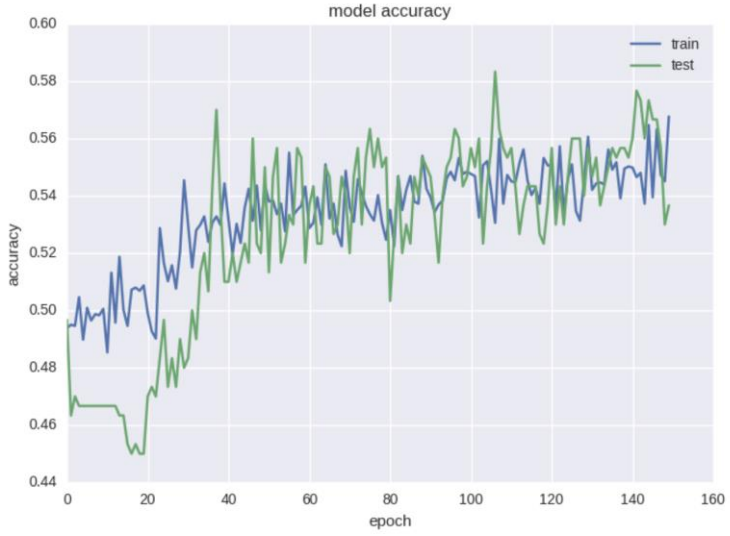


Рисунок 3.11 – Зображення model accuracy з dropout

Як бачимо, графіки помилки та точності адекватні, якщо зупинити навчання мережі трохи раніше, можемо отримати 68% точності передбачення руху ціни, що вже точно краще за випадкове ворожіння.

Ще один цікавий та інтуїтивно зрозумілий момент прогнозування фінансових часових рядів полягає в тому, що коливання наступного дня має випадкову природу, але коли ми дивимося на графіки, свічки, ми таки можемо помічати тренд на наступні 5-10 днів. Давайте перевіримо, чи

можуть з таким завданням впоратися наша нейронка. Спрогнозуємо рух ціни через 5 днів з останньою вдалою архітектурою і заради інтересу навчимо на більшій кількості епох. Графік точності та помилки можна побачити рисунку 3.12 та рисунку 3.13.

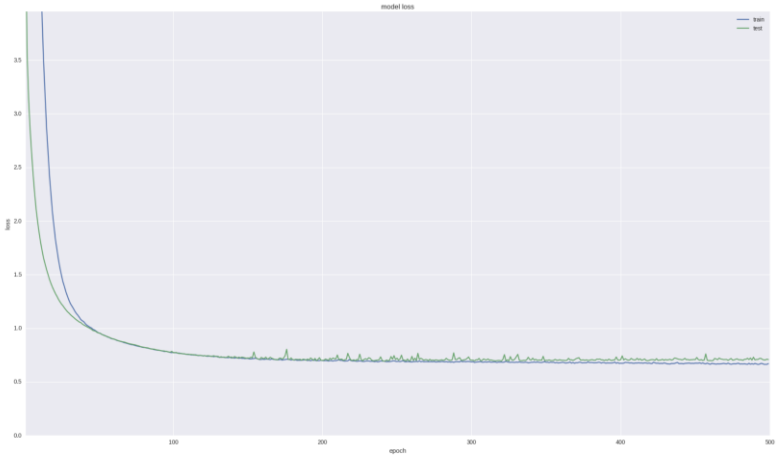


Рисунок 3.12 – Зображення model loss з більшою кількістю епох

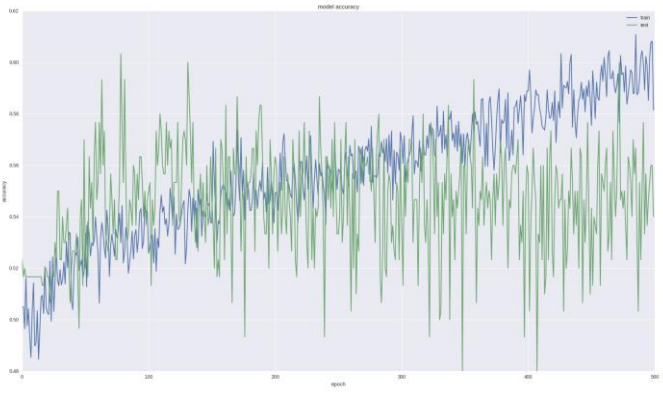


Рисунок 3.13 – Зображення model accuracy з більшою кількістю епох

Як бачимо, якщо ми зупинимо навчання досить рано та отримали 70% точності, що дуже непогано.

