Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи нейромережевого прогнозування часових рядів

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Соколов І.С.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Руденко О.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

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

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Соколову Івану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи нейромережевого прогнозування часових рядів

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1487 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи

1) Документація мови програмування Python
2) Документація бібліотеки машинного навчання Keras
3) Середовище розробки Jupiter Notebook

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз існуючих методів прогнозування
2) Огляд архітектур нейронних мереж
3) Реалізація нейромережевого прогнозування
4) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Плакати – 10 арк. ф. А4

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури про прогнозування часових	03.11.20 – 9.11.20	
	рядів		
2	Аналіз літератури про нейронні мережі та їх	10.11.20 – 16.11.20	
	архітектури		
3	Огляд документації бібліотек Python для	17.11.20 – 23.11.20	
	машинного навчання		
4	Розробка нейромережевих методів для	24.11.20 – 30.11.20	
	Прогнозування часових рядів		
5	Оформлення пояснювальної записки та	01.12.20 – 11.12.20	
	демонстраційних матеріалів		

Дата видачі завдання 2 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Руденко О.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 66 с., 15 рис., 9 дод., 16 джерел.

НЕЙРОННА МЕРЕЖА, ПРОГНОЗУВАННЯ, ЧАСОВІ РЯДИ, ПЕРЦЕПТРОН, НЕЙРОН, PYTHON, KERAS.

Метою атестаційної роботи є аналіз існуючих методів прогнозування та їх засстосування різними ахітектурами нейронних мереж.

У ході виконання атестаційної роботи виконан аналіз існуючих ахітектур нейронних мереж для виявлення вимог к функціоналу програми. Було програмно реалізовано декілька нейронних мереж для прогнозування часових рядів різними методами.

Представлені проекти нейронних мереж прогнозують поведінку цін акцій компанії Microsoft. В якості мови програмування обраний Python, для якого розроблено безліч бібліотек нейромережевої розробки.

ABSTRACT

Master’s thesis: 66 pages, 15 figures, 9 appendices, 16 sources.

NEURAL NETWORK, FORECASTING, TIME SERIES, PERCEPTRON, NEURON, PYTHON, KERAS.

The major goal of this thesis is the analysis of existing methods of forecasting and their application by different architectures of neural networks.

In order to the attestation work, the analysis of the existing architectures of neural networks was performed to identify the requirements for the functionality of the program. Several neural networks were programmatically implemented to predict time series by different methods.

The presented neural network projects predict the behavior of Microsoft stock prices. Python has been chosen as the programming language, for which many neural network development libraries have been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НМ – нейронна мережа

MLP – багатошаровий перцептрон (англ., Multilayer Perceptron)

LSTM – мережа довгої короткочасної пам’яті (англ., Long Short-Term Memory).

RNN – рекурентна нейронна мережа (англ., Recurrent neural network)

GRU – керований рекурентний нейрон (англ., Gated recurrent unit)

RMSE або RMSD – середньоквадратична помилка (англ., Root mean square deviation)

ВСТУП

Прогнозування часових рядів було і залишається актуальною, особливо останнім часом, коли стали доступні потужні засоби збору та обробки інформації. Прогнозування часових рядів є важливою науково-технічною проблемою, тому що дозволяє передбачити поведінку різних факторів в екологічних, економічних, соціальних та інших системах.

За останні десятиліття розвиток прогнозування як науки привів до створення багатьох моделей і методів, процедур, прогнозування часових рядів, неоднакових за значенням. На думку експертів, результати прогнозування вже визначають більше методів прогнозування, внаслідок того, що в завдання входять методи вибору, що забезпечують відповідні прогнози для досліджуваних процесів або систем. Жорстка статистична підготовка щодо властивостей часових рядів часто обмежує можливості класичних методів прогнозування. Використання нейронних мереж у цій проблемі через наявність у більшості часових рядів складних закономірностей, які не виявили відомих лінійних методів.

Нейромережеві методи обробки інформації почали використовувати кілька десятиліть тому. З часом інтерес до нейромережевих технологій слабшав, а потім відроджувався, така мінливість безпосередньо пов’язана з практичними результатами досліджень. Сьогодні можливості нейромережевих технологій використовуються у багатьох галузях науки – від медицини та астрономії до інформатики та економіки. Здатність нейронної мережі багато в чому обробляти інформацію випливає з її здатності узагальнювати та виділяти приховані взаємозв'язки між вхідними та вихідними даними. Великою перевагою нейронних мереж є те, що вони здатні засвоїти та узагальнити накопичені знання.

Прогнозування часових рядів – одна з найважливіших прикладних задач машинного навчання і штучного інтелекту в цілому, так як

удосконалення методів прогнозу дозволить більш точно передбачити поведінку різних факторів в економіці, техніці, геоекології та інших областях. Основною метою будь-якого прогнозу є побудова, ідентифікація, налаштування і перевірка моделей часових рядів. Традиційно такі моделі ґрунтуються на методах статистичного аналізу і математичного моделювання XX століття в роботах Дж. Боксу, Г. Дженкінса, Ч. Холта, Р. Брауна. Однак в останні десятиліття поряд з традиційними методами широко застосовуються методи, засновані на технологіях машинного навчання, де на перший план виходять нейромережеві моделі. Нейронна мережа добре зарекомендувала себе в тих областях, де необхідне застосування людського інтелекту, зокрема, при вирішенні задач класифікації, розпізнавання образів, аналізу та прогнозування часових рядів.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПРОГНОЗУВАННЯ

1.1 Загальне визначення поняття часовий ряд

Часовий ряд – зібраний в різні моменти часу статистичний матеріал про значення будь-яких параметрів (в найпростішому випадку одного) досліджуваного процесу (рисунок 1.1). Кожну одиницю статистичного матеріалу називають вимірюванням або підрахунком, також допустимо називати його рівнем у вказаний разом з ним час. У часових рядах для кожного еталона повинен бути вказаний час вимірювання або номер вимірювання в порядку. Динамічний ряд істотно відрізняється від простої вибірки даних, оскільки аналіз враховує взаємозв'язок вимірювань з часом, а не лише статистичну різноманітність та статистичні характеристики [1].

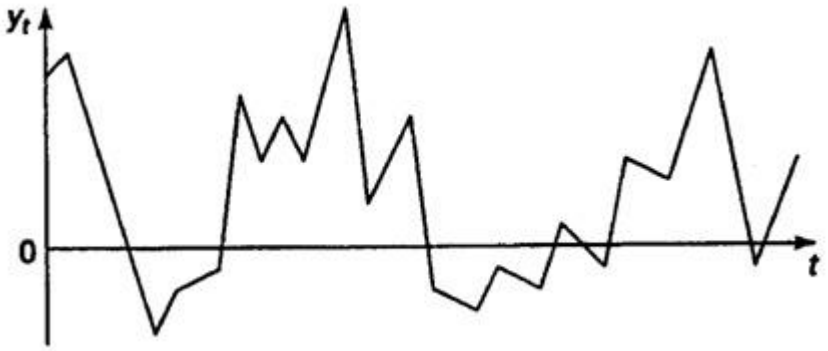

Рисунок 1.1 – Приклад часового ряду

Під часовим рядом розуміють індексовану послідовність точок даних, що відображають розвиток в часі деякого процесу, зафіксованих через рівні проміжки часу.

Часові ряди, як правило, виникають в результаті вимірювання деякого показника. Це можуть бути як показники (характеристики) технічних систем, так і показники природних, соціальних, економічних та інших систем

(наприклад, погодні дані). Типовим прикладом часового ряду можна назвати біржовий курс, при аналізі якого намагаються визначити основний напрям розвитку (тенденцію або тренд).

Часові ряди грають дуже велику роль в технологіях аналізу даних. Аналіз часових рядів дозволяє виявляти тенденції і закономірності в досліджуваних процесах, будувати прогнози і передбачати майбутні зміни в бізнес-оточенні компанії.

Часові ряди бувають одномірні і багатовимірні. Перші містять спостереження за зміною тільки одного параметра досліджуваного процесу або об'єкта, а другі – за двома або більше параметрами.

Вивчення часових рядів відрізняється від інших завдань аналізу даних як за програмними цілями, так і по використовуваних при цьому методів і алгоритмів. Це пов'язано з тим, що дані в часових рядах мають природну впорядкованість. Тому аналіз часових рядів виділяють в самостійну і достатньо велику область.

1.2 Компоненти часового ряду

Закономірна (детермінована) складова часового ряду – послідовність значень, елементи якої можуть бути обчислені відповідно до певної функцією. Закономірна складова часового ряду відображає дію відомих факторів і величин.

Знаючи функцію, яка описує закономірність, відповідно до якої розвивається досліджуваний процес, ми можемо обчислити значення детермінованої складової в будь-який момент часу.

Типові структури, які можна виділити в часовосу ряду – тренд, сезонна компонента, циклічна компонента. Тоді детермінована складова може бути записана у вигляді:

$$d_i = t_i + s_i + c_i, \qquad (1.1)$$

де t_i – тренд;

s_i – сезонна компонента;

c_i – циклічна компонента.

Тренд – повільно змінююча компонента часового ряду, яка описує вплив на часовий ряд довготривало діючих факторів, що викликають плавні і тривалі зміни ряду.

Сезонна компонента – складова часового ряду, що описує регулярні зміни його значень в межах деякого періоду і представляє собою послідовність майже повторюваних циклів.

Сезонна компонента може бути прив'язана до певного календарного часового інтервалу: дня, тижня, місяця – або до якої-небудь події, яка прямо не співвідноситься з конкретними календарними інтервалами. Сезонну компоненту із змінним періодом іноді називають плаваючою.

Часто часові ряди містять зміни, занадто плавні і помітні для випадкової складової. У той же час такі зміни не можна віднести ні до тренду, оскільки вони не є достатньо протяжними, ні до сезонної компоненті, оскільки вони не є регулярними. Подібні зміни називаються циклічної компонентою часового ряду.

Циклічна компонента часового ряду – інтервали підйому або спаду, які мають різну протяжність, а також різну амплітуду розташованих в них значень.

Випадкова (стохастична) складова часового ряду – послідовність значень, яка є результатом впливу на досліджуваний процес випадкових факторів. Випадкова складова і її вплив на часовий ряд можуть бути оцінені тільки за допомогою статистичних методів.

Випадкова складова проявляється як результат впливу набору випадкових факторів на досліджуваний процес і зазвичай виражається в підвищеній мінливості часового ряду, а також у відхиленні значень детермінованою складової. Результуюче значення часового ряду – це результат взаємодії детермінованою і випадкової складових. Найпростіший

вид такої взаємодії – випадок, коли, кожне значення часового ряду можна розглядати як суму (різницю) двох значень, одне з яких обумовлено детермінованою складової, а інше – випадкові.

Побудова та аналіз часових рядів дозволяють виявляти та виміряти закономірності розвитку різних явищ у часі на графіку часового ряду. Ці закономірності не чітко простежуються на кожному рівні, а лише в тренді, у досить тривалій динаміці. Інші, насамперед випадкові, іноді сезонні впливи накладаються на основний закон динаміки. Визначення головної тенденції зміни рівнів, яка називається тенденцією, є одним з головних завдань аналізу часових рядів.

1.3 Методика прогнозування часових рядів

Процес прогнозування є ключовим моментом при прийнятті рішень в управлінні різними актуальними системами. Для побудови такого прогнозу необхідно здійснити детальний аналіз часових рядів відповідних показників з метою виявлення тенденцій, що склалися і побудови моделей досліджуваних процесів. Найпростішими засобами такого аналізу є статистичні методи кластерного і регресійного аналізу.

Прогнозування часових рядів передбачає, що дані, отримані в минулому, допомагають пояснити значення в майбутньому. Важливо розуміти, що в ряді випадків ми маємо справу з деталями, які не відбитими в накопичених даних. Наприклад, з'явиться новий конкурент, який може несприятливо вплинути на майбутні доходи або швидкі зміни в складі робочої сили, які можуть вплинути на показники рівня безробіття. У подібних ситуаціях прогнозування часових рядів не може бути єдиним підходом. Найчастіше різні підходи до прогнозування об'єднують, щоб забезпечити найбільш точні прогнози.

Прогнозування – це передбачення майбутніх подій. Метою будь-якого прогнозування є створення моделі, яка дозволяє заглянути в майбутнє і

оцінити тенденції в змінах того чи іншого фактора. Якість прогнозу в такому випадку залежить від наявності передісторії змінюваного чинника, похибок вимірювання розглянутої величини і інших чинників. формально задача прогнозування формулюється так: знайти функцію f, що дозволяє оцінити значення змінної x в момент часу (t + d) по її N попереднім значенням, так щоб:

$$x(t + d) = f(x(t), x(t -1),..., x(t -N +1)), \quad (1.2)$$

де d береться рівним одиниці, тобто функція f прогнозує наступне значення x.

