МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ»
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»
Завідувач кафедри
_____ Соловйов В.М.
«____» _____ 2018 р.

Реєстраційний № _____
«____» _____ 2018 р.

Алгоритми глибокого навчання у прогнозуванні часових рядів

Кваліфікаційна робота студента
групи Ім-13
ступеня вищої освіти «магістр»
спеціальності
014.09 Середня освіта (Інформатика)
Єгорова Владислава Юрійовича

Керівник: д.ф.-м.н., проф. Соловйов В.М.

Оцінка:
Національна шкала _____
Шкала ECTS __ Кількість балів ___

Голова ЕК _____ _____
Члени ЕК _____ _____
_____ _____
_____ _____

Кривий Ріг — 2018

ЗМІСТ

ВСТУП

На сьогоднішній день для вивчення властивостей складних систем, в тому числі при експериментальних дослідженнях, широко використовується підхід за допомогою аналізу сигналів, створених системою. Це актуально в тих випадках, коли математично описати процес для вивчення практично неможливо, але в наявності є деяка характерна наглядна величина. В даній роботі описується інтервальне прогнозування таких процесів і виділення їх основних характеристик на базі набору статистичних даних. В якості моделі представлення цих даних взято часовий ряд, який прийнято моделювати, використовуючи поняття випадкового процесу.

Часовий ряд – це одна з можливих реалізацій випадкового процесу на деякому обмеженому відрізку індексованого параметру (часу) [2].

Часові ряди є зручним механізмом опису процесів, які працюють за принципом «чорний ящик». В таких системах немає можливості заглянути всередину процесу, а видно лише кінцевий результат. Іншими словами, часові ряди дозволяють аналізувати дані.

Виділяють дві основі цілі аналізу часових рядів: визначення природи ряду (виділення компонентів, оцінка їх параметрів) і використання отриманих оцінок для завдань прогнозування. Для визначення характеристик ряду використовуються експертні та математичні методи, але для прогнозування майбутніх значень ряду подібні рішення не будуть ефективними. Група експертів не може прийняти до уваги всі фактори, а використання математичних формул при аналізі великих масивів потребує великі обчислювальні потужності та часто не є оптимальним засобом для прогнозування в діяльності підприємств.

Технології штучного інтелекту, тобто системи які базуються на штучних нейронних мережах в останні роки все активніше використовуються для

прогнозування часових рядів. Відмінність цього підходу від стандартних закладається в тому, що він дозволяє зробити систему яка навчається самостійно, що важливо для складних задач.

Предметною областю, в якій вже отримали признання нейромережні алгоритми, являється сфера економіки та фінансів, де будь яка допомога в прийнятті рішень може оптимізувати робочі та матеріальні ресурси. В цій області нейромережні алгоритми знайшли своє використання в формі математичного ядра інтелектуальних систем прийняття рішень, експертних систем, нейромережних баз знань та ін. В даній роботі увага приділяється на одному з напрямків використання нейромереж – це використання нейромережних технологій для прогнозування часових рядів.

Актуальність теми. Задача прогнозування часових рядів має високу актуальність для багатьох предметних областей і є невід'ємною частиною діяльності багатьох компаній. Будь яка допомога в прийнятті рішень в бізнес-процесах, може допомогти зменшити затрати, оптимізувати робочі та матеріальні ресурси, а штучні нейронні мережі – це самий перспективний напрямок в області аналізу даних, а задача прогнозування радів – сама популярна задача машинного навчання.

Для того, щоб застосувати технології штучного інтелекту для вирішення практичних задач пов'язаних з аналізом і прогнозуванням часових рядів в даній кваліфікаційній роботі «Алгоритми глибокого навчання у прогнозуванні часових рядів» необхідно вирішити наступні **завдання дослідження**:

1. Провести аналіз предметної області в області глибокого навчання, прогнозування часових рядів;

2. Проаналізувати сутність комплексу задач з прогнозування часових рядів і виділення їх компонент;

3. Побудувати нейромережну модель LSTM;

4. Охарактеризувати вхідні, постійні, проміжні та вихідні дані нейромережної архітектури LSTM;

5. Реалізувати обраний варіант нейромережної моделі LSTM;

6. Провести навчання та тестування нейромережної моделі LSTM;

7. Дослідити результати прогнозування майбутніх значень часових рядів за допомогою нейромережної моделі LSTM.

Об'єкт дослідження: світовий фондовий ринок, представлений індексами фондових ринків окремих країн.

Предмет дослідження: являється прогнозування майбутніх значень фондових індексів методами глибокого навчання.

Структура роботи. Кваліфікаційна робота складається зі вступу, 3 розділів, висновків, списку використаних джерел (32).

У вступі описується актуальність даного дослідження, формується ціль і ставляться задачі, які необхідно вирішити для досягнення цілі.

У першому розділі описуються теоретичні аспекти аналізу часових рядів, розглядаються структурні компоненти ряду та дається визначення поняттю прогнозування. Вивчаються основні дані про штучні нейронні мережі та їх навчанню, розглядаються існуючі програмні рішення.

У другому розділі проводиться розробка та реалізація нейронної мережі для аналізу часових рядів, ставляться задачі з розробки та обґрунтовується вибір засобів розробки. Моделюється та реалізується код штучної нейронної мережі.

У третьому розділі проводиться тестування створеної нейронної мережі з різними вихідними параметрами. Перевіряється ефективність вивчення, проводиться порівняльний аналіз точності прогнозування для різних моделей штучних нейронних мереж.

У висновку наводяться основні висновки з роботи, які були досягнуті в ході виконання роботи.

РОЗДІЛ І. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ МАЙБУТНІХ ЗНАЧЕНЬ ЧАСОВИХ РЯДІВ

1.1 Основні поняття глибокого навчання

Першим кроком до розуміння того, як працює глибоке навчання являється розуміння між декількома важливими термінами, які представлені нижче.

Нейронна мережа (штучна нейронна мережа) – це спроба відтворення роботи людського мозку на комп’ютері за допомогою шарів нейронів.

Штучний інтелект – можливість машини або програми знаходити рішення за допомогою обчислень.

Під час перших досліджень штучного інтелекту вчені намагалися відтворити людський інтелект для вирішення конкретних задач – наприклад, ігри з людиною. Було введено велику кількість правил, яким повинен слідувати комп’ютер. На основі цих правил комп’ютер приймав рішення згідно з конкретним списком можливих дій.

Машинне навчання – сукупність методів машинного навчання для спроби навчити комп’ютери самостійно навчатися на великій кількості даних замість жорстко постульованих правил.

Машинне навчання дозволяє комп’ютерам самостійно навчатися. Це можливо завдяки обчислювальній потужності сучасних комп’ютерів, які можуть легко обробляти великі набори даних.

Контрольоване навчання та неконтрольоване навчання.

Контрольоване навчання (навчання з учителем) – використання помічених наборів даних, що містять вхідні дані та очікувані вихідні результати. Піж час навчання штучного інтелекту за допомогою

контрольованого навчання, подаються як вхідні дані, так і очікувані вихідні результати.

Якщо результат генерований штучним інтелектом неправильний, він скорегує свої розрахунки. Цей ітераційний процес закінчується тоді, коли штучний інтелект припинить робити помилки.

Прикладом задачі штучного інтелекту з контрольованим навчанням являється прогнозування погоди. Штучний інтелект навчається робити прогнозування погоди з використанням історичних даних. Навчальні дані містять вхідні дані (тиск, волога, швидкість вітру) і вихідні результати (температура).

Неконтрольоване навчання (навчання без учителя) – це машинне навчання з використанням наборів даних без визначеної структури.

Під час навчання нейронної мережі неконтрольовано, вона самостійно проводить логічну класифікацію даних. Прикладом задачі з неконтрольованим навчанням є прогнозування поведінки відвідувачів інтернет-магазинів. В такому випадку мережа не навчається на помічених даних. Замість цього вона самостійно класифікує вхідні дані та відповідає на питання, які користувачі частіше всього купують різноманітні товари.

Тепер можна перейти до вивчення глибокого навчання, та розібратися як воно працює.

Глибоке навчання – це метод машинного навчання. Глибоке навчання дозволяє навчати штучний інтелект прогнозувати результати за допомогою вхідного набору даних. Для навчання мережі можна використовувати як контрольоване, так і неконтрольоване навчання.

Для того, аби показати різницю між глибоким та машинним навчанням зручно представити її у вигляді порівняльної діаграми (рис. 1.1).

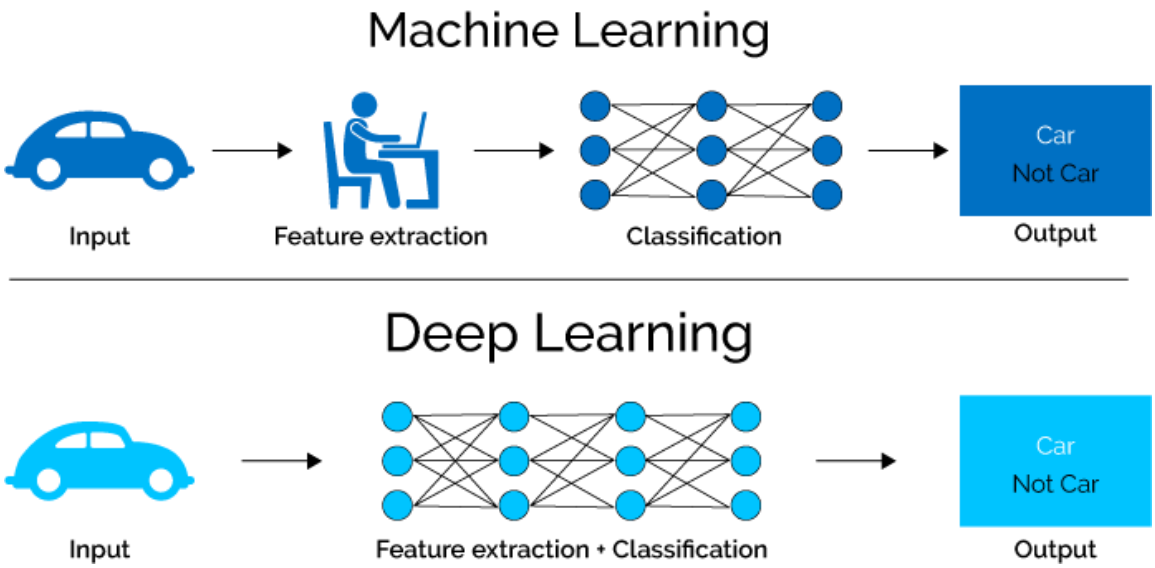


Рисунок 1.1 – Різниця між машинним навчанням (Machine Learning) і глибоким навчанням (Deep Learning)

Глибоке навчання характеризується, як клас алгоритмів машинного навчання, який:

– використовує багатошарову систему нелінійних фільтрів для вилучення ознак з перетворенням. Кожен наступний шар отримує на вході вихідні дані попереднього шару. Система глибокого навчання може поєднувати алгоритми навчання з учителем і без учителя, при цьому аналіз зразка являє собою навчання без учителя, а класифікація – навчання з учителем.