Для завдання регресії візьмемо нашу останню успішну архітектуру для класифікації бо вона вже показала, що вміє вивчати потрібні ознаки, приберемо Dropout та навчимо на більшій кількості ітерацій.

Також в даному випадку ми можемо дивитися вже не тільки на значення помилки, а й візуально оцінити якість прогнозування за допомогою коду, який наведений у лістингу 3.12.

Лістинг 3.12 – Код для першою моделі з боку задачі регресії

```
pred = model.predict(np.array(X_test))
original = Y_test
predicted = pred
plt.plot(original, color='black', label = 'Original
data')
plt.plot(predicted, color='blue', label = 'Predicted
data')
plt.legend(loc='best')
plt.title('Actual and predicted')
plt.show()
```

Успішна архітектура усієї мережі для задачі регресії буде виглядати, як наведено у лістингу 3.13.

Лістинг 3.13 – Архітектура мережі для задачі регресії

```
model = Sequential()
model.add(Dense(64, input_dim=30,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(1))
model.add(Activation('linear'))
```

Побачити результати можна на рисунку 3.14.

Рисунок 3.14 – Зображення результатів моделі з боку задачі регресії

Здалеку виглядає непогано, але якщо придивитися, ми побачимо, що наша нейронна мережа просто запізнюється зі своїми передбаченнями, що можна вважати неточністю. Якщо ж навчити на змінах цін, то отримаємо результати які відображені на рисунку 3.15.

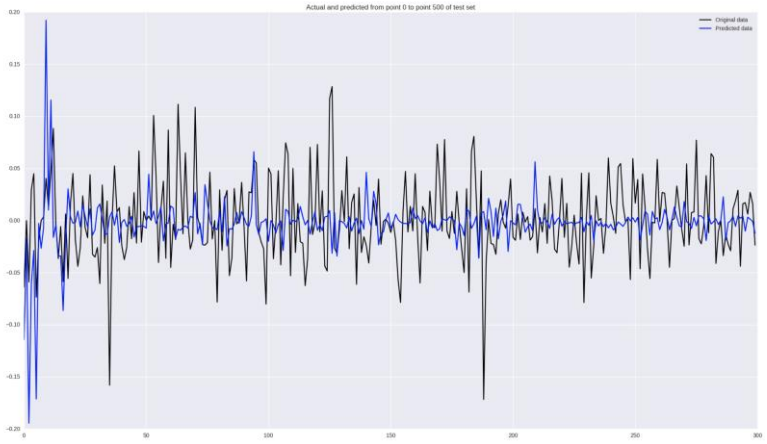


Рисунок 3.15 – Зображення результату мережі з боку класифікації

Використовуючи найуспішнішу архітектуру спробуємо повторити експеримент на іншому дата сеті. Для цього знову зайдемо на Yahoo Finance та завантажимо зміну ціни криптовалюти Ethereum за часовий проміжок з 2015 року до сьогодні. Розділивши дані на таку ж пропорцію як і біткоїн, а

саме 85% на навчання і 15% на перевірку працездатності, запустимо навчання нейронної мережі. При використанні такої же кількості епох як для задачі класифікації так і для задачі регресії отримаємо дуже схожі результати, які зображено на рисунку 3.16 та на рисунку 3.17.