Для завдань прогнозування аналітику доводиться прийняти рішення щодо таких характеристик часового ряду як тренд, сезонна і циклічна компоненти, робити припущення про модель часового ряду – адитивної, мультиплікативної та ін. Автоматичного способу виявлення трендів у часових рядах не існує. У той же час при вивченні кривою, що відбиває результати спостережень, аналітику важко робити припущення щодо повторюваності форми кривої через рівні проміжки часу. загальним недоліком статистичних моделей є складність вибору типу моделі і підбору її параметрів. Все це істотно збільшує суб'єктивний внесок учасників процесу аналізу і прогнозування часового ряду. Таким чином, результат аналізу і прогнозування часового ряду залежить як від кваліфікації аналітика в предметної галузі, так і від його кваліфікації в методах аналізу.

При вирішенні задачі прогнозування необхідно ідентифікувати змінні, які будуть прогнозуватися, часові параметри і ступінь точності прогнозу. Часто при вирішенні завдань прогнозування виникає необхідність передбачення не самою змінною, а змін її значень. Точність прогнозу, необхідна для вирішення конкретної завдання, має великий вплив на що прогнозує систему. Помилка прогнозу залежить від використовуваної системи прогнозу. Чим більше ресурсів має така система, тим більше шансів

отримати більш точний прогноз. Однак прогнозування не може повністю знищити ризики при прийнятті рішень. Тому завжди враховується можлива помилка прогнозування. Точність прогнозу характеризується помилкою прогнозу.

Узагальнена модель прогнозу. Набір вхідних змінних xi (i = 1, ..., n) – вихідні дані для прогнозу. Набір вихідних змінних yj (j = 1, ..., m) – набір прогнозованих величин, n>m.

Коли вирішується завдання прогнозування часового ряду, що описує динаміку зміни деякого бізнес-процесу, вхідні значення – спостереження за розвитком процесу в минулому, а вихідні – прогнозовані значення процесу в майбутньому. При цьому часові інтервали минулих спостережень і часові інтервали, за якими потрібно отримати прогноз, повинні відповідати один одному.

До прогностичних моделей відносять:

- наївна модель прогнозування;
- екстраполяція;
- прогнозування методом середнього і рухомого середнього;
- регресивні моделі.

1.3.1 Наївна модель прогнозування

Прогнозування наївними моделями передбачає, що дані за останній період прогнозованого часового ряду найкраще описує майбутнє цього прогнозованого ряду. В наївних моделях прогноз зазвичай є дуже простою функцією значної прогнозованої зміни часового ряду в недавнє минуле значення ряду.

Наївна модель припускає, що останній період прогнозованого часового ряду найкраще описує майбутнє цього ряду.

Найпростіша модель:

$$y(t+1)=x(t), \quad (1.3)$$

де x (t) – останнє спостережуване значення;

y (t + 1) – прогноз.

Не слід очікувати великої точності від такої примітивної моделі. Він не тільки не враховує механізми, що визначають прогнозні дані (цей серйозний недолік є спільним для багатьох методів статистичного прогнозування), але і не захищений від випадкових коливань, не враховує сезонних коливань та тенденцій.

Щоб модель враховувала наявність можливих трендів, її можна дещо ускладнити, наприклад перетворивши до виду y(t+1) = x(t) + [x(t) - x(t-1)] або y(t+1) = x(t)[x(t) / x(t-1)]. При необхідності урахування сезонних коливань модель модифікується в такий спосіб:

$$y(t+1)=x(t-s), \quad (1.4)$$

де s – показник, що враховує сезонні зміни прогнозованого часового ряду.

1.3.2 Екстраполяція

Якщо значення функції f(x) відомі в деякому інтервалі $[x_0,x_n]$, то метою екстраполяції є визначення найбільш вірогідного значення в точці x_{n+1}.

Екстраполяція може бути застосована тільки в тих випадках, коли функція f (x), а відповідно і описуваний їй часової ряд, досить стабільна і не схильна до різких змін.

Найбільш популярний метод екстраполяції в даний час – експоненціальне згладжування. Основний його принцип полягає в тому, щоб врахувати в прогнозі всі спостереження, але з експоненціально зменшеними

вагами. Метод дозволяє взяти до уваги сезонні коливання ряду і передбачити поведінку трендової складової.

В разі ряду з «нульовим» трендом, можна вибрати наступну модель експоненціального згладжування:

$$y(t+1) = \lambda\, y(t) + (1-\lambda)\, x(t), \qquad (1.5)$$

де x(t) – останнє спостережуване значення;

y(t) – прогноз на момент часу t;

y(t+1) – прогноз на момент часу t+1;

$0<\lambda<1$ – параметр згладжування або параметр адаптації, що характеризує міру знецінення спостереження за одиницю часу. Інструментом прогнозу є модель, первісна оцінка параметра λ проводиться по декількох перших спостереженнями. На її основі робиться прогноз, який порівнюється з фактичними спостереженнями.

Далі модель коригується відповідно до величини помилки прогнозу і знову використовується для прогнозування наступного рівня, аж до вичерпання всіх спостережень. Таким чином, вона постійно «вбирає» нову інформацію, пристосовується до неї, і до кінця періоду спостереження відображає тенденцію, що склалася на поточний момент. Прогноз виходить як екстраполяція останньої тенденції.

Алгоритм може бути досить успішно використаний як супутний та допоміжний для обробки даних при прогнозуванні проблем. Наприклад, для прогнозування продажів у більшості випадків необхідно розкласти часові ряди (тобто виділити тенденцію, сезонні та нерегулярні компоненти). Одним із методів вибору компонентів тренду є використання експоненціального згладжування.

1.3.3 Прогнозування методом середнього і рухомого середнього

Найбільш проста модель цієї групи – звичайне усереднення набору спостережень прогнозованого ряду:

$$y(t+1)=(x(t)+x(t-1)+x(t-2)+\ldots+x(1))/t. \quad (1.6)$$

При усередненні згладжуються різкі зміни і викиди даних, що робить результати прогнозу більш стійкими до мінливості ряду, але в цілому ця модель прогнозу так само примітивна як «наївна». У формулі прогнозу на основі середнього передбачається, що ряд усереднюється по всьому спостереженнями, але старі значення часового ряду могли формуватися на основі інших закономірностей і втратити актуальність. Щоб підвищити точність прогнозу, можна використовувати «рухоме середнє»:

$$y(t+1)=(x(t)+x(t-1)+x(t-2)+\ldots+x(t-T))/(T+1), \quad (1.7)$$

тобто модель «бачить» минуле на T відліків часу і прогноз будується тільки на цих спостереженнях.

Рухоме середнє або ковзне середнє – один із інструментів аналізу випадкових процесів та часових рядів, який полягає в обчисленні серединної підмножини значень. Ковзне середне може мати ваги, наприклад, щоб збільшити вплив нових даних у порівнянні зі старими. Рухоме середнє – це не скаляр, а випадковий процес. Розмір підмножини, з якого розраховується середнє значення, може бути як постійним, так і змінним [2].

Рухоме середнє один із найдавніших і найпоширеніших інструментів технічного аналізу часових рядів. Метод показує середнє значення цін останнім часом.

Іноді метод рухомого середнього виявляється навіть ефективніше ніж методи, засновані на довгострокових спостереженнях.

1.3.4 Регресивні моделі

Регресія – це умовне математичне сподівання неперервної залежної (вихідної) змінної при спостережуваних значеннях незалежних (вхідних) змінних. Лінійна регресія заснована на гіпотезі, що шукана залежність – лінійна. Кожна незалежна змінна вносить адитивний внесок в результуюче значення з деяким вагою, званому коефіцієнтом регресії.

Регресія називається простою, якщо вхідна змінна одна. Однак така модель є занадто грубим наближенням дійсності, і на практиці, як правило, цікаві залежності від декількох змінних (множинна регресія).

Один з методів прогнозування часових рядів – визначення факторів, які впливають на кожне значення часового ряду. Для цього виділяється кожна компонента часового ряду, обчислюється її внесок в загальну складову, а потім на його основі прогнозуються майбутні значення часового ряду. Даний метод отримав назву декомпозиції часового ряду. Вихідний часовий ряд представляється як композиція тренда, сезонної і циклічної компоненти. Для побудови прогнозу виконується виділення цих компонент з ряду, тобто розкладання ряду по компонентах [3].

Розглянемо прогнозування методом декомпозиції за допомогою тренда. Якщо тренд лінійний, що типово для багатьох реальних часових рядів, то він являє собою пряму лінію, описувану рівнянням:

$$y=a+b*t, \tag{1.8}$$

де y – значення ряду;

a і b – коефіцієнти, що визначають розташування і нахил лінії тренду;

t – час.

Якщо рівняння лінії тренда відомо, то з його допомогою можна розрахувати значення тренда в будь-який момент часу:

$$y_{t+k}=a+b(t'+k), \quad (1.9)$$

де t' – початок прогнозу, k – горизонт прогнозу.

При використанні сезонності для прогнозування методом декомпозиції спочатку з часового ряду забирається тренд і згладжується можлива циклічна компонента. Тоді можна вважати, що залишені дані будуть обумовлені в основному сезонними коливаннями. На основі цих даних обчислюються так звані сезонні індекси, які характеризують зміни часового ряду в часі. Наприклад, часовий ряд містить спостереження за місяцями протягом року. Сезонний індекс, рівний 1, буде встановлений для місяця, очікуване значення в якому становить 1/12 від загальної суми по місяцях. Якщо для деякого місяці встановлюється індекс 1.2, то очікуване значення для цього місяця становить 1/12 + 20%, а якщо 0.8 – то 1 / 12 - 20% і т.д. Ясно, що сума сезонних індексів за рік повинні рівнятися 12.

Використовувати сезонність для прогнозування можна тоді, коли сезонні коливання мають хорошу повторюваність.

За допомогою таблиці значень минулих спостережень можна підібрати (наприклад, методом найменших квадратів) коефіцієнти регресії, налаштувавши тим самим модель.

Працюючи з регресією, слід дотримуватися певної обережності, і необхідно перевіряти відповідність значущих моделей. Існують різні способи перевірити це. Статистичний аналіз залишків є обов’язковим. Корисно, як у випадку з нейронними мережами, мати незалежний набір складів, де можна перевірити якість робочих моделей.

1.4 Актуальність прогнозування часових рядів

Завдання прогнозування часових рядів зустрічаються в багатьох сферах життєдіяльності людини, таких як:

- діагностичні завдання в медицині. Об'єктами даних завдань є

пацієнти. Ознаками – результати обстежень, присутні симптоми захворювань і застосовувалися раніше методи лікування. Приклади ознак: вік, стать, наявність всіляких болів, венозний і артеріальний тиск, частота серцевих скорочень і т.п. Признаковий опис пацієнта є формалізованою історією хвороби. У медичній статистиці, в залежності від цілей і завдань, а також наявних інформаційно-довідкових баз даних, порівняння проводиться з результатами аналогічних або контрольних спостережень, з діючими стандартами і галузевими нормами. Але найчастіше, в практиці організації охорони здоров'я підсумкові оцінки будуються на характеристиці динаміки статистичних показників за кілька років. В оперативних цілях такий аналіз може проводитися і за більш короткі проміжки часу: поміcячно чи поквартально;

- інвестиційні завдання на фінансовому ринку. Для даної задачі вміння добре прогнозувати ціни на фінансові інструменти найбезпосереднішим чином перетворюється в прибуток. Якщо інвестор очікує, що цінний папір виросте, він купує її, бажаючи продати пізніше за більш високою ціною. І, навпаки, чекаючи зниження ціни, інвестор продає цінні папери, щоб пізніше викупити їх назад за нижчою ціною. Завдання інвестора в тому, щоб правильно вгадати напрямок майбутнього зміни ціни – зростання або падіння. Завдання прогнозування фінансових часових рядів була і залишається актуальною, оскільки передбачення є необхідним елементом будь-якої інвестиційної діяльності, адже сама ідея інвестування – вкладення грошей з метою отримання доходу в майбутньому – грунтується на ідеї прогнозування майбутнього. Останнім часом, коли стали доступні потужні засоби збору та обробки інформації, завдання прогнозування фінансових часових рядів також ставати і однією з найпопулярніших завдань для практичного застосування різних Data Mining методів. Широке застосування Data Mining методів в даній області обумовлено наявністю в більшості часових рядів складних закономірностей, які не виявляються лінійними методами;

- завдання прогнозування попиту споживача. Актуальна для супермаркетів, торгових роздрібних мереж і оптовиків. Для якісного управління торговельною мережею важливо з мінімальною похибкою прогнозувати обсяги продажів для певного товару на кілька днів вперед. Спираючись на ці прогнози, магазини планують обсяги закупівель, формують асортимент, визначають оптимальні ціни. Відмінною особливістю даного завдання є кількість товарів, яке може налічувати десятки або навіть сотні тисяч різних найменувань. Прогнозування продажів та прийняття рішень по кожному товару «вручну» зайняло б занадто багато людино-годин і зажадало серйозних витрат. Даними, на основі яких здійснюється прогнозування, є часові ряди цін і обсягів продажів по окремих товарах і магазинам. Сучасні технології дозволяють збирати цю інформацію прямо з касових апаратів. Для збільшення якості прогнозів слід також брати до уваги різні зовнішні чинники, які можуть вплинути на попит, такі як рівень заробітних плат, податків, інфляції, географічні умови, соціально-демографічні умови, рекламні кампанії, акції конкуруючих магазинів і постачальників, політичні санкції та ін.