– володіє декількома шарами виявлення ознак або параметрів представлення даних (навчання без учителя). При цьому ознаки організовані ієрархічно, ознаки більш широкої області високого рівня являються похідними від ознак більш низького рівня.

– являться частиною більш широкої області машинного навчання вивчення поданих даних.

– формує в процесі навчання шари на декількох рівнях уявлень, які відповідають різним рівням абстракції; шари створюють ієрархію понять.

Всі визначення констатують:

– наявність декількох шарів нелінійної обробки;

– навчання з учителем або без учителя ознак кожного шару, формуючи ієрархію від низького до високого рівня [27].

Склад конкретних шарів залежить від вирішуваної проблеми. Використовуючи як приховані шари нейронної мережі, так і шари логічних перетворень [28]. Система може містити приховані змінні, організовані пошарово в глибоких генеративних моделях, так як вузли в глибокій мережі довіри чи глибокій обмеженій машині Больцмана.

Алгоритми глибокого навчання протиставлені алгоритмам неглибокого навчання за кількістю параметризованих перетворень, з якими стикається сигнал, що розповсюджується від вхідного до вихідного шару, де параметризованими перетворенням вважається такий блок обробки даних, у якого є навчальні параметри, такі як вага або поріг [29]. Ланцюжок перетворень від входу до виходу називається CAP – шляхом передачі відповідності (англ. credit assignment path, CAP). CAP описують потенційні причинні зв’язки вздовж мережі від входу до виходу, при цьому шлях в різних гілках може мати різну довжину. Для нейронної мережі прямого розподілу (feedforward) глибина CAP не відрізняться від глибини мережі та рівна кількості прихованих шарів плюс один (вихідний шар також параметризований). Для рекурентних нейронних мереж, в який сигнал може переміщуватись через шари минаючи проміжні, CAP через зворотній зв’язок потенційно необмежений в довжині. Не існує універсально узгодженого порогу глибини поділу неглибокого навчання, але зазвичай вважається, що глибоке навчання характеризується нелінійними шарами (CAP > 2) [30].

Глибоке навчання виражається набором алгоритмів машинного навчання для моделювання високорівневих абстракцій використовуючи архітектури, які включають багаточисельні нелінійні перетворення [27-30].

В першу чергу до глибокого навчання відносяться наступні методи та їх варіації:

– визначення системи навчання без учителя, такі як обмежена машина Больцмана для попереднього навчання, автокодувальник, глибока мережа довіри, генеративно-змагальна мережа.

– визначені система навчання з учителем, такі як згортаюча нейронна мережа, яка вивела на новий рівень технології розпізнавання образів.

– рекурентні нейронні мережі, які дозволяють навчатися на процес в часі.

– рекурсивні нейронні мережі, які дозволяють включати зворотній зв’язок між елементами схеми та ланцюжками.

Комбінуючи ці методи створюються складні системи, які відповідають різноманітним задачам штучного інтелекту.

Глибоке навчання являється апробаційною вибіркою із великої кількості методів машинного навчання для представлення даних, найбільш відповідних характеру задачі. Зображення, наприклад, може бути представлено багатьма способами, такими як вектор інтенсивності значень на піксель, або (в більш абстрактній формі) як множина примітивів, областей визначеної форми і т.д. Вдале подання даних полегшує рішення конкретних задач – наприклад, розпізнавання обличчя та виразів обличчя [30]. В системах глибокого навчання автоматизує сам процес вибору і налаштування ознак проводячи навчання ознак без учителя або з частковим залученням учителя, використовуючи для цього ефективні алгоритми та ієрархічне вилучення ознак [31].

Дослідження в цій області дозволили удосконалити моделі роботи з великими об’ємами немаркованих даних. Деякі підходи виникли в результаті досліджень в області нейронаук, успіхів інтерпретації обробки інформації, побудови комунікаційних моделей в нервовій системі, таких як нейронне кодування пов’язане з визначенням відношення між стимулом і нейронними

реакціями і в взаємозв’язками електричної активності між нейронами в головному мозку [32].

Системи глибокого навчання знайшли використання в таких областях, як комп’ютерний зір, розпізнавання мови, аудіо розпізнавання, біоінформатика, де для ряду задач були продемонстровані суттєво кращі результати в порівнянні з минулими.

1.2 Основні поняття аналізу часових рядів

Поняття аналізу часових рядів використовується для того, щоб відрізнити цю задачу від більш простих задач аналізу даних (коли немає природного порядку надходження спостережень) і від аналізу просторових даних, в якому спостереження часто пов’язані з географічним положенням. Часові ряди можуть буди представлені як графічно, так і в табличному вигляді.

Часові ряди відрізняються від простих статистичних вибірок в фіксований момент часу наступними ознаками:

– послідовні в часі показники часових рядів являються взаємопов’язаними, особливо це відноситься до близько розташованих спостережень;

– в залежності від моменту спостереження показники часового ряду мають різну інтенсивність: інформаційна цінність спостережень спадає в міру їх видалення від поточного моменту часу;

– зі збільшенням кількості показників часового ряду точність статистичних характеристик не буде збільшуватися пропорційно кількості спостережень, а при появі нових закономірностей розвитку вона може навіть зменшуватися.

Основою побудови часових рядів є необхідність забезпечення співставлення його окремих показників. Для цього всі елементи повинні характеризувати явище, що вивчається за різні відрізки часу або фіксувати

стан ознаки через однакові інтервали. Кожне значення показника в часовому ряді необхідно розрахувати за єдиною методикою та виражати в одних тих самих одиницях виміру. Кількість вимірів повинно бути досить великим, щоб виявити стійку тенденцію. Також слід враховувати, що використання досить великої кількості значень може призвести до перебільшенню минулих тенденцій, нечутливості тренду до змін.

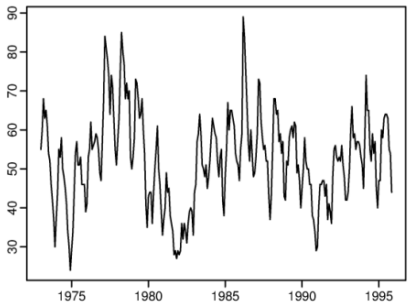
1.2.1 Методи визначення структури часового ряду

Для більш ефективного аналізу та прогнозування часових рядів, необхідно вивчити вихідні графіки та виділити їх основні складові. Визначимо компоненти, які прийнято виділяти в часових рядах [14]:

1. Тренд – динаміка, яка характеризує загальний розвиток часового ряду.
2. Циклічна компонента –динаміка, яка має фазу зростання та спадання, період який займає достатньо великий проміжок часу.
3. Сезонна компонента – це регулярні коливання рівнів ряду в визначений час.
4. Календарні ефекти – стрибки часового ряду, пов’язані з деякими передбачуваними календарними подіями (наприклад, свята або вихідні).
5. Аномальні явища – непередбачувані стрибки, які приводять до різких короткочасних відхиленням ряду від загальної тенденції розвитку.
6. Структурні зрушення – непередбачувані стрибки, які приводять до відхилення ряду від загальної тенденції розвитку, які впливають на його подальшу поведінку.
7. Випадкова компонента – хаотичні рухи достатньо великої частоти, які пов’язані з впливом великої кількості невідомих факторів.

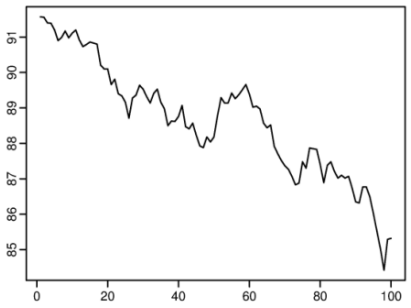
Деякі часові ряди являють собою ту чи іншу компоненту в чистому вигляді, однак на практиці такі ряди зустрічаються вкрай рідко. Загалом, часовий ряд може являти собою змішану комбінацію деяких компонент.

Приклади змішування компонент показані на (рис. 1.2) [16]. Крім цього, далеко не всі часові ряди мають просту структуру для розкладання на вказані компоненти.



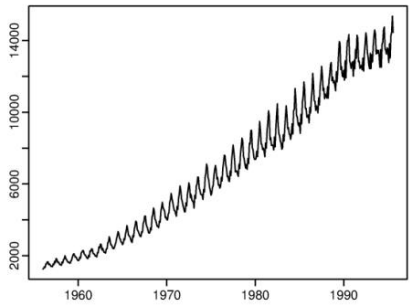
Роки

1. Продаж житла (в мільйонах)



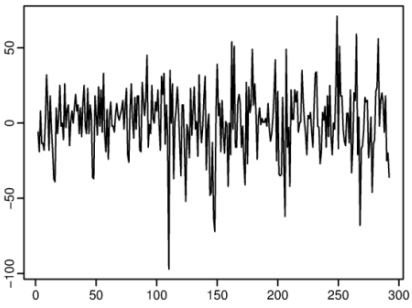
Дні

2. Попит на товар



Роки

3. Виробництво електроенергії



Дні

4. Зміна індексу Доу Джонса

Рисунок 1.2 – Приклади часових рядів реальних даних. 1) Сезонна компонента та циклічна компонента кожні 6-10 років; 2) спадний тренд; 3) зростаючий тренд і сезонна компонента; 4) випадкова компонента

Аналіз часового ряду виконується на основі побудованої моделі, параметри якої підлягають оцінці. Більшість регулярних складових часових рядів належать до двох класів: вони є або трендом, або сезонною компонентною.

Тренд являється систематичною компонентною часового ряду, яка може змінюватися з часом. Трендом називаються невипадкову функцію, яка формується під дією загальних або довготривалих тенденцій, які впливають на

часовий ряд [15]. Прикладом тенденції може бути, наприклад, фактор зросту досліджуваного ринку.

При аналізі тренду виділяють чотири його основних типи:

1. Поліноміальний тренд. Використовують для опису часових рядів з плавною, повільною динамікою, що змінюється з часом.

2. Експоненційний тренд. Доцільно використовувати для швидко зростаючих часових рядів.

3. Гармонійний тренд. Використання гармонійного тренду виправдано, якщо в поведінці часового ряду проглядається періодичність.

4. Тренд, виражений логістичною функцією. Відрізняється від інших тим, що має асимптоту. Часто зустрічається при аналізі часових рядів демографічних показників.

Функції чотирьох основних видів трендів вказані в таблиці 1.1, де

Поліноміальний	$y_t = \alpha_0 + \alpha_1 t + \cdots \alpha_p t^p$
Експоненційний	$y_t = e^{\alpha_0 + \alpha_1 t + \cdots \alpha_p t^p}$
Гармонійний	$y_t = \mathrm{A}\cos(2\pi f t + \varphi)$
Логістичний	$y_t = \frac{k}{1 + be^{-at}}$

Таблиця 1.1 – Основні види трендів часових рядів

Для оцінки параметрів тенденцій першого та другого виду достатньо застосувати метод найменших квадратів (МНК), який зводиться до розв’язку системи лінійних рівнянь.