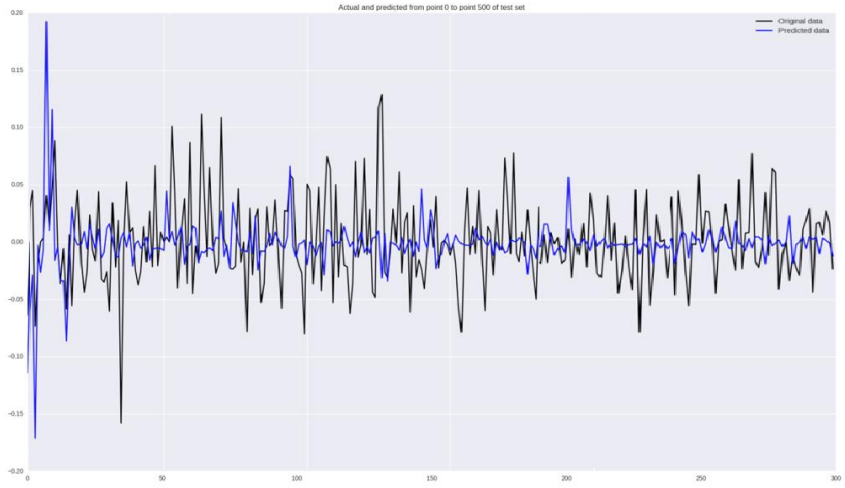


Рисунок 3.16 – Результати навчання на дата сеті Ethereum з боку класифікації

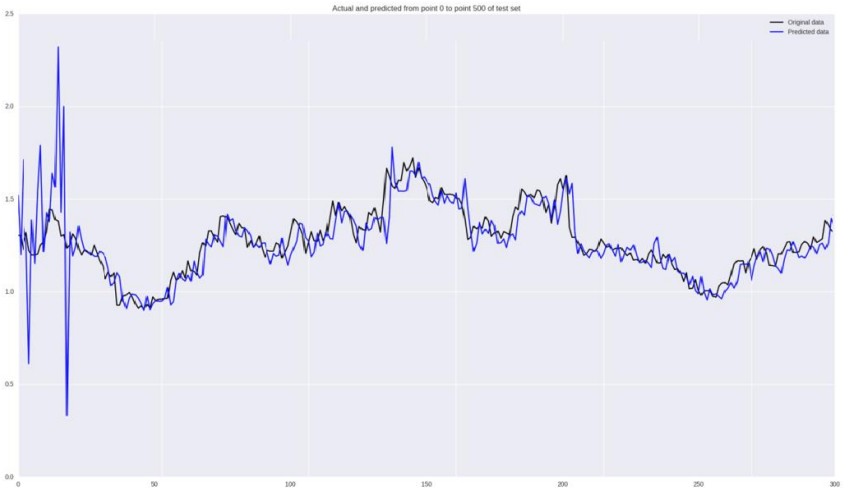


Рисунок 3.16 – Результати навчання на дата сеті Ethereum з боку регресії

А саме 67% точності у випадку задачі класифікації, та 65 відсотків точності у випадку задачі регресії. Причому для останнього проблема запізнення даних залишилися.

3.5 Порівняння отриманих результатів

Отже, виконавши однаковий експеримент з двома дата сетами, ми можемо дійти висновку, що побудована архітектура нейронної мережі непогано передбачає деякі значення та подекуди правильно вгадує тренд. Але все ж таки мережа з боку задачі класифікації краще та показує результати які дійсно можна використовувати для передбачення майбутнього розвитку подій та на основі цього планувати свої інвестиції.

Є ряд кроків, які дозволяють вивести точність на рівень 75-80%, наприклад:

– додати використання сентимент-аналізу, а саме прогноз ціни активу на основі текстових джерел, виходячи з аналізу настрою людей, в каналах новин, соціальних мережах. Це доповнення дозволило б зменшити ризик випадковості;

– навчати на високочастотних даних. Наприклад, оновлення кожну годину або кожні п'ять хвилин. Це забезпечує більше даних, які в свою чергу генерують більше патернів та запобігають меншому перенавчанню;

– використовувати більш сучасні архітектури нейронних мереж, які призначені для роботи з послідовностями. Наприклад, convolutional neural networks або recurrent neural networks;

– використовувати не лише ціну закриття, а всі змінні дані у дата сеті;

– оптимізувати гіперпараметри такі як розмір вікна, кількість нейронів у прихованих шарах та крок навчання;

– використовувати більш підходящі для нашого завдання функції втрат.

ВИСНОВКИ

При підготовці роботи розглянуті основні питання цифрової економіки та необхідність використання саме нейронних мереж для прогнозування зміни ціни цифрового активу. При побудові нейронної мережі розглянуто різні підходи навчання: з вчителем, без вчителя, з підкріпленням. Для розв’язання завдання було використано підхід навчання з вчителем. З метою збудувати найбільш ефективну мережу були проаналізовані різні типи та архітектури нейронних мереж.

Метою роботи зазначено дослідження і розробка ефективного методу прогнозування цін цифрового активу базуючись на глибокої нейронної мережі. Що і було виконано у ході дослідження.

Для досягнення мети було необхідно навчити нейронну мережу прогнозувати зміни цін цифрового активу з точки зору задачі класифікації та з точки зору задачі регресії. У першому випадку намагалися вгадати піде ціна вниз чи вгору, у другому намагалися передбачити число цієї зміни. Дані для навчання були отримані з дата сету, що складався десь з 2700 записів закриття ціни Bitcoin. Та 2490 записів закриття ціни Ethereum. Серед яких було виділено близько 85 відсотків для навчання та 15 відсотків для перевірки роботи нейронної мережі.