2 ОГЛЯД АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ

2.1 Біологічна нейронна мережа

Нейрон – будівельний елемент людського мозку. Він аналізує складні сигнали за мікросекунди і відправляє відповіді нервовій системі, яка вирішує складні завдання. У всіх нейронів одна і та ж архітектура. Це означає, що структурні шари у них однакові. Якщо розташувати такі шари послідовно, то можна відтворити копію нашого мозку. Ці послідовні "шари" допомагають нам займатися рутинними справами, приймати складні рішення і розуміти мову один одного.

Нейронні мережі виникли в результаті досліджень у галузі штучного інтелекту, а саме спроб відтворити здатність біологічних нервових систем вчитися та виправляти помилки, моделюючи структуру мозку низького рівня. Основною областю досліджень з штучного інтелекту в 60-ті – 80-ті роки були експертні системи. Такі системи базувалися на високорівневому моделюванні процесу мислення (зокрема, на думці, що процес нашого мислення заснований на маніпулюванні символами). Незабаром стало ясно, що такі системи, хоча і можуть бути корисними в деяких областях, не проникають у деякі ключові аспекти людського інтелекту. Відповідно до однієї точки зору, причиною цього є те, що вони не в змозі відтворити структуру мозку. Для створення штучного інтелекту потрібно побудувати систему з подібною архітектурою.

Мозок складається з дуже великого числа (приблизно 10 млрд.) нейронів, з'єднаних численними зв'язками (в середньому кілька тисяч зв'язків на один нейрон, проте це число може сильно коливатися). Нейрони здатні поширювати електрохімічні сигнали. Нейрон має розгалужену структуру введення інформації (дендрити), ядро і розгалужується вихід (аксон). Аксони клітини з'єднуються з дендритами інших клітин за допомогою синапсів. При

активації нейрон посилає електрохімічний сигнал по своєму аксону. Через синапси цей сигнал досягає інших нейронів, які можуть в свою чергу активуватися. Нейрон активується тоді, коли сумарний рівень сигналів, які прийшли в його ядро здендрітов, перевищить певний рівень (рисунок 2.1).

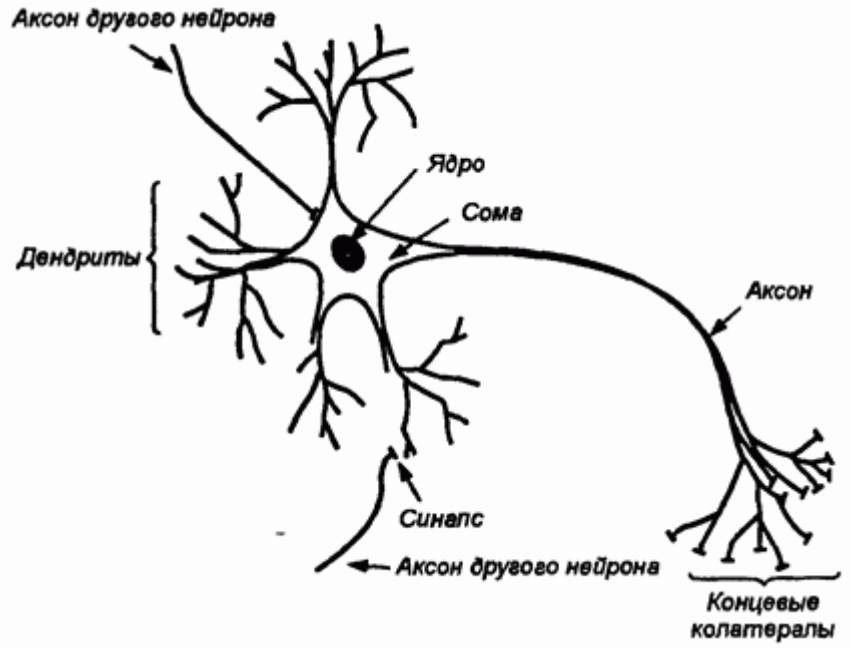

Рисунок 2.1 – Біологічний нейрон

Інтенсивність сигналу, що надходить нейроном (отже, якщо він активований) сильно залежить від активності синапсів. Кожен синапс має тривалість дії, а також спеціальні хімічні речовини, які передають на нього сигнал. Один з найавторитетніших дослідників нейронних систем, Дональд Хебб, висловив постулат, що навчання – це насамперед зміна «сили» синаптичних зв’язків. Наприклад, в класичному досвіді Павлова, кожен раз безпосередньо перед годуванням собаки дзвонив дзвоник, і собака швидко навчилася пов'язувати дзвінок дзвоника з їжею. Синаптичні зв'язки між ділянками кори головного мозку, відповідальними за слух, і слинних залоз посилилися, і при порушенні кори звуком дзвіночка у собаки починалося слиновиділення.

2.2 Визначення поняття штучна нейронна мережа

Штучні нейронні мережі – це новий інструмент аналізу даних, який доцільно застосовувати там, де формалізація обчислювального процесу для досліджуваного явища неможлива або надзвичайно неефективна. Він дозволяє вирішувати завдання, спираючись на неповну, зашумленну або перекручену інформацію, з чим не можуть впоратися традиційні методи.

Перша робота яка передбачува математичну архітектуру штучного нейрона і конструкцію штучної нейронної мережі, була стаття Уорена Маккалоха і Уолтера Питтса, опублікована в 1943 році. Автори зазначили, що через бінарної природи нейронної активності – нейрон або «включений», або «вимкнений», практично без проміжних станів, нейрони зручно описувати в термінах пропозіціональной логіки, а для нейронних мереж розробляють цілий логічний апарат, формалізує ациклічні графи. Сама конструкція штучного нейрона, який у Маккалоха і Питтса називається Threshold Logic Unit (TLU), або Linear Threshold Unit, вийшла дуже сучасною: лінійна комбінація входів, яка потім надходить на вхід нелінійності у вигляді «сходинки», що порівнює результат з деяким порогом (threshold) [4].

В середині двохтисячних років, у розвитку штучного інтелекту відбулася революція: люди навчилися навчати нейронні мережі. Серії завдань, які здавалися неможлива для штучного інтелекту, наприклад, розпізнавання картинок, зараз є тенденцією. На сьогоднішній день існує велика кількість програмних продуктів на базі нейромережевих алгоритмів, які також є невід'ємною частиною серед користувачів. Нейронні мережі поділяють на мережі прямого поширення сигналу, в яких немає циклів і реккурентное нейронні мережі, в яких використовуються цикли для повторюваного навчання.

Нейронні мережі можуть бути синхронні і асинхронні. У синхронних нейронних мережах в кожен момент часу свій стан змінює лише один

нейрон. В асинхронних – стан змінюється відразу у цілої групи нейронів, як правило, у всього шару.

Основу кожної штучної нейронної мережі складають відносно прості, в більшості випадків – однотипні, елементи (осередки), що імітують роботу нейронів мозку (рисунок 2.2).

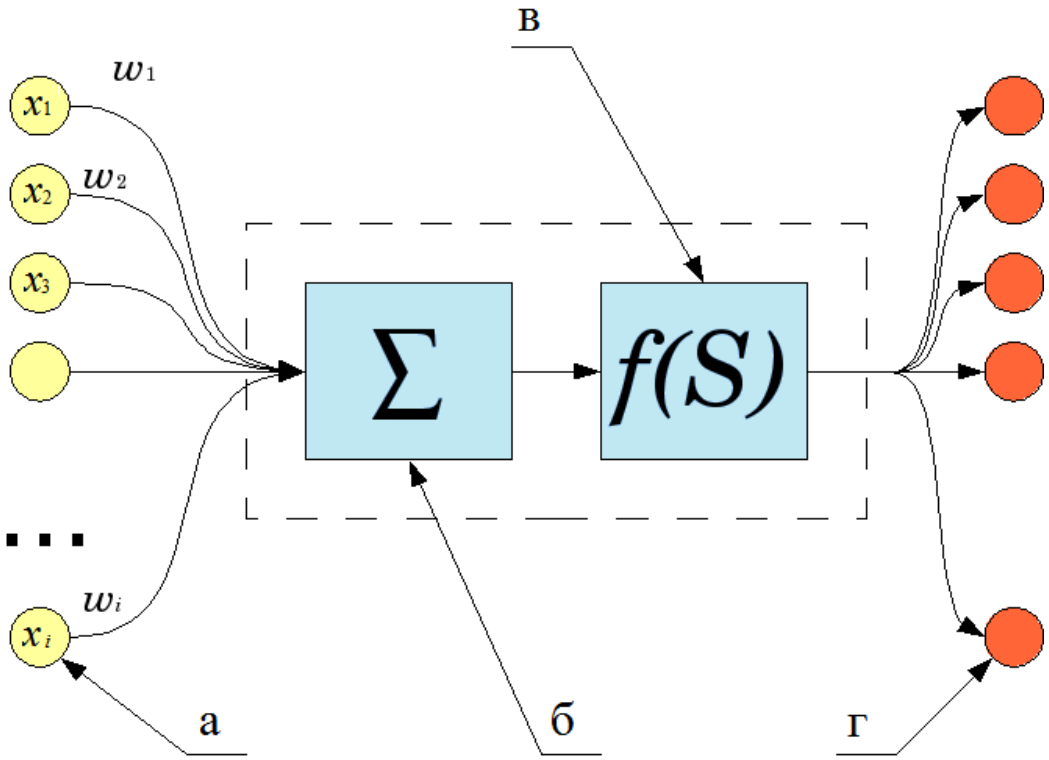

Рисунок 2.2 – Штучний нейрон

Схема штучного нейрона:

а) нейрони, вихідні сигнали яких надходять на вхід;

б) суматор вхідних сигналів;

в) обчислювач передавальної функції;

г) нейрони, на входи яких подається вихідний сигнал данної ваги;

д) w_i. – ваги вхідних сигналів.

Нейрон має групу синапсів – односпрямованих вхідних зв'язків які з'єднані з виходами інших нейронів. Кожен синапс характеризується величиною синоптичної зв'язку або її вагою w_n.

Кожен нейрон має поточний стан, який зазвичай визначається, як зважена сума його входів:

$$s = \sum_{n=1}^{i} x_n w_n \quad . \tag{2.1}$$

Функція f називається функцією активації (рисунок 2.3).

Основні типи функції активації:

а) ступінчаста функція;

б) кусково-лінійна функція;

в) функція гіперболічного тангенсу;

г) сигмоїда.

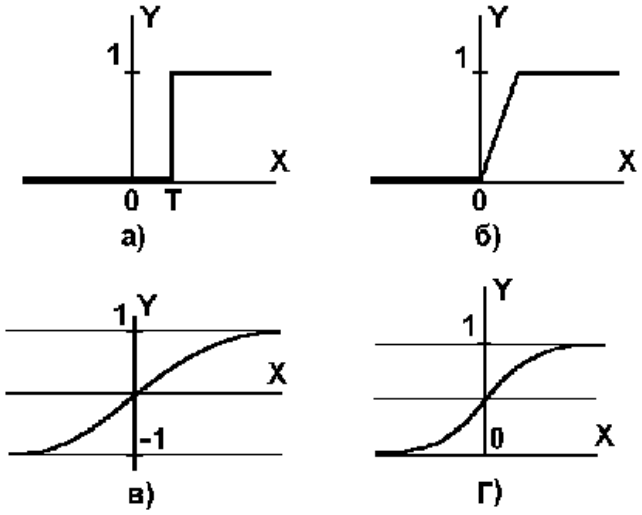

Рисунок 2.3 – Типи функції активації

Ступінчаста функція (англ. Binary step function) є пороговою функцією активації:

$$f(x) = \begin{cases} 1, & x \geq T \\ 0, & x < T \end{cases}. \tag{2.2}$$

Така функція відмінно працює для бінарної класифікації. Але вона не працює, коли для класифікації потрібна більша кількість нейронів і кількість можливих класів більше двох.