Параметри тренду третього типу не можуть бути оцінені за допомогою МНК, але якщо частота відома, гармонійний тренд нескладно представити у вигляді лінійної комбінації sin і cos вказаної в формулі 1.1.

$$\text{A}\cos(2\pi ft+\varphi)=\alpha\cos(2\pi ft)+\beta\cos(2\pi ft) \quad (1.1)$$

Далі можна застосувати МНК для оцінки параметрів α і β формула

$$F(\alpha,\beta)=\sum_{i=1}^{n}(y_i-(ax_i+b))^2 \quad (1.2)$$

приймає найменше значення [8]. Тобто при даних a і b сума квадратів відхилень експериментальних даних від знайденої прямої буде найменшою.

Таким чином, рішення зводиться до знаходження екстремуму функцій двох змінних.

1.2.2 Постановка задачі прогнозування майбутніх значень часового ряду

Прогнозування – це передбачення майбутніх подій. Ціллю прогнозування є зменшення ризику при прийнятті рішень. Розглянемо наступну задачу: існує система, що працює за принципом «чорний ящик». Система розглядається як маюча деякий «вхід» для вводу інформації та «вихід» для відображення результатів роботи, при цьому процеси що виконуються в ході роботи системи спостерігачу невідомі. Припускається, що стан виходів функціонально залежить від стану входів.

На вхід такій системі подається набір сигналів $(x_1,\dots,x_m)$, а на виході ми маємо деяку величину $y(x_1,\dots,x_m)$. Результат роботи такої системи можна описати за допомогою матриці, описаної в формулі 1.3

$$(Y,X)=\begin{matrix} y_1 & x_{11} & \dots & x_{1m} \\ y_2 & x_{21} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ y_n & x_{m1} & \dots & x_{mm} \end{matrix} \quad (1.3)$$

Тут $(x_{i1},\dots,x_{im})$ – набір управляючих сигналів (регресори), в i-му вимірі, y_i – результат i-го виміру. Для того, щоб побудувати прогноз по цим

даним, необхідно виявити залежність між набором керуючих сигналів і значеннями на виході.

Теоретичним обґрунтуванням такого підходу являється теорема Такенса. Дана теорема говорить, шо якщо тимчасово ряд породжується динамічною системою, тобто множина значень часового ряду є довільна функція такої системи, то існує таке число d (приблизно рівно ефективному числу степенів свободи даної динамічної системи), що d попередніх значень часового ряду однозначно визначають наступне значення.

В статистичних моделях залежність майбутніх значень від минулих задається рівняннями. До них відносяться: регресивні моделі, авторегресивні моделі, моделі експоненційного згладжування і т. д.

Вибір моделі повинен будуватися на типі використаних даних. Наприклад, регресивна модель використовує значення великої кількості факторів, що впливають на кінцеве значення випадкового процесу. Чим більше факторів, тим точніше прогноз. Авторегресивні моделі пропонують, що значення процесу залежить від деяких попередніх значень того ж процесу, а експоненційне згладжування постійно переглядає прогнозні значення у міру надходження фактичних.

Оскільки деякі залежності можуть мати дуже складний вигляд, на практиці таку функцію вдається знайти не завжди. Тому обмежуються пошуком деякої апроксимуючої залежності. Саме це роблять структурні моделі прогнозування, які задають залежність у вигляді структури та правил переходу по ній. За таким принципом працюють штучні нейронні мережі, моделі на базі ланцюгів Маркова, моделі на базі класифікаційно-регресивних дерев і т. д.

1.3 Структурна модель прогнозування на основі нейронних мереж

Нейронні мережі (НМ) – математичні моделі, а також їх програмна або апаратна реалізація, побудовані за принципом організації функціонування біологічних нейронних мереж – мереж нервових клітин живого організму [13]. Щоб побудувати математичну модель процесів, що відбуваються в мозку, зробимо декілька припущень:

– кожен нейрон володіє деякою передаточною функцією, що визначає умову його збудження в залежності від сили отриманих сигналів; крім того, передаточні функції не залежать від часу;

– при проходженні синапсу сигнал змінюється лінійно, тобто сила сигналу множиться на деяке число; це число називають «вагою» синапсу або вагою відповідного входу нейрона;

– діяльність нейронів синхронізована, тобто час проходження сигналу від нейрону до нейрону фіксований та однаковий для всіх зв’язків; також це відноситься до часу обробки прийнятих сигналів.

Необхідно помітити, що ваги синапсів можуть змінюватися з часом – це принципова особливість, яка дає перевагу цій моделі прогнозування над іншими.

НМ здатні витягувати приховані закономірності з потоку даних. При цьому дані можуть бути неповні, протилежні та навіть завідомо спотворені. Якщо між вхідними та вихідними даними існує якийсь зв’язок НМ здатна автоматично налаштуватися на неї з заданою степеню точності [4].

НМ можна розглядати як мережу «нейронів», організованих в шарах. Предиктори (або вхідні нейрони) створюють нижній шар, а прогнози (або виходи) формують верхній шар. Можуть бути і проміжні шари, що містять «приховані нейрони».

Прості мережі не мають прихованих шарів і еквівалентні регресії. В такому випадку регресія буде більш ефективним методом для вивчення

моделі. Після додавання проміжного шару з прихованими нейронами, НМ стає нелінійною. Нелінійна НМ демонструється на (рис. 1.3).

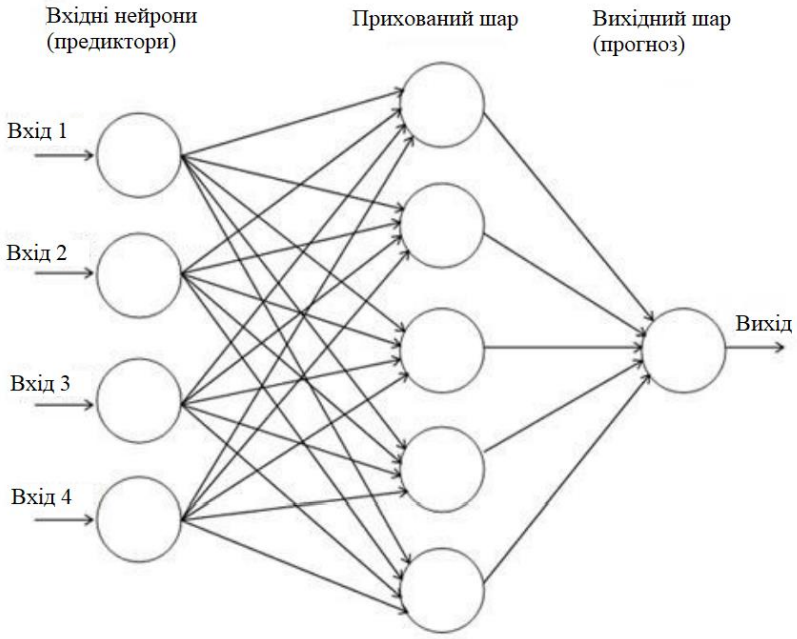


Рисунок 1.3 – Модель нелінійної НМ

Це відомо, як багатошарова одно направлена мережа, де кожен шар вузлів приймає вхідні сигнали від попередніх шарів. Виходи вузлів в одному шарі являються входами до наступного шару. Входи кожного вузла об’єднуються з використанням зваженої лінійної комбінації. Потім результати прихованих шарів і кількість вузлів в кожному прихованому шарі, повинні бути точно визначені завчасно.

1.3.1 Методи навчання штучних нейронних мереж

На початку роботи НМ, її ваги приймають випадкові значення та оновлюються з використанням спостережених даних. Отже, в прогнозах, отриманих за допомогою НМ присутній елемент випадковості. Як правило, роботу НМ перевіряють декілька разів з використанням різних вхідних даних і результати роботи усереднюють. Процес усереднення ваг штучних нейронів з метою отримати бажаний результат на виході, називається навчанням НМ, а вхідні дані для навчання – навчальні параметри або навчальна вибірка.

Навчальна вибірка – вибірка, за якою будується модель залежностей. Бажаний результат називається тестовою вибіркою. За ним оцінюється якість моделей залежностей. Оцінку якості проводять для вибору найкращої моделі. Для ефективної перевірки моделі її перевіряють на даних, які не брали участь в навчанні.

Зазвичай формування навчальних прийомів здійснюється за принципом «ковзного вікна»: тобто береться деякий відрізок ряду і з нього виділяються декілька спостережень, які і будуть представляти собою вхідний вектор. Значенням бажаного виходу в навчальному прикладі буде наступне за порядком спостереження. Потім «ковзне вікно» здвигається на одну позицію в напрямку зросту часу, і процес формування наступної пари навчальної вибірки повторюється.

1.3.2 Архітектура рекурентної нейромережної моделі LSTM

Людина не починає мислити з початку кожної секунди. Читаючи щось, ви розумієте кожне слово на основі розуміння попередніх слів. Ви не відкидаєте всю попередню інформацію, щоб почати мислити з нуля. Ваші думки мають постійність.

Традиційні нейронні мережі позбавлені цієї якості, що є їх основним недоліком. Наприклад, потрібно класифікувати події кінофільму, що відбувають в кожен момент часу. Незрозуміло, яким чином традиційна нейронна мережа могла б використовувати інформацію про попередні події для аналізу наступних.

Рекурентні нейронні мережі (РНС, recurrent neural network, RNN) вирішують цю проблему. Вони містять в собі зворотні зв’язки, що дозволяють зберігати інформацію.

На (рис. 1.4) представлений фрагмент РНС А, який приймає на вході величину x_i і дає на виході величину h_i. Зворотній зв’язок дозволяє передавати інформацію від одного кроку виконання до наступного.

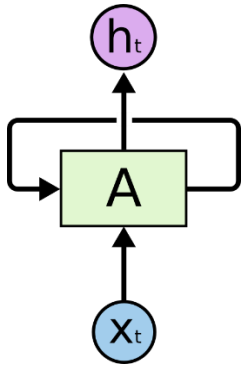


Рисунок 1.4 – Зворотній зв'язок нейронної мережі

Через наявність зворотних зв’язків РНС набувають деяку таємничість. Однак, якщо задуматись, виявляється, що вони не дуже відрізняються від звичайних нейронних мереж. РНС можна представити як множина копій однієї й тієї ж нейронної мережі, кожна з яких передає інформацію наступній. На (рис. 1.5) представлений вигляд РНС, якщо розгорнути зворотній зв’язок

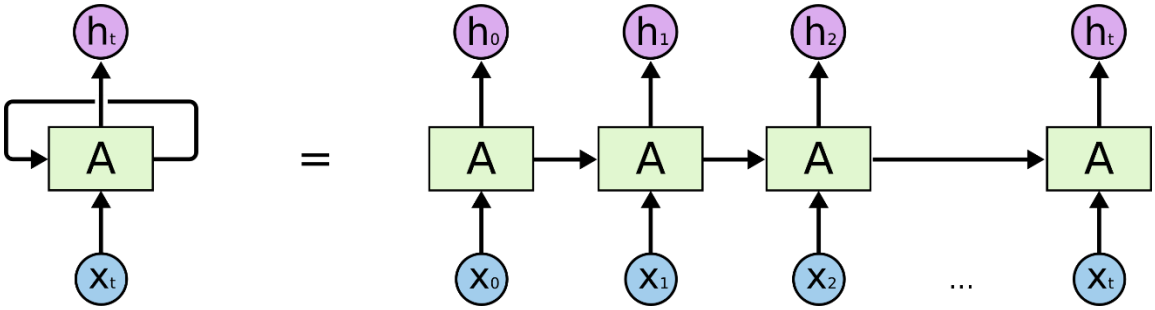


Рисунок 1.5 – Розгорнута РНС

Структура, що має вигляд ланцюга, говорить про те, що РНС тісно пов’язані з послідовностями. Архітектура РНС є природною архітектурою для роботи з таким типом даних.