У другому етапі досягнення мети необхідно було побудувати архітектуру нейронної мережі, яка складалась з декількох частин. Перша частина являє собою набір шарів мережі для виділення важливих ознак тренду. Друга повнозв'язна частина необхідна для класифікації або регресії. Шар dropout потрібен для регуляризації та запобіганню перенавчання. Вихідний шар використовуємо для прогнозування тренду на наступні 5 днів або прогнозування числа зміни ціни на наступні декілька днів.

Після загрузки дата сету та побудови архітектури почалося навчання нейронної мережі та аналіз результатів.

У ході яких було зрозуміло що знайдені прийоми побудови нейронної мережі можна використовувати для будь-яких часових рядів, головне правило вибрати перед обробку даних, визначити архітектуру мережі та оцінити якість алгоритму.

Було передбачено з точністю до 70% тренд зміни ціни через 5 днів, використовуючи вікно цін у попередні 30 днів, що можна вважати хорошим результатом. Кількісний прогнозом зміни ціни вийшов не достатньо точний, для цього завдання доцільно використовувати більш серйозні інструменти та статистичний аналіз часового ряду. Також були виявлені основні принципи для покращення результатів у випадку виконання деяких модифікацій проекту. Серед яких:

– використання сентимент аналізу на основі текстових джерел;

– навчати на даних, які часто обновлюються і є максимально актуальні на теперішній час;

– використовувати сучасні та найбільш ефективні архітектури нейронних мереж у рамках дії з послідовностями;

– брати до уваги усі дані з дата сету;

– провести більше експериментів з розміром вікна, кількість нейронів у шарах мережі та різою кількістю епох навчання;

– знайти найефективніші функції втрат у рамках нашої задачі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Троценко О. І. Нейромережевий підхід до вирішення задач прогнозування цін цифрового активу. *25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті».* Зб. матеріалів форуму. Т. 6. 2021. С. 41–42.

2. Малыхин Е.М. Алгоритмический трейдинг для профессионалов. СПб.: БХВ-Петербург, 2021. 176 с.

3. Kud A.A. Substantiation of the Term “Digital Asset”: Economic and Legal Aspects. *International Journal of Education and Science.* 2019. Vol.2(1). P. 41-52.

4. Янсен С. Машинное обучение для алгоритмической торговли на финансовых рынках. Практикум. СПб.: БХВ-Петербург, 2020. 560 с.

5. Бринк Х. Ричардс Дж., Феверолф М. Машинное обучение. СПб.: БХВ-Петербург, 2017. 338 с.

6. Шакла Н. Машинное обучение и TensorFlow. СПб.: БХВ-Петербург, 2019. 336 с.

7. Cho K., Van Merriënboer B., Bahdanau D., Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. arXivpreprint arXiv:1409.1259, 2014. 8 p.

8. Trinh Xuan Tuan, Tu Minh Phuong. 3D convolutional networks for session-based recommendation with content features. In RecSys, 2017. 138–146.

9. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. М.: ДМК Пресс, 2018. 652 с.

10. Шапиро Л., Стокман Д. Компьютерное зрение / ред. С. М. Соколова; пер. с анг. А. А. Богуславский 2015. 763 с.

11. Назаров О.С. Теорія прогнозування: навч. посіб. Харків:ХНУРЕ, 2017. 300 с.

12. Mital A. Understanding RNN and LSTM. *Medium*. 2019. URL: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e (Last accessed: 15.04.2022).

13. Marcos Lopez de Prado Advances in Financial Machine Learning. New Jersey : Wiley, cop., 2018. 393 с.

14. Liu D., Peck I., Dangi S. Left Ventricular Ejection Fraction Assessment: Unraveling the Bias between Area- and Volume-based Estimates. *Medical Imaging: Ultrasonic Imaging and Tomography*. 2019.

15. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation International. *Computer Science - Computer Vision and Pattern Recognition.* 2015. pp 234-241.

16. Головко В.А. Нейронные сети: обучение, организация и применение. М.:ИПРЖР, 2001, 256 с.

17. Круглов В.В., Дли М. И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети. *Учеб. пособие.* М.:Физматлит, 2001. 224 с.

18. Browniee J. A Gentle Introduction to Long Short-Term Memory Networks by the Experts. *Long Short-Term Memory Networks.* 2017. URL: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (Last accessed: 16.04.2022).

19. Ярушкина Н.Г. Основы теории нечетких и гибридных систем. М.: Финансы и статистика, 2004. 320 с.

20. Witten H., Eibe Frank and Mark A. Hall Data Mining: Practical Machine Learning Tools and Techniques. 3rd Edition. Morgan Kaufmann, 2011. P. 664.