Кусково-лінійна функція – функція, визначена на множині дійсних чисел, лінійна на кожному з інтервалів, що становлять область визначення:

$$f(x) = \begin{cases} 1, & x \geq 1 \\ 0, & x \leq 0 \\ x, & 0 < x < 1 \end{cases}. \tag{2.3}$$

Сігмоідна функція (англ. Sigmoid function), яку також називають логістичною (англ. Logistic function), є гладкою монотонно зростаючою нелінійної функцією:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.4}$$

Так як ця функція нелінійна, то її можна використовувати в нейронних мережах з великою кількістю шарів, а також навчати ці мережі методом зворотного поширення помилки.

Незважаючи на безліч сильних сторін сігмоідной функції, у неї є значний недолік. Похідна такої функції вкрай мала у всіх точках, крім порівняно невеликого проміжку. Це сильно ускладнює процес поліпшення ваг за допомогою градієнтного спуску. Більш того, ця проблема посилюється в разі, якщо модель містить багато шарів. Дана проблема називається проблемою зникаючого градієнта.

Перевага сігмоідной функції над іншими – в нормалізації вихідного значення. Іноді, це буває вкрай необхідно. Наприклад, коли підсумкове значення шару повинно представляти ймовірність випадкової величини. Крім

того, цю функцію зручно застосовувати при вирішенні задачі класифікації, завдяки властивості "притискання" до асимптотам.

Функція гіперболічного тангенсу є переважно сигмоїд у випадках, коли немає необхідності в нормалізації. Це відбувається через те, що область визначення даної функції активації центрована щодо нуля, що знімає обмеження при підрахунку градієнта для переміщення в певному напрямку. Крім того, похідна гіперболічного тангенса значно вище поблизу нуля, даючи велику амплітуду градиентному спуску, а отже і більш швидку збіжність.

Безліч всіх нейронів штучної нейронної мережі можна розділити на підмножини – так звані шари. Взаємодія нейронів відбувається пошарово.

Шар нейронної мережі – це сукупність нейронів, які в кожному тактовому циклі паралельно приймають сигнали від інших нейронів даної мережі . Вхідний рівень використовується просто для внесення значних змін. Кожен із прихованих та оригінальних нейронів зв’язаний з усіма елементами попереднього шару. Під час роботи (використання) мереж у вхідних елементах задаються значення вхідних змін, а потім послідовно передаються нейронам проміжного та вихідного шарів. Кожен з них використовує своє значення активації. Потім значення активації перетворюються за допомогою функцій активації, і результатом є вихід нейрона (рисунок 2.4).

Після того, як вся мережа відпрацює, вихідні значення елементів вихідного шару приймаються за вихід всієї мережі в цілому.

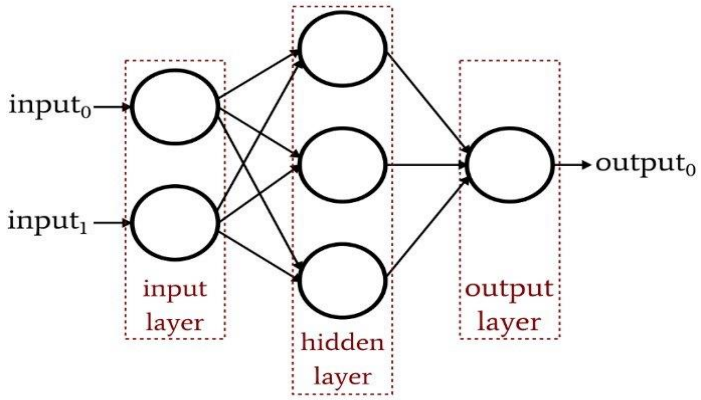

Рисунок 2.4 – Приклад нейронної мережі

2.3 Архітектури нейронних мереж

Нейронна мережа може розглядатися як спрямований граф зі зваженими зв'язками, в якому штучні нейрони є вузлами. За архітектурою зв'язків нейронні мережі можуть бути згруповані в два класи:

- мережі прямого поширення, в яких графи не мають петель;
- рекурентні мережі, або мережі з зворотними зв'язками.

У найбільш поширеному сімействі мереж першого класу, званих багатошаровим перцептроном, нейрони розташовані шарами і мають односпрямовані зв'язку між шарами . Рекурентні мережі є динамічними, так як в силу зворотних зв'язків в них модифікуються входи нейронів, що призводить до зміни стану мережі. Мережі прямого поширення є статичними в тому сенсі, що на заданий вхід вони виробляють одну сукупність вихідних значень, що не залежать від попереднього стану мережі [5].

Рисунок 2.5 – Класифікація нейронних мереж

Вибір архітектури штучної нейронної мережі визначається завданням. Оптимальні конфігурації вже існують для деяких класів завдань. Якщо проблему неможливо згадати в жодному з відомих класів, розробник повинен вирішити проблему синтезу нової конфігурації (рисунок 2.5).

Здатність нейронної мережі передбачати безпосередньо випливає з її здатності узагальнювати та виділяти приховані взаємозв'язки між вхідними та вихідними даними. Навчана мережа здатна прогнозувати майбутнє значення послідовності часових рядів на основі кількох попередніх значень ряду або деяких існуючих факторів. Прогнозування можливе лише тоді, коли попередні зміни певною мірою визначають майбутні. Наприклад, прогнозування котирувань акцій на основі котирувань минулого тижня може бути успішним, а може і не бути успішним, тоді як прогнозування завтрашніх результатів лотерей на основі даних за останні 50 років майже напевно не спрацює.

2.3.1 Мережі прямого поширення сигналу

Нейронні мережі прямого поширення сигналу передбачають поширення сигналу тільки в одному напрямку від входу до виходу. В архітектурі нейронних мереж прямого поширення відсутні петлі, тобто вихідні значення будь-якого шару не впливають на цей шар. Такі мережі застосовуються для асоціації з певним правилом вхідних і вихідних параметрів і тому часто використовуються для розпізнавання образів і завдань класифікації [6].

Мережі прямого поширення сигналу підходять для моделювання співвідношення між набором вхідних параметрів і одним або більше походить параметрів. Це означає, що дані мережі підходять для проблеми знаходження впливу набору входять параметрів на сходить результат. Багатошарові мережі прямого поширення (багатошаровий персептрон) найбільш часто використовувані нейронні мережі на практиці для подібних завдань. При порівнянні ефективності використання багатошарового персептрона і мереж радіальнобазісних функцій для задач класифікації в було показано, що при наявності великої кількості вхідних даних найбільш ефективним є використання багатошарового персептрона. При наявності

малої кількості вхідних параметрів найбільш ефективні мережі радіально-базисних функцій.

Найбільш яскравим представником штучних нейронних мереж з прямою передачею інформації є багатошаровий персептрон. У загальному випадку кожен нейрон даного шару однонаправленно пов'язаний з усіма нейронами наступного шару. Ці зв'язку організовані через синаптичні ваги, які діють як підсилювачі в відповідних каналах. Наприклад, в тришаровому персептроні, нейрони згруповані в послідовні шари: нульовий, перший, другий і третій. Нейрони нульового шару, іноді званого вхідним, не виробляють ніяких обчислень, а слугують лише для передачі вектора вхідних сигналів на нейрони шару, званого першим прихованим шаром. Сигнали з першого шару надходять на другий прихований шар і далі на третій. Останній третій шар є вихідним і в ньому формується вектор вихідних сигналів у нейронну мережу в цілому [7].

Основні ознаки багатошарового персептрона:

- кожен нейрон має нелінійну гладку функцію активації. Найбільш популярною функцією є сигмоїдальна функція;

- мережа містить один або кілька шарів прихованих нейронів, який дозволяють мережі отримувати найбільш важливі ознаки об'єкта з вхідного вектора;

- мережа має високий ступінь зв'язності, що реалізовується за допомогою синаптичних з'єднань.

Комбінація цих ознак забезпечує обчислювальну потужність цього перцептрона, але також є причиною неповноти сучасних знань про поведінку таких мереж.

Пік популярності MLP в машинному навчанні припав на 1980-і роки в таких областях, як розпізнавання мови і зображень, системах машинного перекладу. Однак пізніше вони зіткнулися з конкуренцією з іншими технологіями машинного навчання, такими, як машини опорних векторів. Інтерес до багатошаровим персептронам повернувся завдяки успіхам

глибокого навчання для прогнозування часових рядів та інших актуальних обсичлень та классифікації даних.

MLP показали можливість знаходити наближені рішення для надзвичайно складних завдань. Зокрема, вони є універсальним аппроксиматором функцій, тому з успіхом використовуються в побудові регресійних моделей. Оскільки класифікацію можна розглядати як окремий випадок регресії, коли вихідна змінна категоріальна, на основі MLP можна будувати класифікатори.

Вперше багатошаровий персептрон був запропонований Ф. Розенлаттом. Однак в тому вигляді, в якому він використовується в даний час, багатошаровий персептрон був розроблений Д. Румельхартом.

Персептрон Румельхарта відрізняється від персептрона Розенблатта за такими властивостями:

- використання нелінійної активаційної функції;
- число прихованих шарів більш одного (зазвичай не більше трьох);
- вхідні сигнали бінарні, а кодуються десятковими числами, нормованими до інтервалу [0,1];
- вихідна помилка мережі визначається не як число помилково розпізнаних прикладів, а як деяке значення нев'язки;
- навчання проводиться не до мінімізації помилки, а до стабілізації ваг мережі, що дозволяє уникнути перенавчання.

Поява алгоритму зворотного поширення в 1986 році призвело до широкого використання багатошарового персептрона як для задач розпізнавання, так і для задач класифікації.

В даний час багатошарові персептрони як і раніше є популярними інструментом аналізу і прогнозування даних і входять в більшість платформ бізнес-аналітики.

Навчання методом зворотного поширення помилки такий нейронної мережі передбачає два проходження по всіх нейронах мережі – прямий прохід (від вхідного шару до вихідного для отримання результуючого

вихідного сигналу) і зворотний прохід (від вихідного шару до вхідного для коригування ваг зв'язків нейронної мережі). Ваги зв'язку між нейронами налаштовуються з метою максимального наближення вихідного сигналу мережі до бажаного в статистичному сенсі.

Для даного виду навчання можна виділити два типи сигналів, що передаються по нейронної мережі:

- функціональний сигнал, який передається в прямому напрямку. При передачі сигналу в кожному нейроні обчислюється функція активації на підставі прийшли до нього сигналів від інших нейронів. В результаті будуть отримані значення сигналів для всіх нейронів вихідного шару;
- сигнал помилки, розповсюджуваний в зворотному напрямку. Даний сигнал розраховується кожним нейроном за допомогою функції помилки (зазвичай обчислення градієнта поверхні помилки) на підставі значення помилки, що надходить із сусіднього шару нейронів.

Існує безліч спірних питань при проектуванні мереж прямого поширення – наприклад, скільки шарів необхідні для даного завдання, скільки слід вибрати елементів у кожному шарі, як мережа буде реагувати на дані, не включені в навчальну вибірку (яка здатність мережі до узагальнення), і який розмір навчальної вибірки необхідний для досягнення "хорошою" здатності мережі до узагальнення.

Незважаючи на те, що навчання штучної нейронної мережі є ресурсоємним процесом, обчислення, що виконуються для одного нейрона мережі, описуються досить просто і не вимагають великих ресурсів. Разом з тим, в нейронних мережах прямого поширення, нейрони одного шару не мають міжнейронних зв'язків між собою. Ці два фактори роблять можливим розробку ефективного алгоритму для розпаралелювання обчислень окремого шару нейронної мережі.

Багато параметрів ще повинні бути визначені шляхом проб і помилок для широкого застосування багатошарових мереж прямого поширення для класифікації та апроксимації функцій часових рядів. Існуючі теоретичні

результати дають лише слабкі орієнтири для вибору цих параметрів в практичних додатках.

2.3.2 Рекурентні нейронні мережі

Мережі зі зворотними зв'язками RNN (англ. Recurrent neural network) – штучні нейронні мережі, в яких вихід нейрона може знову подаватися на його вхід. У більш загальному випадку це означає можливість поширення сигналу від виходів до входів [8].

У мережах із зворотними зв'язками виходи нейронів можуть повертатися на входи. У мережах прямого поширення вихід мережі визначається вхідним сигналом і ваговими коефіцієнтами при штучних нейронах. Це означає, що вихід якогось нейрона визначається не тільки його вагами і вхідним сигналом, але ще і попередніми виходами (так як вони знову повернулися на входи).