За останні декілька років РНС з великим успіхом були використані для вирішення різноманітних задач, таких як розпізнавання мови, моделювання мови, переклад, створення описів до зображень та ін. Більша частина успіхів

була досягнута за допомогою особливого типу РНС, так званої LSTM-мережею (long short-term memory, довготривало-короткочасна пам’ять), який при вирішенні різноманітних задач значно перевершує стандартний варіант.

Проблема довготривалих залежностей нейронної мережі LSTM

Однією з ключових ідей РНС є те, що вони повинні бути здатні використовувати попередню інформацію для вирішення поточної задачі, наприклад, використовувати інформацію про попередній кадр відео для обробки поточного кадру. Якщо РНС могли б робити це, вони були б надзвичайно корисні.

Іноді для виконання конкретної задачі нам необхідна тільки недавня інформація. Наприклад, розглянемо мовну модель, яка намагається передбачити наступне слово на основі попередніх. Якщо ми хочемо передбачити останнє слово фрази «хмари в *небі*», то тут немає необхідності додаткового контексту, тому що й так очевидно, що останнім словом буде «небо». В подібних ситуаціях, коли «відстань» від місця появи необхідної інформації до місця її використання невелика, РНС може успішно використовувати цю інформацію (рис. 1.6).

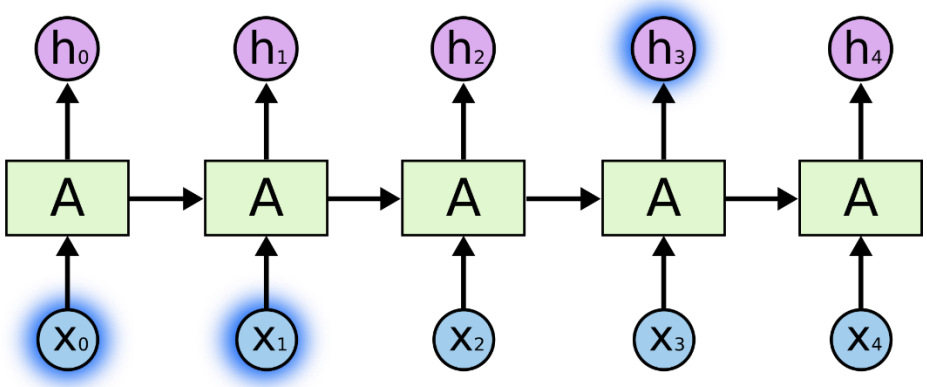


Рисунок 1.6 – Невелика відстань від появи необхідної інформації до місця її використання.

Але бувають випадки, коли необхідний більш широкий контекст. Наприклад, ми намагаємось передбачити останнє слово в тексті: «Я виріс у Франції… Я вільно розмовляю на *французькому*». Локальний контекст говорить нам про те, що наступне слово ймовірно є назвою мови, але щоб визначити, про яку саме мову йдеться, нам необхідний більш широкий контекст, який містить слово «Франція». Необхідна інформація цілком може виявитися дуже «далеко» від того місця, де вона необхідна (рис. 1.7).

Нажаль, при збільшенні цієї «відстані», РНС втрачає можливість використовувати подібну інформацію.

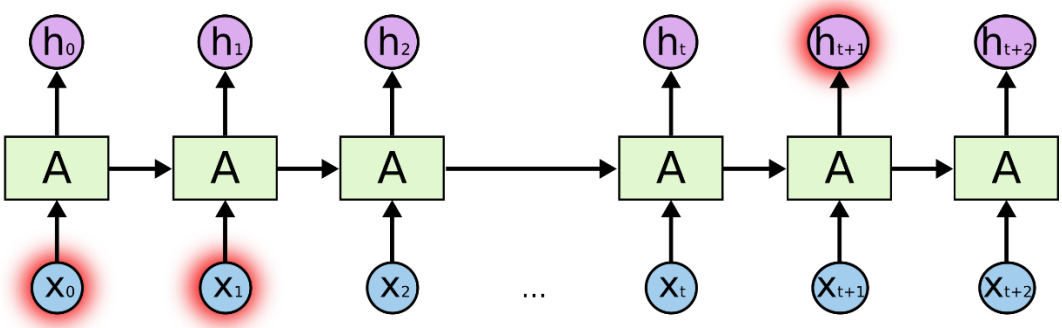


Рисунок 1.7 – Велика відстань від появи необхідної інформації до місця її використання.

В теорії, РНС повинні справлятися з такими «довготривалими залежностями». Можна ретельно підбирати параметри мережі для вирішення експериментальних задач подібного типу. Але в відношенні практичних задач, нажаль, реалізувати це неможливо. Дана проблема була детально вивчена в роботах Hochreiter et al., 1991 і Bengio et al., 1994. В рамках цих досліджень були виявлені фундаментальні причини подібного роду обмежень РНС. Але LSTM-мережі позбавлені цього недоліку.

LSTM-мережа – це особливий тип РНС, здатний навчатися довготривалими залежностями. LSTM-мережі були представлені в роботі Hochreiter and Schmidhuber, 1997, а потім оптимізовані та популяризовані в

багатьох наступних роботах. Такі мережі чудово справляються з вирішенням багатьох задач і знаходять широке використання в даний час.

LSTM-мережі розроблені спеціально для того, щоб вирішити проблему довготривалих залежностей. Зберігання інформації протягом тривалих періодів часу – це їх поведінка за замовченням (рис. 1.8).

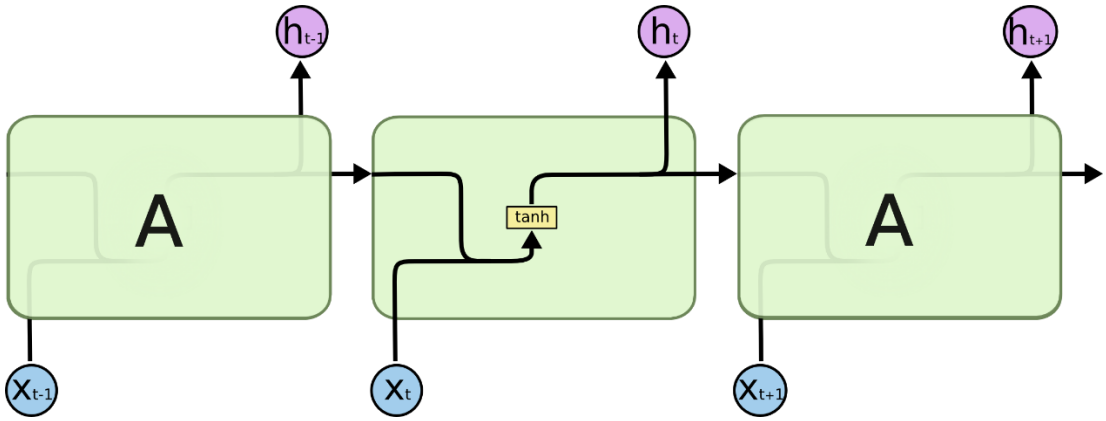


Рисунок 1.8 – Повторюваний модуль стандартної РНС (одношаровий)

Всі РНС мають форму ланцюга повторюваних модулів. Повторюваний модуль стандартної РНС має дуже просту структуру, наприклад, єдиний tanh-шар (функція активації – гіперболічний тангенс).

LSTM-мережа являє собою аналогічний ланцюг, але повторюваний модуль має іншу структуру. Замість одного шару він містить чотири шари, які взаємодіють особливим чином (рис. 1.9).

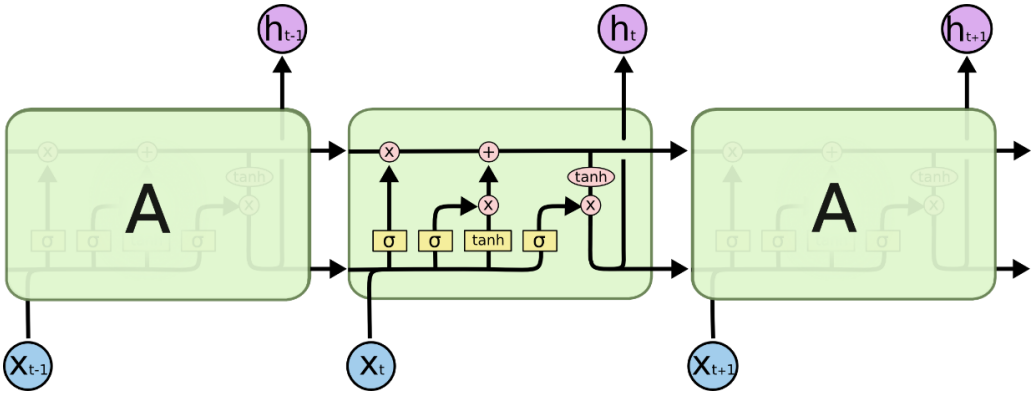
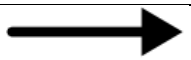


Рисунок 1.9 – Повторюваний модуль LSTM-мережі (чотири взаємодіючих шари)

Для розбору представленої вище схеми, будемо використовувати умовні позначення (таблиця 1.2):

	Шар нейронної мережі
	Операція
	Передача вектору
	Об’єднання
	Копіювання

Таблиця 1.2 – Умовні позначення

На представленій вище схемі LSTM-мережі кожна лінія означає передачу вектору з виходу одного вузла на вхід іншим. Рожеві кружечки означають операції, наприклад, додавання векторів, а жовті прямокутники – навчені шари. Злиття ліній передбачає об’єднання, а розгалуження лінії говорить про те, що інформація копіюється, і копії відправляються в різні точки призначення. Ключовою особливістю LST-мережі є стан комірки (cell state), представлений горизонтальною лінією у верхній частині (рис. 1.10).