Перші глибокі мережі з'явилися ще в середині 1960-х років: вони були описані в роботах радянського вченого А.Г. Івахненко. Він розробив метод групового обліку аргументів, суть якого полягає в наступному:

- спочатку вибирається загальний вигляд, параметричне сімейство моделей, які будуть навчатися. А.Г. Івахненко пропонував використовувати поліном Колмагорова - Габора, тобто звичайні многочлени з невідомими коефіцієнтами;
- будуються і навчаються різні варіанти моделей;
- за допомогою метрики якості вибирається кілька кращих моделей, в разі, якщо кращу якість уже досягнуто, то цикл закінчується;
- разі, якщо потрібне якість не досягнуто, необхідно будувати моделі наступного рівня, використовуючи виходи підібраних параметрів моделей на попередньому кроці, як входи для наступних;
- даний процес слід повторювати рекурсивно до тих пір, поки якість моделі або не досягне необхідного рівня, або не перестане поліпшуватися.

Головними особливостями рекурентних нейронних мереж є здатність використовувати вихідну інформацію для поточного завдання, а також подавати на вхід значення рекурсивно, за рахунок чого параметри оптимізуються і покращують якість результату. Реккурентні нейронні мережі будують залежності на короткострокових даних, актуалізуючи їх на кожному циклі навчання (рисунок 2.6). Наприклад, для прогнозу ціни акції сьогодні нам не потрібен весь масив даних за десятиліття ведення угод.

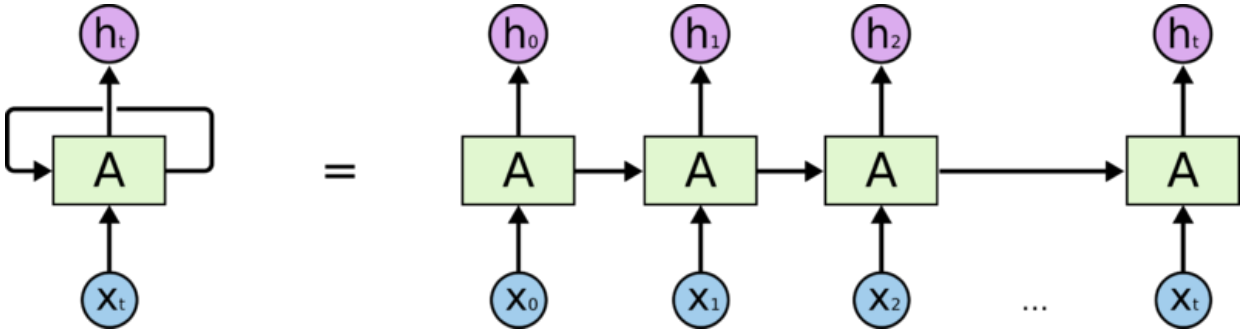

Рисунок 2.6 – Рекурентна нейронна мережа в розгортці

Навчання RNN схоже на тренування звичайної нейронної мережі. Ми також використовуємо алгоритм зворотного поширення помилки (backpropagation), але з невеликою зміною. Через те, що однакові параметри використовуються на всіх етапах мережі, градієнт у кожній освіті не залежить лише від розрахунків поточного потоку, а й від попередніх кварталів часу(рисунок 2.7).

Бувають ситуації, коли необхідно більше інформації для аналізу патернів, а реккурентное нейронні мережі в міру зростання відстані інформаційного потоку втрачають здатність аналізувати залежності в часових рядах для подальшого прогнозування на основі проаналізованих даних. Рішення даної проблеми було запропоновано Юргеном Шмідхубером і Зепп Хохрайтером [9].

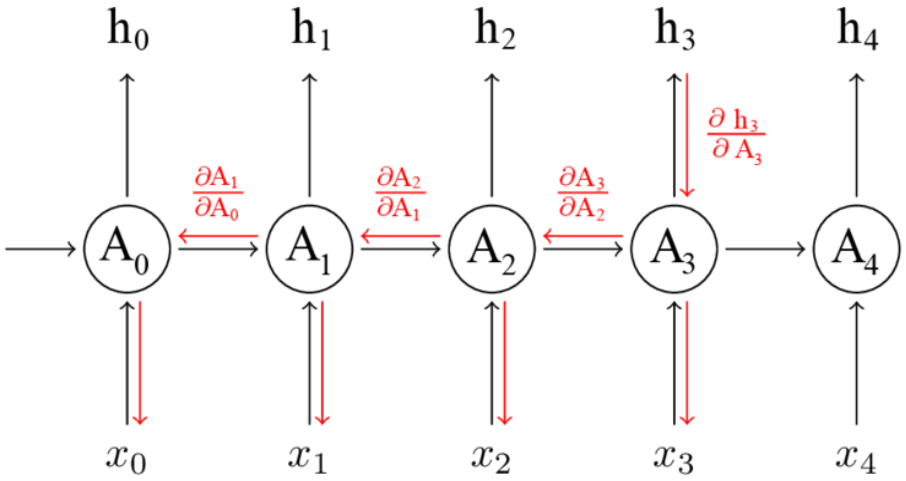

Рисунок 2.7 – Алгоритм зворотного поширення помилки крізь час

Була запропонована особлива конфігурація реккурентної нейронної мережі з довгостроково – короткостроковою пам'яттю LSTM (англ. «Long Short-Term Memory»). Ідея полягає в моделювання «пам'яті» шляхом додавання спеціальної клітинки gate, що дозволяє усувати проблему зростання інформаційного потоку (рисунок 2.8).

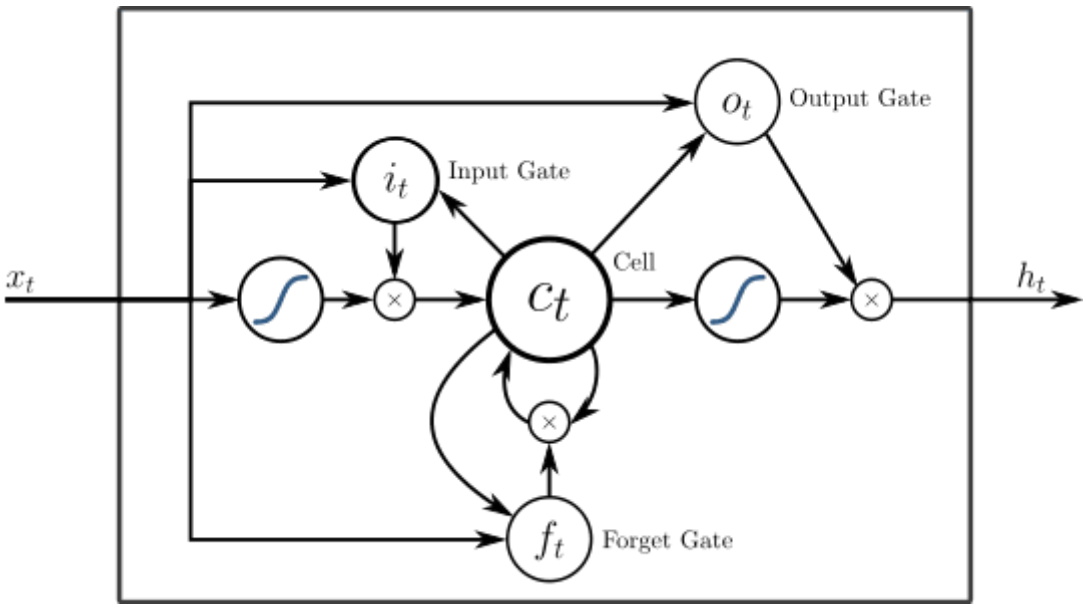

Рисунок 2.8 – Блок LSTM

Мережа довгої короткострокової пам'яті LSTM – різновид архітектури рекуррентной нейромережі, створена для більш точного моделювання часових послідовностей і їх довгострокових залежностей, ніж традиційна рекуррентная мережа. Часто LSTM використовується в блоках з декількох елементів. Ці блоки складаються з 3 або 4 гейтів або воріт (наприклад, вхідних, вихідних і забуваючих воріт), які керують побудовою інформаційного потоку з логістичної функції. Мережа LSTM не використовує функцію активації в періодичних компонентах, збережені значення не змінюються, а градієнт не має тенденції зникати під час навчання.

Ключові компоненти LSTM-модуля: стан осередку і різні фільтри. Про стан осередку можна говорити, як про пам'ять мережі, яка передає відповідну інформацію по всьому ланцюжку модулів. Таким чином, навіть інформацію з ранніх часових кроків часовго ряду можна отримати пізніше, нівелюючи ефект короткочасної пам’яті.

По мірі того, як відбувається навчання, стан осередку змінюється, інформація про значення часових рядів додається або видаляється зі стану осередку структурами, званими фільтрами. Вони складаються з шару сигмоїдної нейронної мережі і операції поточечного множення. Фільтри контролюють потік інформації на входах і на виходах модуля на підставі деяких умов.

Сигмоїдальний шар повертає числа в діапазоні [0; 1], які позначають, яку частку кожного блоку інформації слід пропустити далі по мережі. Множення на це значення використовується для пропуску або заборони потоку інформації значень часових рядів всередину і назовні пам'яті. Вихідний фільтр контролює міру того, в якій мірі значення часовго ряду, що знаходиться в пам'яті, використовується при розрахунку вихідний функції активації. Наприклад, вхідний фільтр контролює міру входження нового значення в пам'ять, а фільтр забування контролює міру збереження значення в пам'яті [10].

Основні компоненти LSTM:

а) стан осередку;

б) фільтри, які контролюють стан осередку;

1) забування;

2) вхідний;

3) вихідний.

LSTM має наступний принцип роботи.

Спершу "шар фільтра забування" (англ. Forget gate layer) визначає, яку інформацію можна забути або залишити. Значення попереднього виходу h_{t-1} і поточного входу x_t пропускаються через сигмоїдальна шар. Отримані значення знаходяться в діапозоні [0; 1]. Значення, які ближче до 0 будуть забуті, а до 1 залишені:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right). \tag{2.5}$$

Далі вирішується, яка нова інформація буде зберігатися в стані осередки. Цей етап складається з двох частин. Спочатку сигмоїдальна шар під назвою "шар вхідного фільтра" (англ. Input layer gate) визначає, які значення часового ряду слід оновити. Потім tanh-шар будує вектор нових значень-кандидатів $\tilde{C}_t$, які можна додати в стан осередку:

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C). \end{aligned} \tag{2.6}$$

Для заміни старого стану осередку C_{t-1} на новий стан C_t. Необхідно помножити старе стан на ft, забуваючи те, що вирішили забути раніше. Потім додаємо $i_t * \tilde{C}_t$. Це нові значення-кандидати, помножені на t – на скільки оновити кожне з значень стану:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \,. \tag{2.7}$$

Останній крок – визначає, яка інформація буде видаватися. Вихідні дані базуватимуться на нашому стані комірки, і до них застосовуватимуться деякі фільтри. По-перше, значення попереднього виходу h_{t-1} і поточного входу x_t пропускаються через сигмоїдальний шар, який вирішує, яка інформація зі стану осередку буде виведена. Потім значення стану комірки проходять через шар tanh, щоб отримати вихідні значення з діапазону від -1 до 1, і помножені на вихідні значення сигмоїдного шару, що дозволяє відображати лише необхідну інформацію значень прогнозованого часового ряду:

$$\begin{aligned} o_t &= \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \,. \tag{2.8}$$

Описана вище схема є традиційною для LSTM. Але не всі ідентичні LSTM. Насправді майже в кожній статті використовуються різні версії. Відмінності незначні, але варто згадати деякі з них.

Модель LSTM контрольованих рекурентних нейронів (англ. Gated recurrent units, GRU) відрізняється від стандартних ренурентних мереж. Їх менше на одному фільтрі, і вони підключені дещо інакше. Фільтри для "забуття" та введення об'єкта в одному фільтрі "оновлення" (англ. Update gate). Цей фільтр визначає, скільки інформації зберігається з моменту останнього стану, а також скільки інформації отримує попередня версія. Крім того, стан комісії поєднується із прихованим станом, є й інші незначні зміни. Ворота скидання починає працювати так само, як фільтр забуття, але дещо в іншому місці. Наступний рівень надсилається з повною інформацією про стан, без вихідного фільтра. У більшості випадків GRU працюють так само, як LSTM, найголовніша відмінність полягає в тому, що GRU працюють

трохи швидше і швидше, але мають дещо менш виразні можливості. Результати моделей приблизно кращі, ніж LSTM, і їх популярність незрозумілим чином зростає. Ефективність у вирішенні задач моделювання музичних та мовних сигналів у порівнянні з використанням довгої короткочасної пам'яті [11].