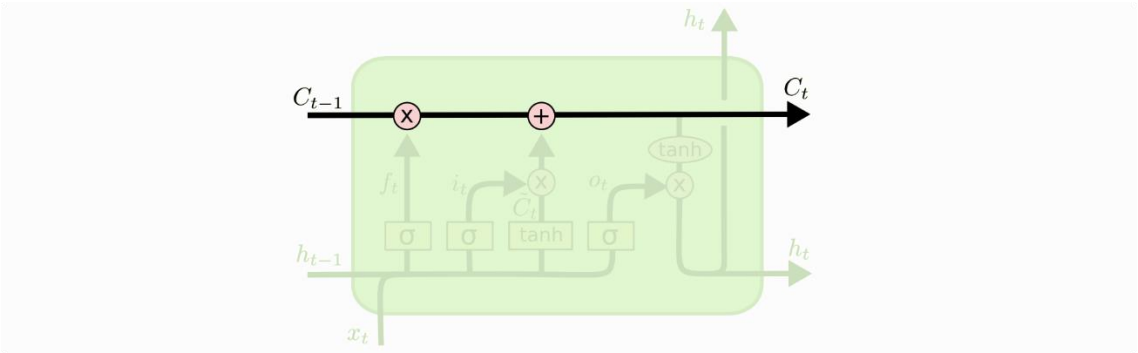


Рисунок 1.10 – Стан комірки (cell state)

Стан комірки можна порівняти з конвеєрною стрічкою. Він проходить через весь ланцюг, лише з незначними лінійними взаємодіями.

LSTM-мережа має можливість видаляти та додавати інформацію в стан комірки. Цей процес регулюється спеціальними структурами, так званими гейтами (gate).

Гейт – це механізм, який дозволяє пропускати інформацію вибірково. Він складається з sigmoid-шару (функція активації – сигмоїда) (рис. 1.11), і операції точкового множення.

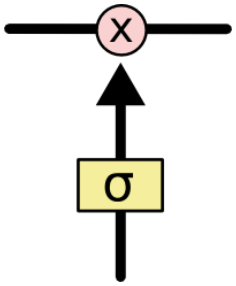


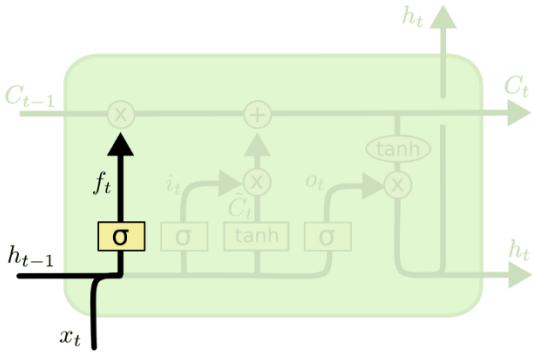
Рисунок 1.11 – Сигмоїдна функція

Виходом sigmoid-шару є число від 0 до 1, яке визначає рівень пропускання. Нуль означає «не пропустити нічого», а одиниця – «пропустити все». Комірка має три гейти, що керують її станом.

1.2.4 Принцип роботи LSTM-мережі

На першому етапі необхідно вирішити, яку інформацію потрібно видалити з стану комірки. Це рішення приймає sigmoid-шар, що називається «гейтом забування» (forget gate). Він приймає на вході h_{t-1} і x_t і дає на виході число від 0 до 1 для кожного значення в стані комірки C_{t-1}. Одиниця означає «повністю зберегти», а нуль – «повністю видалити» (рис. 1.12).

Повертаючись до мовної моделі, яка намагається передбачити наступне слово на основі попередніх слів. В даному випадку стан комірки може зберігати суб’єкт, про який йде мова, що дозволить використовувати правильні займенники. Коли з’являється новий рід суб’єкта, необхідно «забути» рід попереднього.

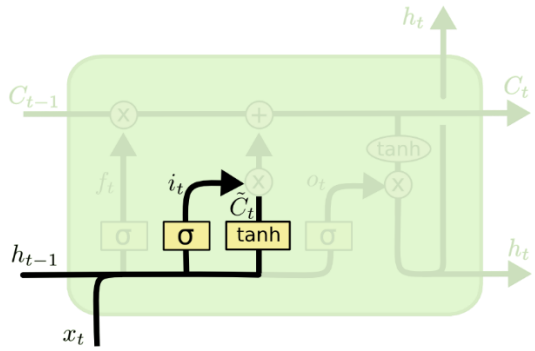


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рисунок 1.12 – Перший етап процесу LSTM-мережі

На наступному етапі необхідно вирішити, яку нову інформацію слід записати в стан комірки (рис. 1.13). Цей етап ділиться на дві частини. Спочатку sigmoid-шар, тобто «вхідний гейт» (input gate), вирішує, які значення необхідно оновити. Потім tanh-шар створює вектор нових значень-кандидатів C_t, які можуть бути додані в стан комірки.

У випадку мовної моделі ми хочемо записати в стан комірки рід нового суб'єкта на заміну роду попереднього суб'єкта, який необхідно «забути».



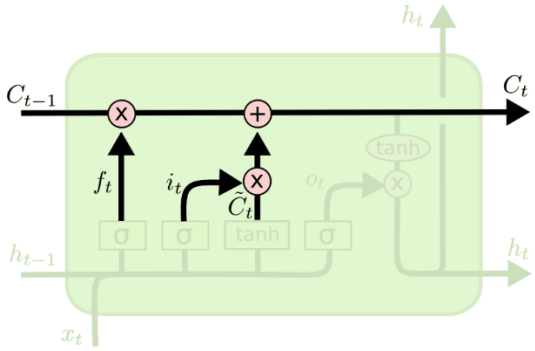
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рисунок 1.13 – Другий етап процесу LSTM-мережі

Тепер необхідно оновити попередній стан комірки C_{t-1} до поточного стану C_t. На попередніх етапах уже було вирішено, що необхідно зробити, тепер залишилося тільки виконати це.

Ми множимо попередній стан комірки на f_t, «забуваючи» таким чином те, що раніше було вирішено «забути». Потім додаємо $i_t * C_t$ – нові значення-кандидати, масштабовані відповідним чином.

У випадку мовної моделі на цьому етапі ми безпосередньо видаляємо інформацію про рід попереднього об’єкта та додаємо нову інформацію згідно з рішенням, прийнятим на попередніх етапах (рис. 1.14).

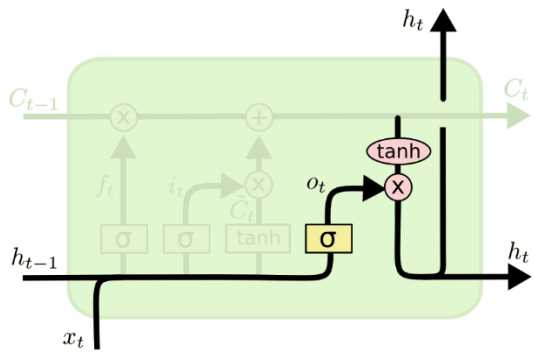


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рисунок 1.14 – Третій етап процесу LSTM-мережі

Нарешті, необхідно вирішити, що потрібно відправити на вихід (рис. 1.15). Вихід буде являти собою відфільтрований стан комірки. Спочатку sigmoid-шар вирішує, які елементи стану комірки необхідно передати на вихід. Потім стан комірки змінюється за допомогою tanh-шару до інтервалу від -1 до 1 і множиться на вихід sigmoid-шару, щоб вивести тільки те, що було вирішено вивести.

У випадку мовної моделі, оскільки в її поле зору тільки но з’явився новий суб’єкт, логічно буде вивести, наприклад, інформацію пов’язану з дієсловом, якщо наступним словом повинно бути дієслово. Зокрема, вона могла б вивести граматичне число, що дозволило б визначити правильну форму дієслова.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 1.15 – Четвертий етап процесу LSTM-мережі

1.4 Аналіз існуючих розробок нейромережного прогнозування

1.4.1 Аналіз існуючих програмних продуктів

Сьогодні розроблено велику кількість програмних продуктів для використання технології нейрообчислень. Існують універсальні нейромережні пакети, призначення для розв’язання будь яких задач, які можна вирішити за допомогою НМ, від розпізнавання мови і образів до вирішення задач прогнозування, але як показує практика такі програмні продукти не завжди зручні для вирішення задач прогнозування часових рядів. Існує клас нейромережних програмних продуктів, призначених виключно для вирішення задач прогнозування часових рядів. Найбільш популярні сьогодні наступні програмні продукти, які реалізують нейромережні підходи до вирішення задач з програмування.

1. Matlab – настільна лабораторія для математичних обчислень, проектування електричних схем і моделювання складних систем. Має вбудовану мову програмування і багатий інструментарій для нейронних мереж – AnfisEditor (навчання, створення, тренування і графічний інтерфейс), командний інтерфейс для програмного завдання мереж, nnTool – для більш тонкої конфігурації мережі.

2. Statistica – потужне забезпечення для аналізу даних і пошуку статистичних закономірностей. В даному пакету робота з нейромережами представлена в модулі STATISTICA Neural Networks (скорочено, ST Neural Networks, нейронно-мережний пакет фірми StatSoft), представляє собою реалізацію всього набору нейромережних методів аналізу даних.

3. BrainMaker – призначений для вирішення задач, для який не знайдені формальні методи та алгоритми, а вхідні дані неповні, зашумлені і

суперечливі. До таких відносяться біржові і фінансові прогнози, моделювання кризових ситуацій, розпізнавання образів та ін.

4. NeuroShell DayTrader – нейромережна система, орієнтована для використання в трейдерській діяльності. Програма являється вузькоспеціалізованою і підходить для торгівлі.

Кожне з розглянутих рішень являється дорогим комерційним продуктом. Наприклад, пакет прикладних програм Matlab оцінюється в 2650$, а модуль Neural Network Toolbox в 1250$. Подібне програмне забезпечення недоступне для особистого використання і ускладнюють використання сучасних технологій штучного інтелекту в бізнес-процесах комерційних підприємств. Крім цього, вони не дозволяють використовувати вихідний код модельованої НМ для особистих розробок.

1.4.2 Аналіз існуючих бібліотек компонентів

Бібліотека в програмуванні – збірник програм або об’єктів, що використовуються для розробки програмного забезпечення [2]. Основним призначенням представлених нижче бібліотек є інтеграція нейромережних технологій в особисті інформаційні та програмні системи, для розширення їх аналітичних можливостей. Реалізація НМ у вигляді компонентів і наявність відкритого коду дозволяє легко вбудовувати подібні бібліотеки в інші програми. Об’єктно орієнтоване виконання придає їх особливу гнучкість для оптимізації під різні задачі. Самі розповсюджені нейромережні бібліотеки для роботи з часовими рядами:

1. ANN – безкоштовна бібліотека з відкритим вихідним кодом для створення НМ на мові PHP. Вихідний код заснований на роботі Едді Янга. ANN є вільно розповсюдженою бібліотекою при умові зберігання інформації про авторство в вихідному коді.

2. FANN – це безкоштовна бібліотека з відкритим вихідним кодом для створення НМ. Сьогодні бібліотека FANN доступна майже для всіх мов

програмування і середовищ розробки. Вона проста у використанні, універсальна і добре документована.