Є багато інших модифікацій, таких як глибоко керовані рекуррентні нейронні мережі (англ. Depth Gated RNNs). Глибоко керовані рекуррентні нейронні мережі вводять фільтр глибини для з'єднання елементів пам'яті сусідніх шарів. Це вводить лінійну залежність між нижньою та верхньою рекурентними одиницями. Важливо зазначити, що лінійна залежність проходить через функцію затвора, яка називається фільтром забуття. Ця архітектура здатна вдосконалити машинний переклад та моделювання мови.

2.4 Навчання нейронної мережі

Можливість навчатися є сновною перевагою нейронних мереж перед звичайними алгоритмами. Нейронні мережі навчаються а не програмуються як це звичайно розуміють. Технічно тренування допомагає знайти коефіцієнти зв’язку між нейронами. У процесі навчання нейронна мережа здатна виявляти складні взаємозв'язки між входом і освітою, а також виконувати узагальнення. Це означає, що при успішному використанні мереж ми можемо повернути правильний результат у базі даних, якої не було в навчальній програмі, а також невідомі або "галасливі", приватно відкриті дані.

Для проектування навчального процесу, перш за все, необхідно мати модель зовнішнього середовища, в якому функціонує нейронна мережа – знати інформацію часового ряду, доступну мережі. Ця модель визначає парадигму навчання. Як змінити вагові параметри мережі, які правила навчання регулюють процес будівництва, все це потрібно розуміти, щоб навчити мережу для подальшого прогнозування часових рядів. Алгоритм

тренування показує процедуру, яка використовує правила тренувань для регулювання ваги.

Основною силою мозку є здатність вчитися. У контексті нейромережевих процесів навчання можна розглядати як використання архітектурних мереж та вагових зв’язків для ефективного виконання конкретного завдання. Властивість мережі, навченої на складах, робить їх більш привабливими порівняно з системами, що підтримують певні системи правил роботи, сформульовані експертами. Продуктивність мережі покращується в міру повторного контролю ваги. Як правило, нейронна мережа забезпечує регулювання вагового зв'язку відповідно до загальної навчальної вибірки.

Існують три парадигми навчання: "з учителем", "без вчителя" (самонавчання) і змішана. У випадку "з учителем" нейронна мережа має правильними відповідями (виходами мережі) на кожен вхідний приклад. Ваги налаштовуються так, щоб мережа виробляла відповіді як можна більш близькі до відомих правильних відповідей. Посилений варіант навчання з учителем передбачає, що відома тільки критична оцінка правильності виходу нейронної мережі, але не самі правильні значення виходу. При змішаному навчанні частина ваг визначається за допомогою навчання з учителем, в той час як інша виходить за допомогою самонавчання. Навчання без вчителя не вимагає знання правильних відповідей на кожен приклад навчальної вибірки. В цьому випадку розкривається внутрішня структура даних або кореляції між зразками в системі даних, що дозволяє розподілити зразки по категоріях [12].

Складність зразків визначає число навчальних прикладів, необхідних для досягнення здатності мережі до узагальнення. Під ємністю розуміється, скільки зразків може запам'ятати мережа, і які функції і межі прийняття рішень можуть бути на ній сформовані. Занадто мала кількість прикладів може викликати "перенавчання" мережі, коли вона добре дає собі раду на прикладах навчальної вибірки, але погано – на тестових прикладах, підлеглих того ж статистичному розподілу. Відомі 4 основних типи правил навчання:

машина Больцмана, правило Хебба, навчання методом змагання і корекція помилкою [13].

Архітектура мережі радіально-базисних функцій задіє парадигму змішаного навчання, а архітектура перцептрону – парадигму навчання з учителем. Три фундаментальні властивості теорії навчання за прикладами: зразкова складність, ємність і обчислювальна складність.

Алгоритм зворотного поширення (англ. back propagation) – найвідоміший варіант алгоритму навчання нейронної мережі. Існують сучасні алгоритми для іншого порядку, таким чином, як метод сполучених градієнтів, які за кожним завданням працюють суттєво швидше (іноді за порядком). Алгоритм зворотного поширення деяких людей має певні переваги, а також більший досвід для розуміння.

Епохи – це кроки ітеративного алгоритму навчання мережі. Всі навчальні спостереження, початкові значення мережі подаються в кожну епоху на вхід мережі для порівняння з цільовими значеннями та розрахунку помилок. Значення помилки та градієнта помилки використовується для регулювання ваг, всі дії повторюються відповідно до кількості епох. Початкова конфігурація мережі вибирається випадковим чином, і коли процес навчання зупиняється або тоді, коли помилка досягає певного рівня незначності, або коли минула певна кількість епох, або коли помилка перестає зменшуватися [14].

2.5 Нейромережеве прогнозування

Нейронні мережі неймовірно потужні, але їх зазвичай використовують для класифікації. Сигнали проходять через шари нейронів і узагальнюються в один з декількох класів. Однак їх можна дуже швидко адаптувати в регресійні моделі, якщо змінити останню функцію активації.

Замінивши останню функцію активації (вихідний нейрон) лінійною функцією активації, вихідний сигнал можна відобразити на безліч значень,

що виходять за межі фіксованих класів. Таким чином, на виході буде не ймовірність віднесення вхідного сигналу до якого-небудь одного класу, а безперервне значення, на якому фіксує свої спостереження нейронна мережа. У цьому сенсі можна сказати, що нейронна мережа як би доповнює лінійну регресію.

Нейромережева регресія має перевагу нелінійності, яку можна ввести з сігмоідной і іншими нелінійними функціями активації раніше в нейронної мережі. Однак надмірне використання ReLU як функції активації може означати, що модель має тенденцію уникати виведення від'ємних значень, оскільки ReLU ігнорує відносні відмінності між негативними значеннями.

Це можна вирішити або нормалізацією даних до строго позитивного діапазону перед навчанням, або обмеженням використання ReLU і додаванням більшої кількості негативних значень функцій активації.

Препроцесорна обробка, як правило, зводиться до масштабування значень відліків з метою їх приведення до єдиного діапазон, так як часовий ряд представляє собою послідовність числових відліків.

Для рішення задачі прогнозування часових рядів вибирають довільний часовий ряд, що містить N відліків, і розбивають його на дві вибірки: навчальну, тестуючу або контрольну вибірки, які потім подаються на вхід мережі. Значення часового ряду в необхідний момент часу є результатом прогнозування.

Кожна вибірка представляє собою дискретну функцію, задану в точках на інтервалі [0, N] с кроком 1, де N – максимальне значення аргументу цієї функції. Роль нейронної мережі при вирішенні завдань прогнозування полягає в прогнозі майбутньої реакції системи по її попередньому поведінки. Володіючи інформацією про значеннях змінної x в моменти, попередні прогнозування x (k-1), x (k-2), ..., x (k-N), мережа виробляє рішення, яким буде найбільш ймовірне значення послідовності x (k) в поточний момент k. Для адаптації вагових коефіцієнтів мережі використовуються фактична похибка прогнозування $\varepsilon = x(k) - x(k)$ в [0, k] моменти часу [15].

3 РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖЕВОГО ПРОГНОЗУВАННЯ

3.1 Огляд бібліотек машинного навчання

Keras – відкрита нейромережева бібліотека яка написана на мові Python. Це доповнення до фреймворків Deeplearning4j, TensorFlow та Theano. Спрямована на оперативну роботу з мережами глибокого навчання, розробленими для компактності, модульності та розширення. Він був створений в рамках дослідницьких зусиль проекту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором і підтримкою є Франсуа Шолль (фр. François Chollet), процюючий інженером Google.

Keras підтримує основні види шарів і структурні елементи. Підтримує як рекурентні, так і згорткові нейромережі, має в своєму складі реалізацію відомих архітектур нейромереж (наприклад, VGG16). Деякий час назад шари з даної бібліотеки стали доступні всередині бібліотеки Tensorflow. Існують готові функції для роботи з зображеннями і текстом. Інтегрована в Apache Spark за допомогою дистрибутива dist-keras. Дана бібліотека дозволяє на більш високому рівні працювати з нейронними мережами. Як бібліотеки для бекенд може використовуватися як Tensorflow, так і Theano.

Scikit-learn (також відома як sklearn або scikits.learn) – це безкоштовна бібліотека програмного забезпечення для машинного навчання для мови програмування Python, яка забезпечує функціональність для створення та навчання різноманітних алгоритмів класифікації, регресії та кластеризації, таких як лінійна регресія, випадковий ліс, посилення градієнта, і працює спільно з бібліотеками NumPy та SciPy.

NumPy – модуль з відкритим кодом для python, який забезпечує загальні математичні та числові операції у вигляді попередньо скомпільованих швидких функцій. Вони можуть відвідувати високоякісні

пакети. Вони можуть використовувати функціональність, порівнянну з функціональністю MatLab. NumPy (Numeric Python) забезпечує основні методи маніпулювання великими масивами та матрицями. SciPy (Scientific Python) розширює функціональну кількість numpy величезною колекцією корисних алгоритмів, таких як мінімізація, перетворення Фур'є, регресія та інші компоненти математичних методів.

Matplotlib – бібліотека Python для побудови якісних двовимірних графіків. Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy і IPython надає можливості, подібні MATLAB.

3.2 Задача прогнозування

Можна сформулювати прознозування часових рядів як задачу регресії.

Можна написати просту функцію для перетворення нашого окремого стовпця даних у n-колонковий набір даних: перші стовпці, що містить ціну акцій дня та попередніх n-1 днів [t-n-1, t], а останній стовпець, що містить ціну акцій наступного дня (t+1), що прогнозується.

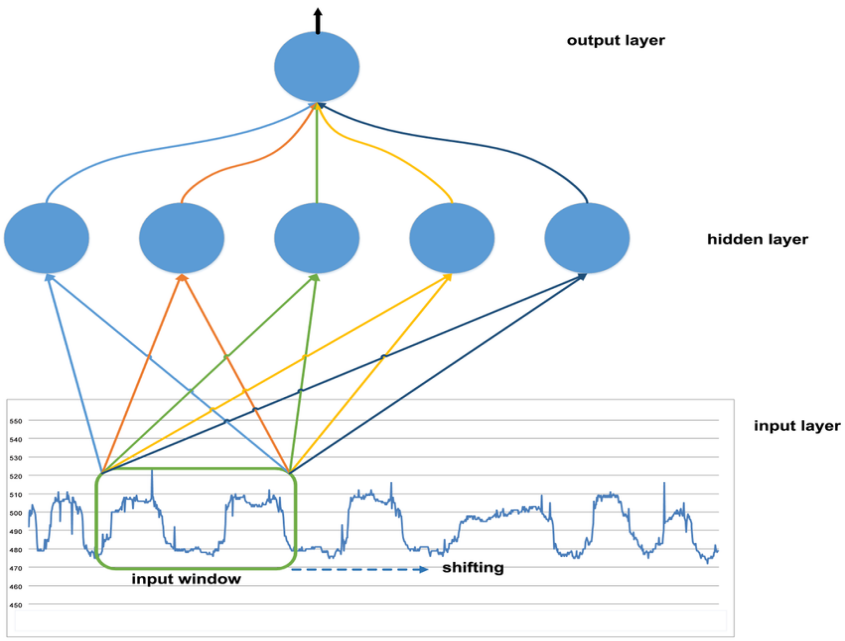

Рисунок 3.1 – Вікно введення послідовності

Будемо використовуваття для прогнозування часових рядів данні про зміну ціни акцій компанії Microsoft за 2018 з сайту біржі Nasdaq [16].

В якості архітектур нейронних мереж будемо використовувати кофігурацію багатошарового перцептрона та рекурентну мережу довгої короткочасної пам’яті.

Тобто, враховуючи ціну акцій Microsoft (у доларах США) цього дня, якою буде ціна акцій наступного дня. Для цього виділяється кожна компонента часового ряду, обчислюється її внесок в загальну складову, а потім на його основі прогнозуються майбутні значення часового ряду.

Вікно введення послідовності сигналів (рисунок 3.1) використовується для подачі до кожного прихованого нейрону. Вікно введення зміщується на вхідній послідовності, щоб проаналізувати всю послідовність.

Спрогнозуємо часові ряди за допомогою мережі прямого поширення MLP і рекурентної мережі LSTM, які будуть приймати на вхід один та декілька компонент часового ряду. А також, порівняємо якість моделей прогнозування за допомогою середньоквадратичної помилки RMSE.