3. Encog – безкоштовна бібліотека для створення НМ на мовах Java, C# і C++. Encog підтримує різноманітні алгоритми навчання, такі як баєсовські мережі та приховані Марковські моделі. Його основна сила закладається в нейромережних алгоритмах. Encog має класи для створення мереж з різноманітними архітектурами, а також допоміжні класи для нормалізації та обробки НМ.

4. NeuralBase – це бібліотека з відкритим вихідним кодом, реалізована мережа Хопфілда та багатошарова нейронна мережа, що навчається за алгоритмом зворотного розповсюдження. Бібліотека є інтелектуальною власністю компанії BaseGroup Labs – професіонального постачальника програмних продуктів і рішень в області аналізу даних.

Keras – відкрита нейромережева бібліотека, написана мовою Python. Вона здатна працювати поверх Deeplearning4j, TensorFlow та Theano. Спроектовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була мінімальною, модульною та розширюваною. Її було створено як частину дослідницьких зусиль проекту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором та підтримувачем є Франсуа Шоллє (фр. François Chollet), інженер Google.

Різноманіття доступних бібліотек дозволяє використовувати їх у вирішенні будь якої задачі незалежно від обраної мови програмування та архітектури НМ. Відкритий вихідний код дає можливість для тонкого налаштування розробленої програмної системи, що особливо важливо в задачах про прийняття рішень.

Висновки до розділу 1

Останнім часом точність штучний нейронних мереж дуже швидко зростає. Існує множина прикладів, коли нейронні мереж з’являються в якійсь області та повністю замінюють класичні алгоритми вирішення задач. Для того щоб використання штучний нейронних мереж було виправдано, необхідно щоб задача мала наступні ознаки:

– Є достатньо велика множина прикладів, не відомі принципи рішення задачі (відсутній алгоритм);

– Рішення задачі передбачає розгляд великого об’єму вхідної інформації;

– Дані неповні, надлишкові, частково суперечливі.

Виходить, що використання штучний нейронних мереж добре підходять для рішення задач класифікації, розпізнавання образів, проведення оптимізації та прогнозування.

РОЗДІЛ ІІ. ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖНОЇ МОДЕЛІ LSTM

2.1 Обґрунтування вибору засобів розробки

2.1.1 Обґрунтування вибору мови програмування

Для реалізації програмного додатку для аналізу часових рядів було обрано Python – мова програмування високого рівня загального призначення, орієнтована на підвищення швидкодії розробника та читаності коду. Синтаксис ядра Python мінімалістичний. Одночасно стандартна бібліотека включає великий об’єм корисних функцій.

Python підтримує декілька парадигм програмування, в тому числі структурне, об’єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване.

Основі архітектурні риси – динамічна типізація, автоматичне управління пам’яттю, повна інтроспекція, механізм обробки виключень, підтримка багато потокових обчислень і зручні високо рівневі структури даних. Код в Python організовується в функції та класи, які можуть об’єднуватися в модулі (вони в свою чергу можуть об’єднуватися в пакети).

2.1.2 Обґрунтування вибору середовища розробки

Основними вимогами до вибору середовища розробки є підтримка мови програмування Python і можливість інтерактивного вводу та виводу даних, таким середовищем являється IPython – потужний інструмент для роботи з мовою Python. Базові компоненти IPython – це інтерактивна оболонка з широким набором можливостей і ядро для Jupyter. Jupyter notebook являє собою графічну веб оболонку для IPython, яка розширює ідею консольного підходу до інтерактивних обчислень.

Основі особливості даної платформи – це комплексна інтроспекція об’єктів, зберігання історії вводу протягом усіх сеансів, хешування вихідних

результатів, розширювана система «магічних» команд, додатковий командний синтаксис, підсвічування коду, доступ до системної оболонки, стикування з pdb відладчиком і Python профайлером.

IPython дозволяє підключати багатьох клієнтів до одного обчислювального ядра і завдяки своїй архітектурі, може працювати в паралельному кластері.

В Jupyter notebook можна розробляти, документувати та виконувати додатки на мові Python, він складається з двох компонентів: веб-додаток і ноутбуки – файли, в яких можна працювати з вихідним кодом програми, запускати його, вводити та виводити дані та ін.

Веб додаток дозволяє:

- редагувати Python код в браузері, з підсвічуванням синтаксису, автовідступами та автодоповненням;
- запускати код в браузері;
- відображати результати обчислень з медіа представленням (схеми, графіки);
- працювати з мовою розмітки Markdown і LaTeX.

Ноутбуки – це файли в яких зберігається вихідний код і вихідні дані, отримані в рамках сесії. Фактично він є записом вашої роботи, але при цьому дозволяє заново виконати код, присутній в ньому. Ноутбуки можна експортувати в формати PDF, HTML.

2.1.3 Обґрунтування використання бібліотек компонентів

Для програмної реалізації архітектури нейронної мережі та значень можна використовувати готові безкоштовні бібліотеки компонентів, призначених для роботи з часовими рядами. Для цих цілей була обрана бібліотека Keras – відкрита нейромережна бібліотека, написана на мові програмування Python. Вона являє собою надстройку над фреймворками

Deeplearning4j, TensorFlow і Theano. Націлена на оперативну роботи з мережами глибокого навчання, при цьому спроектована так, щоб бути компактною модульною та розширюваною.

Ця бібліотека містить багаточисельні реалізації широко використовуваних будівельних блоків нейронних мереж, таких як шари, цільові та передаточні функції, оптимізатори, і множину інструментів для спрощення роботи з зображеннями і текстом.

Зручність у використанні. Keras – API, призначений для людей, а не машин тому зручний у використанні. Це робить користувальницький досвід основою та центром. Keras дотримується практики зменшення когнітивного навантаження: він пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для звичайних випадків використання та забезпечує чіткий та дієвий відгук щодо помилок користувача.

Модульність. Модель розуміється як послідовність, або графік автономних, повністю налаштованих модулів, які можуть бути підключені разом із якомога меншими обмеженнями. Зокрема нейронні шари, функції витрат, оптимізатори, схеми ініціалізації, функції активації, схеми регуляризації – це автономні модулі, які можна поєднати для створення нових моделей.

Легко розширювана. Нові модулі легко додавати (як нові класи та функції), а існуючі модулі наводять достатньо прикладів. Можливість легко створювати нові модулі дозволяє тотальну виразність, що робить Keras придатним для передових досліджень.

Робота з Python. Немає окремих файлів конфігурації моделей у декларативному форматі. Моделі описані в коді Python, який є компактним, простим у налагодження та легко розширюваний.

2.2 Представлення вхідних даних

Ефективність навчання НМ залежить від об’єму навчальної вибірки. В даному програмі на першій епосі навчання НМ буде розглядатися тільки 10 перших значень часового ряду і після цього робити випадкове припущення про одинадцятий.

Епоха – один крок в процесі навчання, включає представлення всіх прикладів із навчальної множини [19].

Порівнюючи прогноз одинадцятого значення множини НМ буде корегувати ваги своїх нейронів враховуючи обчислену помилку прогнозування. Чим більше значень із навчальної множини розгляне НМ і чим більше епох вона зробить, тим більш точними будуть прогнозовані значення.

Вхідні та вихідні дані для програмного комплексу будуть представлені в форматі .csv (від анг. Comma-Separated Values) – текстовий формат, призначений для представлення табличних даних. Файли такого формату можна відкривати як у звичайному редакторі, так і в офісних програмних продуктах. Багато додатків (Microsoft Excel, Google Docs та ін.) підтримують імпорт та експорт даних в форматі .csv, що робить його зручним для роботи та дозволяє використовувати значення для навчальної вибірки.

Формат .csv має наступні вимоги для навчальної множини:

1. Кожен рядок відповідає одній даті під час якої були зроблені виміри.
2. Перший рядок файлу повинен мати назву стовпчиків в латиниці.
3. Назва стовпчиків повинні бути розділені комою. Дата і наступні значення повинні бути розділені комою. Значення в усіх стовпчиках повинні бути розділені комою.
4. Кожен рядок повинен закінчуватися символом переносу рядка.
5. Ціла та дробова частина в дійсних чисел повинна бути розділена крапкою.

Для тестування програми, в якості вхідних даних для НМ було обрано масив реальних даних (ціни акцій компанії Apple) з 2005 по 2018 рік (рис. 2.1).

AAPL (1).csv

```
Date,Open,High,Low,Close,Adj Close,Volume
2004-12-31,4.635000,4.642857,4.573571,4.600000,3.080449,69647200
2005-01-03,4.627143,4.650714,4.471428,4.520714,3.027354,172998000
2005-01-04,4.556428,4.676429,4.497857,4.567143,3.058446,274202600
2005-01-05,4.604286,4.660714,4.575000,4.607143,3.085232,170108400
2005-01-06,4.619286,4.636428,4.523571,4.610714,3.087624,176388800
2005-01-07,4.642857,4.973571,4.625000,4.946429,3.312439,556862600
2005-01-10,4.987857,5.050000,4.848571,4.925714,3.298568,431327400
2005-01-11,4.875000,4.939286,4.581429,4.611429,3.088102,652906800
2005-01-12,4.675000,4.707143,4.521429,4.675714,3.131152,479925600
2005-01-13,5.265000,5.315714,4.980714,4.985714,3.338748,791179200
2005-01-14,5.017857,5.122857,4.942143,5.014286,3.357881,442685600
2005-01-18,4.989286,5.050000,4.839286,5.046429,3.379406,251615000
2005-01-19,5.035000,5.104286,4.982143,4.991428,3.342574,187973800
2005-01-20,4.975000,5.090714,4.962143,5.032857,3.370318,228730600
2005-01-21,5.093572,5.114286,5.000000,5.035000,3.371752,227833200
2005-01-24,5.070000,5.127143,5.039286,5.054286,3.384666,210407400
2005-01-25,5.097857,5.202857,5.067143,5.146429,3.446372,242307800
2005-01-26,5.190000,5.196429,5.087143,5.160714,3.455938,184874200
2005-01-27,5.154286,5.208571,5.110714,5.188571,3.474593,124056800
2005-01-28,5.187143,5.284286,5.174286,5.284286,3.538689,200403000
```

Рисунок 2.1 – Представлення даних в форматі .csv. Навчальна вибірка (ціни акцій компанії Apple) – шість стовпчиків зі значеннями

2.3 Програмна реалізація нейромережної моделі LSTM

2.3.1 Побудова нейромережної моделі LSTM

LSTM – (Long short-term memory) є модифікацією RNN (Recurrent neural network). Дана модель була обрана з причини, що при прогнозуванні вона враховує так званий «контекст». Тобто дані, які вона бачила до цього. Нижче на (рис. 2.2) наведена конфігурація обраної нейронної мережі:

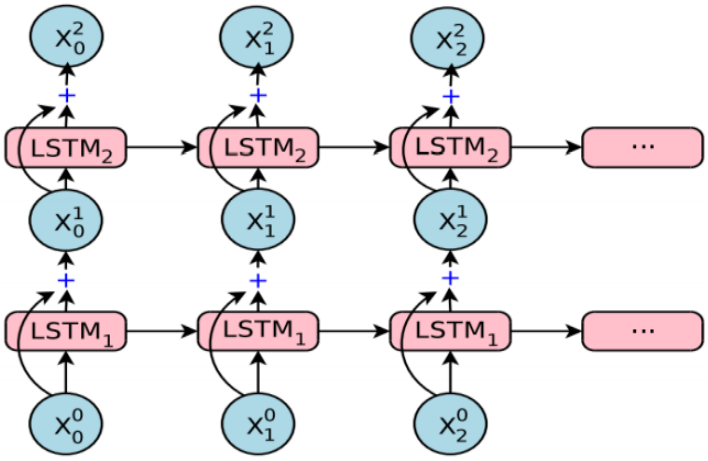


Рисунок 2.2 – Графічне представлення LSTM

Рекурентні нейронні мережі – вид нейронних мереж, де зв'язки між елементами створюють направлену послідовність (рисунок 2.3). Завдяки цьому з'являється можливість обробляти серії подій в часі від багатошарових перцептронів, рекурентні мережі можуть використовувати пам'ять для обробки послідовностей довільної довжини. Тому мережі RRN використовуються в таких задачах, де дещо цілісне розбито на сегменти, наприклад: розпізнавання рукописного тексту або розпізнавання мови. Було запропоновано багато різних архітектурних рішень для рекурентних мереж від простих до складних [9].