Формула для вимірювання середньоквадратичної помилки RMSE або RMSD (англ. Root mean square deviation) між двома часовими рядами $x_{1,t}$ і $x_{2,t}$,:

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{T} (x_{1,t} - x_{2,t})^2}{T}}. \qquad (3.1)$$

3.3 Завантаження даних

Перш ніж розпочати, спочатку імпортуємо всі потрібні функції та класи (приклад 3.1). Це передбачає робоче середовище SciPy із встановленою бібліотекою глибокого навчання Keras.

```
import numpy
import matplotlib
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

Приклад 3.1 – Імпорт функцій та бібліотек

Для завантаження набору даних використано функцію read_csv з бібліотеки Pandas(у .csv з Nasdaq дані завантажені в зворотному порядку - від 2018 до 2008, так що спочатку треба їх "перевернути" за допомогою [::-1]). Враховуючи, що кожне спостереження розділяється однаковим інтервалом в один день, можна виключити перший стовпець. Будемо використовувати останні 200 спостережень.

Потім можна витягти масив NumPy з фрейму даних і перетворити цілі значення у значення з плаваючою комою, які більше підходять для моделювання за допомогою нейронної мережі.

Нейронні мережі чутливі до масштабу вхідних даних, зокрема, коли використовуються функції активації сигмоїда (за замовчуванням) або tanh. Тому треба масштабувати дані до діапазону від 0 до 1, що називається нормалізацією. Можна легко нормалізувати набір даних, використовуючи клас попередньої обробки MinMaxScaler з бібліотеки scikit-learn (приклад 3.2).

Після завантаження можна легко побудувати графік даних. Код для завантаження та побудови графіку даних наведено нижче (рисунок 3.2).

```
dataframe = read_csv('HistoricalQuotes.csv', usecols=[1], engine='python', skipfooter=3)[:200][::-1]
dataset = dataframe.values
dataset = dataset.astype('float32')
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

Приклад 3.2 – Завантаження даних

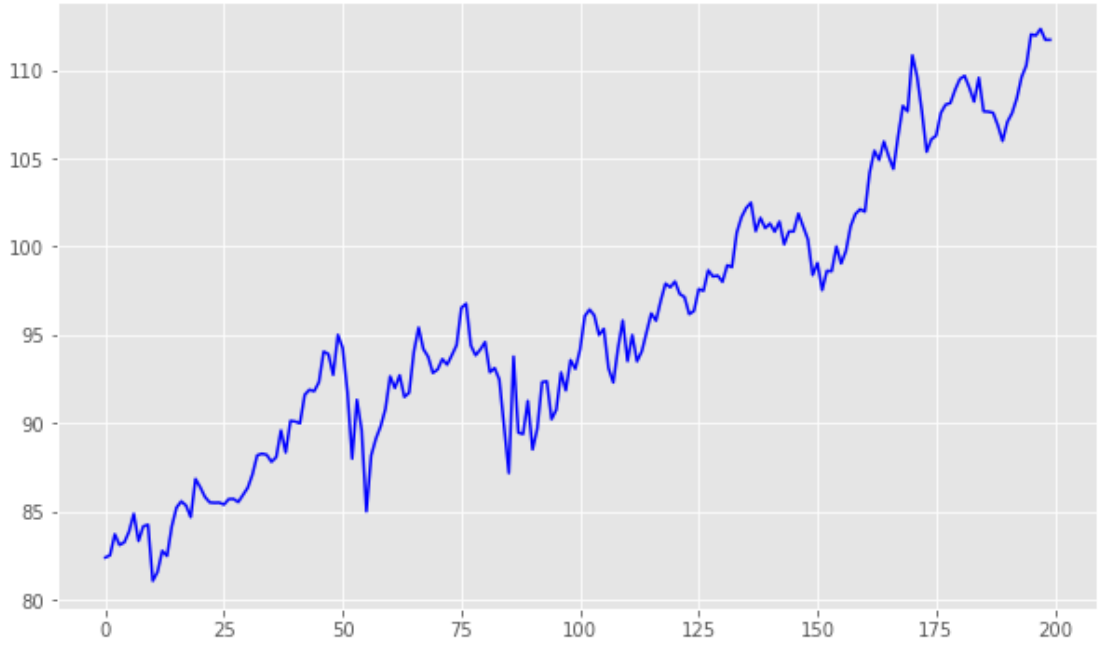

Рисунок 3.2 – Графік часового ряду

3.4 Розподіл даних

Після того, як будуть змодельовані дані та оцінене вміння прогнозувати значення моделі на навчальному наборі даних, потрібно скласти уявлення про прогнозування моделі на нових тестових даних. Для нормальної проблеми класифікації або регресії робиться використовуючи перехресну перевірку.

Для даних часових рядів важлива послідовність значень. Простий метод, який можна використати – це розподілити впорядкований набір даних на навчальні та тестові набори даних. Створимо код що обчислює індекс точки розбиття та розділяє дані на набори навчальних даних із 67% спостережень для навчання нашої моделі і 33% для тестування моделі.

Тепер треба визначити функцію для створення нового набору даних (приклад 3.3).

Функція приймає два аргументи: dataset, який є масивом NumPy, для перетворення на набір даних, і look_back , тобто кількість попередніх кроків

часу, які використовуються як вхідні змінні для прогнозування наступного періоду часу.

За замовчуванням буде створено набір даних, де X – ціна акцій на даний момент (t), а Y – ціна акцій на наступний час (t+1).

```
def create_dataset(dataset, look_back=1):
    data_X, data_Y = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        data_X.append(a)
        data_Y.append(dataset[i + look_back, 0])
    return numpy.array(data_X), numpy.array(data_Y)

train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))
```

Приклад 3.3 – Функція для створення нового набору даних

Один або декілька останніх кроків часу (у разі вікна) можуть бути використані для прогнозування наступного часового кроку. Для 1 кроку вхідною змінною буде t, а вихідною змінною t + 1, а наприклад, для 5 кроків (або вікна) вхідними змінними будуть t-4, t-3, t-2, t-1, t, а вихідною змінною t+1.

Використаємо створену функцію create_dataset() (приклад 3.4) для підготовки навчальної та тестової наборів даних для моделювання (приклад 3.5).

```
look_back = 5
train_X, train_Y = create_dataset(train, look_back)
test_X, test_Y = create_dataset(test, look_back)
for i in range(len(test_X)):
    print(test_X[i], test_Y[i])
```

Приклад 3.4 – Створення нових наборів даних

```
[0.65888    0.6755202  0.68511987 0.6336     0.6575999 ] 0.63904
[0.6755202  0.68511987 0.6336     0.6575999  0.63904   ] 0.64735985
[0.68511987 0.6336     0.6575999  0.63904    0.64735985] 0.6326399
[0.6336     0.6575999  0.63904    0.64735985 0.6326399 ] 0.65087986
[0.6575999  0.63904    0.64735985 0.6326399  0.65087986] 0.6095998
[0.63904    0.64735985 0.6326399  0.65087986 0.6095998 ] 0.6329601
[0.64735985 0.6326399  0.65087986 0.6095998  0.6329601 ] 0.6329601
[0.6326399  0.65087986 0.6095998  0.6329601  0.6329601 ] 0.6652801
[0.65087986 0.6095998  0.6329601  0.6329601  0.6652801 ] 0.6419201
```

Приклад 3.5 – Створенні нові набори даних для вікна в 5 кроків

3.5 Нейронна мережа MLP

Мережа багатошарового перцептрону має вхідний шар з n вхідними нейронами (протестуємо з одним та п’ятьма), прихований шар з 4 нейронами, і вихідний шар з одним нейроном, який робить прогнозування одного значення. Функція активації – ReLu в прихованому шарі і лінійна в вихідному. Мережа навчається протягом 10 епох і використовується розмір партії 1 (приклад 3.6).

```
model = Sequential()
model.add(Dense(4, input_dim=look_back, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(train_X, train_Y, epochs=10, batch_size=1, verbose=2)
```

Приклад 3.6 – Створення мережі MLP

Після того, як модель навчена, ми можемо оцінити ефективність моделі для прогнозування часових рядів на навчальній та тестовій наборах даних (приклад 3.7).

Інвертуємо прогнози перед тим, як обчислювати оцінки помилок, щоб забезпечити звіт про результати роботи в тих самих одиницях, що й вихідні дані (ціна акцій в день).

```
trainPredict = model.predict(train_X)
testPredict = model.predict(test_X)

trainPredict = scaler.inverse_transform(trainPredict)
train_Y = scaler.inverse_transform([train_Y])
testPredict = scaler.inverse_transform(testPredict)
test_Y = scaler.inverse_transform([test_Y])

train_score = math.sqrt(mean_squared_error(train_Y[0],
trainPredict[:,0]))
```

Приклад 3.7 – Розрахунок RMSE помилки

Середньоквадратична помилка MLP з 1-м вхідним нейроном:

- навчальна вибірка: 1.71 RMSE;
- тестова вибірка: 3.54 RMSE.

Середньоквадратична помилка MLP з 5-ма вхідними нейронами:

- навчальна вибірка: 1.66 RMSE;
- тестова вибірка: 3.17 RMSE.

Тепер, можна генерувати прогнози, використовуючи модель як для навчального, так і для тестового набору даних, щоб отримати прогноз на часовому ряді (рисунки 3.3, 3.4).

Треба змістити прогнози так, щоб вони вирівнювались по осі x із вихідним набором даних. Після підготовки дані складаються в графі, показуючи вихідний набір даних синім кольором, прогнози для навчального набору даних – зеленим кольором, а також прогнози на тестовому наборі даних – червоним кольором (приклад 3.8).

```
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] =
trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] =
testPredict
plt.plot(scaler.inverse_transform(dataset), color='blue')
plt.plot(trainPredictPlot, color='green')
plt.plot(testPredictPlot, color='red')
```