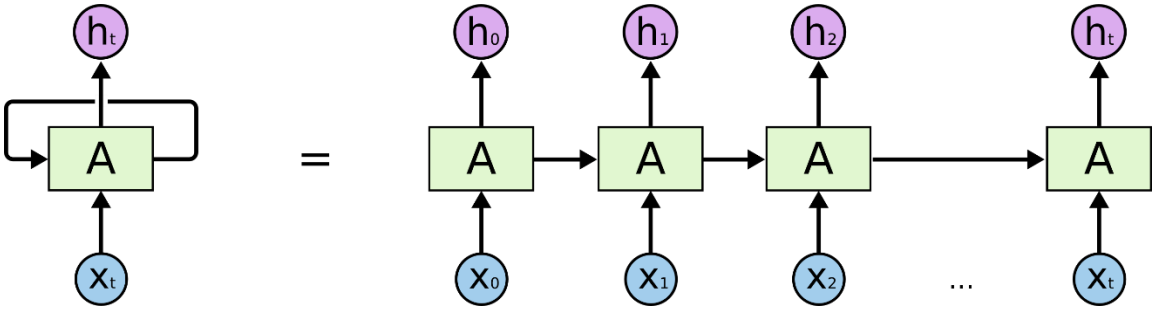
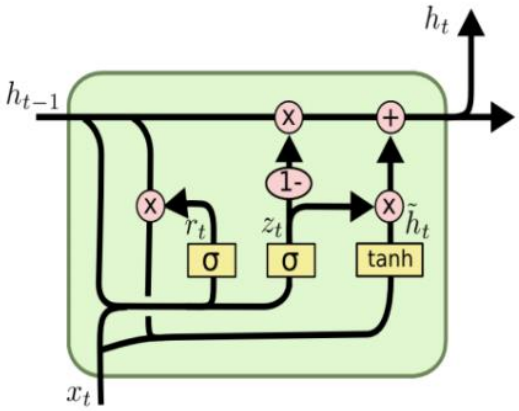


Рисунок 2.3 – Рекурентна мережа в розгорненні

Останнім часом велику популярність отримали мережі з довготривалою і короткотривалою пам'яттю (LSTM) і керуючий рекурентний блок (GRU) [9].



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Рисунок 2.4 – Варіація LSTM мережі

Довга короткочасна пам’ять (Long short-term memory) – різновид архітектури рекурентних нейронних мереж. Як і більша частина рекурентних нейронних мереж, LSTM-мережа являється універсальною в тому сенсі, що при достатній кількості елементів мережі вона спроможна виконати будь-яке обчислення, на яке спроможний звичайний комп’ютер, для чого необхідна відповідна матриця ваг, яка може бути розглянута як програма. На відміну від класичних рекурентних нейронних мереж, LSTM-мережа добре пристосована до навчанню на задачах класифікації, обробки та прогнозування часових рядів у випадках, коли важливі події розділені часовими лагами з невизначеною тривалістю та границями [13].

2.3.2 Реалізація та навчання нейромережної моделі LSTM

Для початку необхідно отримати історичні дані про ціну акцій компанії APPLE. Для цього завантажується файл з даними, поданий у форматі .csv (рис. 2.5).

```
hist = pd.read_csv('./AAPL.csv')[:]
hist = hist.set_index('Date')
hist.index = pd.to_datetime(hist.index)
hist.head()
```

Рисунок 2.5 – Завантаження даних з файлу

Для наглядного представлення даних доцільно вивести їх на екран у вигляді таблиці, але через досить велику кількість даних беруться лише перші п’ять та останні п’ять значень взятих із файлу (рис. 2.6).

Date	Open	High	Low	Close	Adj Close	Volume
2008-01-10	25.368572	25.857143	25.058571	25.431429	17.030483	370743800
2008-01-11	25.142857	25.407143	24.285715	24.670000	16.520582	308071400
2008-01-14	25.360001	25.631428	25.024286	25.540001	17.103188	275112600
2008-01-15	25.388571	25.602858	23.522858	24.148571	16.171404	585819500
2008-01-16	23.604286	24.144285	22.385714	22.805714	15.272138	553461300

Рисунок 2.6 – Подання історичних даних у вигляді таблиці

Завантажені дані були розділені на тренування та тестовий набір. Використовувалися останні 10% даних для тестування, які розбиваються датою 2017-01-12. Все дані до цієї дати використовуються для навчання, все дані з цієї дати використовуються для тестування навчальної моделі. Нижче, побудований графік стовпчика close вхідних даних, яка являється щоденною ціною закриття, яку буде прогнозовано (рис. 2.7).

Рисунок 2.7 – Розбиття історичних даних про ціну акцій Apple

Для навчання LSTM моделі дані були розділені на вікна 30 днів (кількість довільна) (рис. 2.9). В кожному вікні дані нормалізуються до нульової бази, тобто перший запис кожного вікна рівний нулю і всі інші значення являють собою зміну по відношенню до першого значення (рис. 2.8). В результаті відбувається прогнозування зміни ціни, а не абсолютна ціна.

```
def normalise_zero_base(df):
    # Нормалізація стовпчиків даних для відображення зміни
    return df / df.iloc[0] - 1
def normalise_min_max(df):
    # Нормалізація стовпчиків даних мін/макс
    return (df - df.min()) / (data.max() - df.min())
def extract_window_data(df, window_len=10, zero_base=True):
    # Перетворення даних в послідовність вікон `window_data`.
    window_data = []
    for idx in range(len(df) - window_len):
        tmp = df[idx: (idx + window_len)].copy()
        if zero_base:
            tmp = normalise_zero_base(tmp)
        window_data.append(tmp.values)
    return np.array(window_data)
```

Рисунок 2.8 – Нормалізація даних

```
def prepare_data(df, target_col, window_len=10, zero_base=True, test_size=0.2):
    # Підготовка даних для LSTM
    # Навчально тестове розділення
    train_data, test_data = train_test_split(df, test_size=test_size)
    # Витягування даних вікна
    X_train = extract_window_data(train_data, window_len, zero_base)
    X_test = extract_window_data(test_data, window_len, zero_base)
    # Витягування цільового стовпчика
    y_train = train_data[target_col][window_len:].values
    y_test = test_data[target_col][window_len:].values
    if zero_base:
        y_train = y_train / train_data[target_col][:-window_len].values - 1
        y_test = y_test / test_data[target_col][:-window_len].values - 1
    return train_data, test_data, X_train, X_test, y_train, y_test
```

Рисунок 2.9 – Підготовка даних для LSTM

Використовується проста нейронна мережа LSTM з одним шаром, який складається з 20 нейронів, коефіцієнт відсіву 0,25 і щільний шар з однією активаційною функцією. Крім цього, використовується середня абсолютна помилка – MAE як функція втрат і оптимізатор Адама (рис. 2.10).

Нейронна мережа тренується протягом 50 епох з розміром партії 4.

```
def build_lstm_model(input_data, output_size, neurons=20, activ_func='linear',
                     dropout=0.25, loss='mae', optimizer='adam'):
    model = Sequential()
    model.add(LSTM(neurons, input_shape=(input_data.shape[1], input_data.shape[2])))
    model.add(Dropout(dropout))
    model.add(Dense(units=output_size))
    model.add(Activation(activ_func))
    model.compile(loss=loss, optimizer=optimizer)
    return model
model = build_lstm_model(
    X_train, output_size=1, neurons=lstm_neurons, dropout=dropout, loss=loss,
    optimizer=optimizer)
history = model.fit(
    X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1, shuffle=True)
```

Рисунок 2.10 – Побудова та навчання LSTM

Для навчання необхідний масив навчальних прикладів. Кількість прикладів повинна бути досить велика (в 10-15 разів більше кількості нейронів мережі). Приклади надаються НМ, при цьому ваги зв’язків всередині неї поступово змінюються, щоб реальний вихідний сигнал був якомога ближче до очікуваного значення вихідного фактору. Один цикл надання всіх навчальних прикладів називається епохою. Зазвичай потрібно декілька тисяч ітерацій, щоб навчити НМ. Частина прикладів не бере участі у навчанні, а виділяється в тестову множину. На кожній епохі робота мережі перевіряється на тестовій множині. Таким чином тренується можливість НМ до узагальнення: можливості розповсюдити виявлену закономірність до даних, що не приймали участі в навчанні. Навчання НМ закінчується тоді, коли досягнуте задане середнє або мінімальне значення помилки, коли мережа вичерпала можливості для навчання або коли пройдено деяка визначена кількість епох. Після цього ваги зв’язків фіксуються, і мережа може використовуватися. Тепер, якщо в якості вхідних сигналів мережі вказати параметри вхідної множини, значення на виході буде надавати його прогноз, розрахований на основі виявленої закономірності.

НМ накопичує досвід на основі перегляду набору аналогічних прикладів, і фіксує його у вигляді набору ваг зв’язків. Не завжди НМ досягає гарних результатів навчання і узагальнення.

Серед можливих причин можна виділити наступні:

– Невдало підібрана архітектура мережі (надто багато або надто мало нейронів в прихованих шарах);

– Недостатньо прикладів для навчання;

– Фактори що впливають виділені невдало; в число вхідних параметрів не врахований один або декілька факторів, який найбільш впливає на вихідне значення показників;

– Шукана залежність не існує; навчальні приклади є унікальними, аналогія між ними відсутня.

Висновки до розділу 2

В результаті даної роботи були розглянуті варіанти побудови архітектури нейронних мереж для прогнозування часових рядів. Обрана найбільш підходяща мережа та реалізована за допомогою мови програмування Python.