Приклад 3.8 – Генерація часового ряду

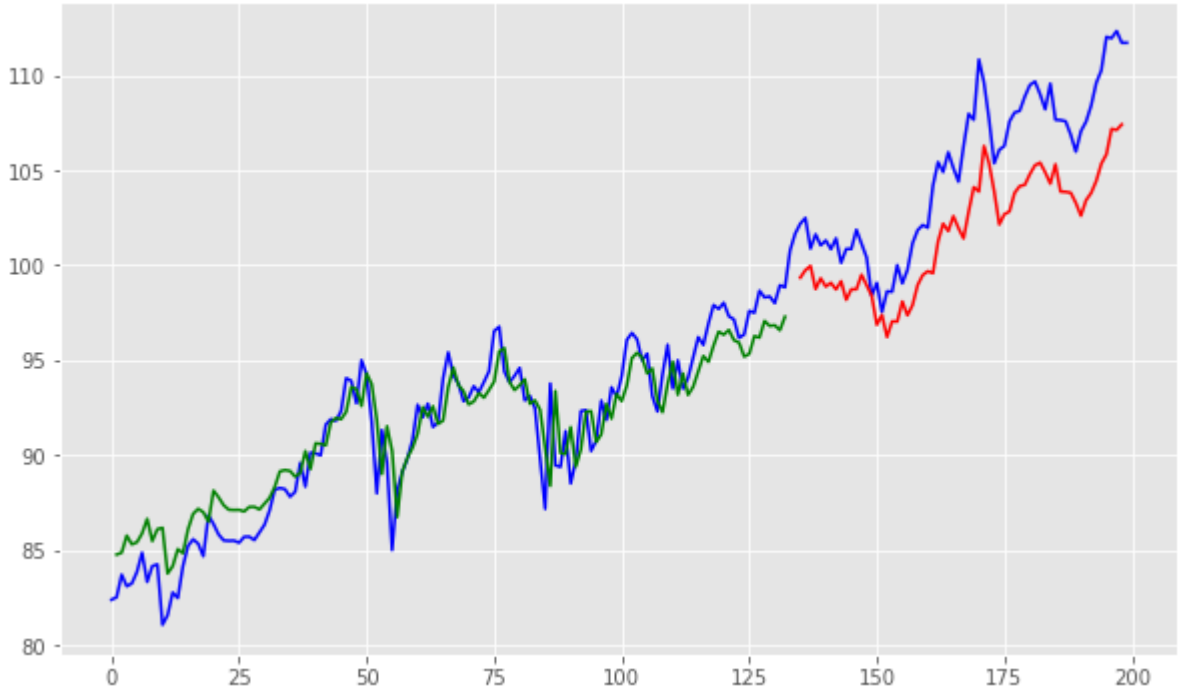

Рисунок 3.3 – Графік прогнозу часового ряду мережею MLP 1-4-1

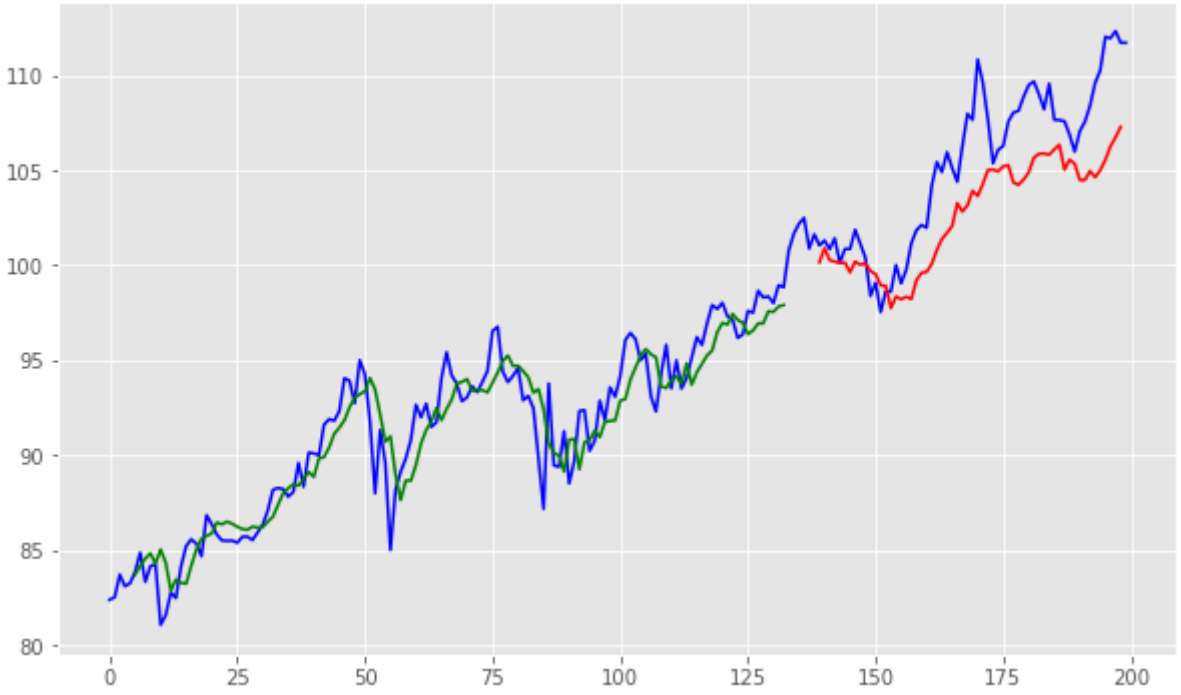

Рисунок 3.4 – Графік прогнозу часового ряду мережею MLP 5-4-1

3.6 Нейронна мережа LSTM

Мережа LSTM очікує, що вхідні дані X будуть забезпечені певною структурою масиву. Треба перетворити підготовлені тренувальні та тестові вхідні дані в очікувану структуру, використовуючи фунцію numpy.reshape() (приклад 3.9).

Замінюємо нейрони прихованого шару на блоки LSTM. Лінійна фунція активації в вихідному шарі, функція сигмоїдної активації за замовчуванням використовується для блоків LSTM. Мережа навчається протягом 10 епох і використовується розмір партії 1 (рисунки 3.5, 3.6):

```
from keras.layers import LSTM

train_X = numpy.reshape(train_X, (train_X.shape[0], 1, train_X.shape[1]))
test_X = numpy.reshape(test_X, (test_X.shape[0], 1, test_X.shape[1]))

model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(train_X, train_Y, epochs=10, batch_size=1, verbose=2)
```

Приклад 3.9 – Створення мережі LSTM

Середньоквадратична помилка LSTM з 1-м вхідним нейроном:

- навчальна вибірка: 1.54 RMSE;
- тестова вибірка: 2.74 RMSE.

Середньоквадратична помилка LSTM з 5-ма вхідними нейронами:

- навчальна вибірка: 1.62 RMSE;
- тестова вибірка: 1.82 RMSE.

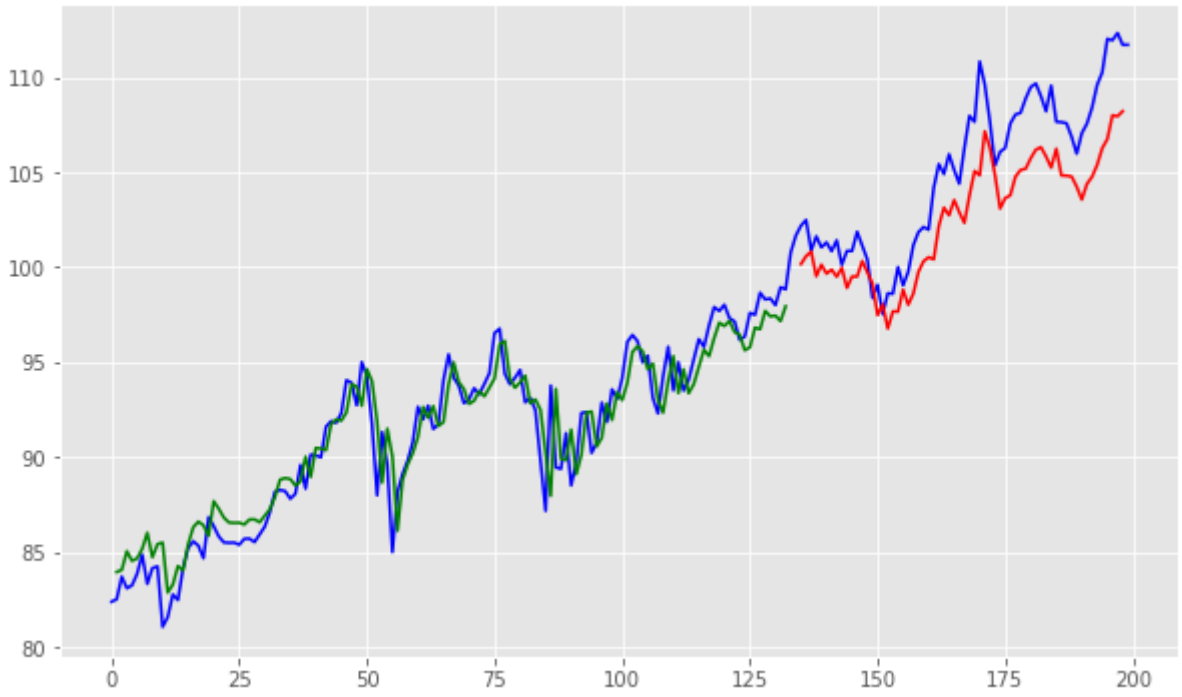

Рисунок 3.5 – Графік прогнозу часового ряду мережею LSTM 1-4-1

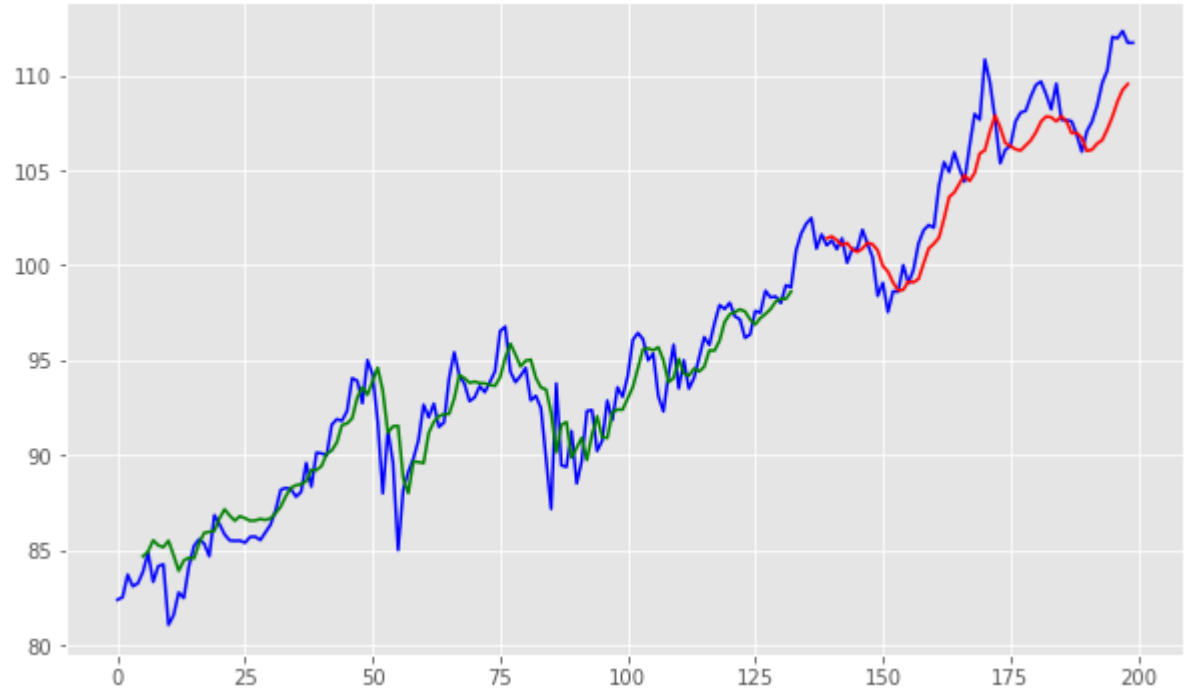

Рисунок 3.6 – Графік прогнозу часового ряду мережею LSTM 5-4-1

Серед розроблених мереж найменшу середньоквадратичну помилку на тестовій вибірці показала мережа довгої короткострокової пам'яті з 5-ма вхідними нейронами, але на навчальній вибірці незначна перевага є у мережі LSTM з одним вхідним нейроном, що може бути списаним на стохастичний характер алгоритму прогнозування.

Аналізуючи графіки прогнозів часового ряду можна побачати більш схожий структурний характер ряду спрогнозованого мережами з одним вхідним нейроном, але значення менш збігаються порівнянно з мережами з декількома вхідними нейронами.

ВИСНОВКИ

У роботі вивчені загальні принципи роботи нейронних мереж: типи архітектур, особливості збору даних, навчання. Показано архітектури які слід використовувати при проведенні прогнозів часових рядів.

Також розглянуто пристрій часових рядів: структура та класифікація, методика їх прогнозування та актуальність.

Було застосовано дві моделі на двох архітектурах нейронних мереж для прогнозування часових рядів акцій Майкрософт. Їх можна використовувати для будь-яких часових рядів, головне – правильно вибрати предобробку даних, визначити архітектуру мережі, оцінити якість роботи алгоритму.

У проекті застосовано два типи архітектури: багатошаровий персептрон та мережа довгої короткострокової пам'яті, кількість вхідних нейронів яких змінювалося для з'ясування їх оптимальної кількості. В якості методу навчання використовувалося навчання з учителем методом зворотного поширення помилки.

Результат дослідження дозволяє стверджувати, що з класу моделей нейронних мереж конфігурація LSTM с декількома вхідними нейронами для вікна даних є оптимальним варіантом при побудові прогнозної моделі, її основною перевагою є більш приближені спрогнозовані значення значення до значень тестової вибірки, на що вказуе найменша число середньоквадратичної помилки RMSE на тестовому наборі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Чучуева, И. А. Модель прогнозирования временных рядов по выборке максимального подобия; Московский государственный технический университет им. Н. Э. Баумана. – М., 2012.

2. Михайлович Т.В., Метод прогнозу водоспоживання з використанням моделі періодичної авторегресії – 2012.

3. Осколкова М. А., Паршаков П. А. Возможности прогнозирования динамики фондового индекса s&p 500 с помощью нейросетевых и регрессионных моделей //Программные продукты и системы. – 2012. – №. 4.

4. Хайкин С. Нейронные сети: полный курс; пер. с англ. М.: Издат. Дом «Вильямс», 2016. 1104 с.

5. Осовский С. Нейронные сети для обработки информации. // Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2-е издание 2016. – 448 с.

6. Рычагов М. Н. Нейронные сети: многослойный перцептрон и сети Хопфилда //Exponenta Pro. Математика в приложениях. – 2003. – №. 1. – С. 29.

7. Бодянский Е. В., Руденко О. Г. Искусственные нейронные сети: архитектуры, обучение, применения. // Харьков: Телетех – 2004. – 372 с.

8. Видмант О. С. Прогнозирование финансовых временных рядов с использованием рекуррентных нейронных сетей LSTM //Общество: политика, экономика, право. – 2018. – №. 5.

9. Hochreiter S., Schmidhuber J. Long Short-Term Memory//Neural Computation, 1997, vol.9, no.8. – C.1735 – 1780.

10. Каллан Р. Нейронные сети краткий справочник. // Вильямс, 2017. – 288 с.

11. Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun & Bengio, Yoshua, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence

Modeling – 2014.

12. Созыкин А. В. Обзор методов обучения глубоких нейронных сетей //Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. – 2017.

13. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский, 2006.

14. Бринк Хенрик и др. Машинное обучение. – Спб.: Питер, 2017. – 336 с.: ил. – (Серия «Библиотека программиста»)

15. Вьюгин В. Математические основы машинного обучения и прогнозирования. – Litres, 2017.

16. Дані курсу акцій Microsoft [Електронний ресурс] – Режим доступу : www / URL: www.nasdaq.com/market-activity/stocks/msft/historical.