РОЗДІЛ ІІІ. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ НЕЙРОМЕРЕЖНОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

3.1 Результати навчання нейронної мережі

Хоча нейромережні моделі є досить ефективними в задач оцінки, пошук відповідних параметрів для проведення навчання може стати досить трудомісткою задачею. Виходячи з популярних методів вибору кількості нейронів в прихованих шарах, визначимо нейромережні моделі, які будуть використовуватися в ході серії експериментальних навчань:

1. Модель з одним прихованим шаром та 20 нейронами в шарі (1-20). Правило використане при моделюванні: Кількість прихований нейронів повинна бути рівна числу в діапазоні від розміру вхідного шару до розміру вихідного.

2. Модель з одним прихованим шаром та 40 нейронами в шарі (1-40). Правило використане при моделюванні: Кількість прихований нейронів повинно бути рівним $^{2}/_{3}$ розміру вхідного шару плюс розмір вихідного шару.

3. Модель з одним прихованим шаром та 80 нейронами в шарі (1-80). Правило використане при моделюванні: Кількість прихований нейронів повинно бути меншим, ніж подвоєна кількість елементів у вхідному шарі.

Кожна модель буде навчена за допомогою 50 епох і навчальній вибірці складеної з значень періоду з 31 грудня 2004 року по 29 грудня 2017 року. Масив вхідних даних використаний для навчання нейронної мережі містить в собі 3274 значень (по одному на кожен день) в зазначеному діапазоні. Процес навчання моделі нейронної мережі з одним прихованим шаром і 20 нейронами зображено на (рис. 3.1). Як тільки НМ перегляне всі 3274 значення, вона знову повернеться до першого, або зупинить свою роботу після досягнення мінімальної помилки прогнозування MAE.

```
Epoch 15/50
2916/2916 [==============================] - 20s 7ms/step - loss: 0.0203
Epoch 16/50
2916/2916 [==============================] - 21s 7ms/step - loss: 0.0204
Epoch 17/50
2916/2916 [==============================] - 20s 7ms/step - loss: 0.0204
Epoch 18/50
2916/2916 [==============================] - 21s 7ms/step - loss: 0.0201
Epoch 19/50
2916/2916 [==============================] - 22s 8ms/step - loss: 0.0199
Epoch 20/50
 328/2916 [==>...........................] - ETA: 24s - loss: 0.0192
```

Рисунок 3.1 – Навчання моделі 1-20

Результати навчання різних моделей представлені в таблиці 3.1.

Помилка MAE	Після 1 епохи	Після 10 епох	Після 30 епох	Після 50 епох
1-20	0,0377	0,0212	0,0202	0,0198
1-40	0,0335	0,0193	0,0187	0,0184
1-80	0,0300	0,0189	0,0181	0,0174

Таблиця 3.1 – Результати навчання моделей (1-20), (1-40), (1-80)

Порівнюючи результати навчання різних моделей, можна зробити висновок, що різниця в помилці прогнозування майже не відрізняється для моделей. Тим не менш, якщо ціллю прогнозування являються фінансові дані або дані бізнес-процесів, то важливою може стати будь-яка різниця в значенні помилки прогнозування, оскільки більш точні дані будуть більш ефективні в процесі прийняття рішень.

Тривалість навчання моделі 1-80 проходило приблизно в 7 разів довше, ніж моделі 1-40 та 1-20, оскільки модель з 80 прихованими нейронами має набагато більшу кількість зв’язків. При цьому дана модель стала прогнозувати найточніші результати в порівнянні з іншими моделями приблизно з 30-ої епохи. Не дивлячись на велику кількість прихованих нейронів і зв’язків,

мережа не буле «недовченою» і довела справедливість побудови нейромережних моделей.

В процесі навчання моделей (1-20), (1-40) і (1-80) протягом 50 епох, кожному прихованому нейрону була надана своя вага. Використаємо результати цього навчання, щоб спрогнозувати значення для навчальної вибірки множини на період з 29 листопада 2017 року по 29 грудня 2017 року. Результати прогнозування представлені в таблиці 3.2.

Дата	(1-20) Факт.	(1-20) Прогноз.	(1-40) Факт.	(1-40) Прогноз.	(1-80) Факт.	(1-80) Прогноз.
2017-12-15	173.970	173.068	173.970	172.375	173.970	172.783
2017-12-18	176.419	174.271	176.419	173.770	176.419	173.948
2017-12-19	174.539	177.037	174.539	176.643	174.539	177.080
2017-12-20	174.350	174.822	174.350	174.614	174.350	175.020
2017-12-21	175.009	174.800	175.009	174.475	175.009	174.746
2017-12-22	175.009	175.787	175.009	175.272	175.009	175.535
2017-12-26	170.570	175.368	170.570	174.695	170.570	170.570
2017-12-27	170.600	170.039	170.600	170.589	170.600	170.696
2017-12-28	171.080	170.525	171.080	170.077	171.080	170.119
2017-12-29	169.229	171.424	169.229	170.745	169.229	171.000

Таблиця 3.2 – Прогнозування після 50 епох навчання моделі 1-20

Як видно з таблиці 3.2, прогнозовані значення значно наближені до існуючих. Слід відмітити, що контрольні значення не приймали участі в навчальній вибірці, а це означає, що дана НМ проходить перевірку на якість і може бути використана в процесах прийняття рішень.

Чим далі горизонт прогнозування від своєї навчальної вибірки, тем більше зростає значення помилки прогнозування. Для отримання більш точних прогнозів необхідно перенавчити НМ враховуючи отримані прогнозовані значення чи добавивши значення контрольної вибірки до значень навчальної. Також для побудови прогнозу на наступні часові періоди необхідно змінити горизонт прогнозування.

3.2 Результати прогнозування майбутніх значень часових рядів

Використовуючи результати навчання моделей 1-20 (рис. 3.2) і 1-80 (рис. 3.3) для прогнозування майбутніх значень часового ряду на основі набору тестів, отримуємо графіки.

Рисунок 3.2 – Прогнозування моделі 1-20

Рисунок 3.3 – Прогнозування моделі 1-80

З даного графіку складно що сказати про результати прогнозування значень часового ряду, тому розглянемо останні 30 днів з графіку моделі 1-20 (рис. 3.4) і моделі 1-80 (рис. 3.5).

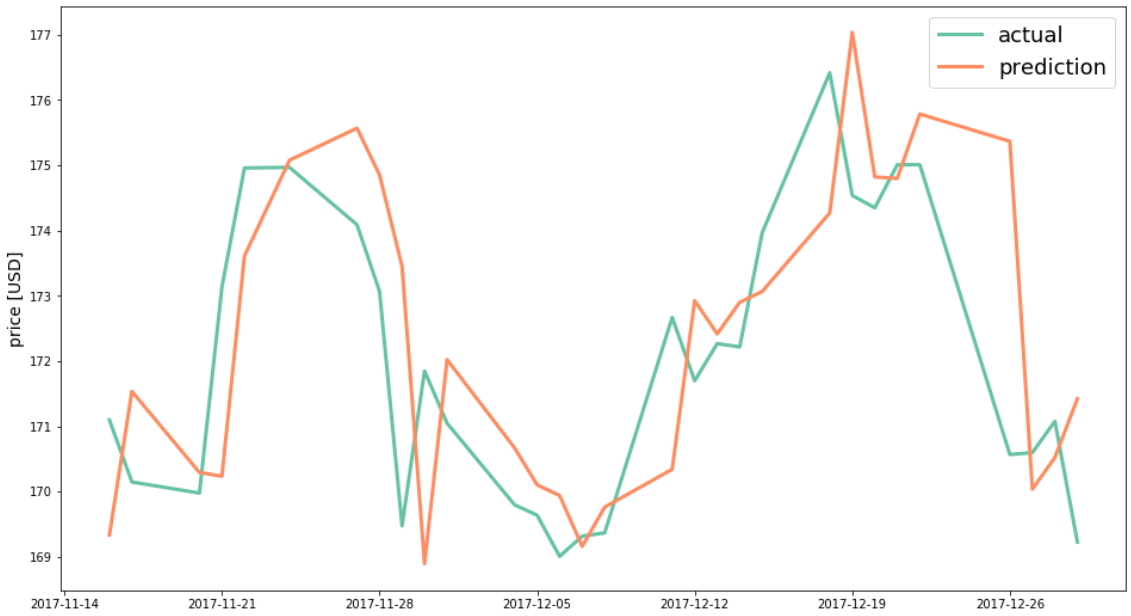


Рисунок 3.4 – Графік останніх 30-и днів прогнозу моделі 1-20

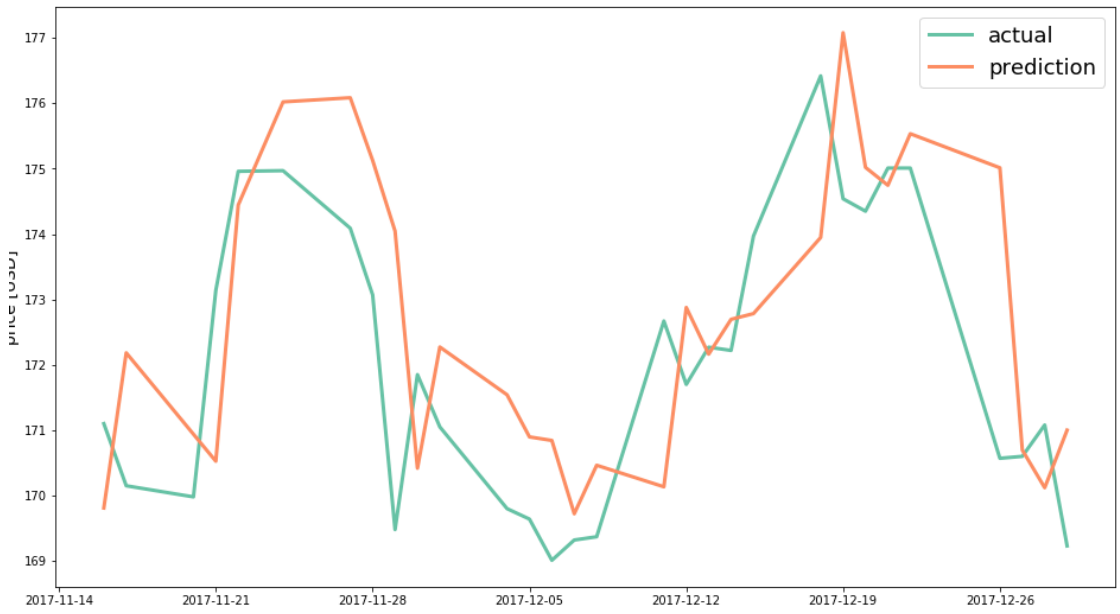


Рисунок 3.5 – Графік останніх 30-и днів прогнозу моделі 1-80

Фундаментальний недолік цих моделей закладається в тому, що для прогнозування конкретного дня вона в основному використовує значення попереднього дня. Лінія прогнозу здається не більш ніж зрушена версія

реальної ціни. Фактично, якщо ми скорегуємо прогнози та зрушимо їх на один день, це спостереження стає ще більш очевидним, модель 1-20 (рис. 3.6) і модель 1-80 (рис. 3.7).

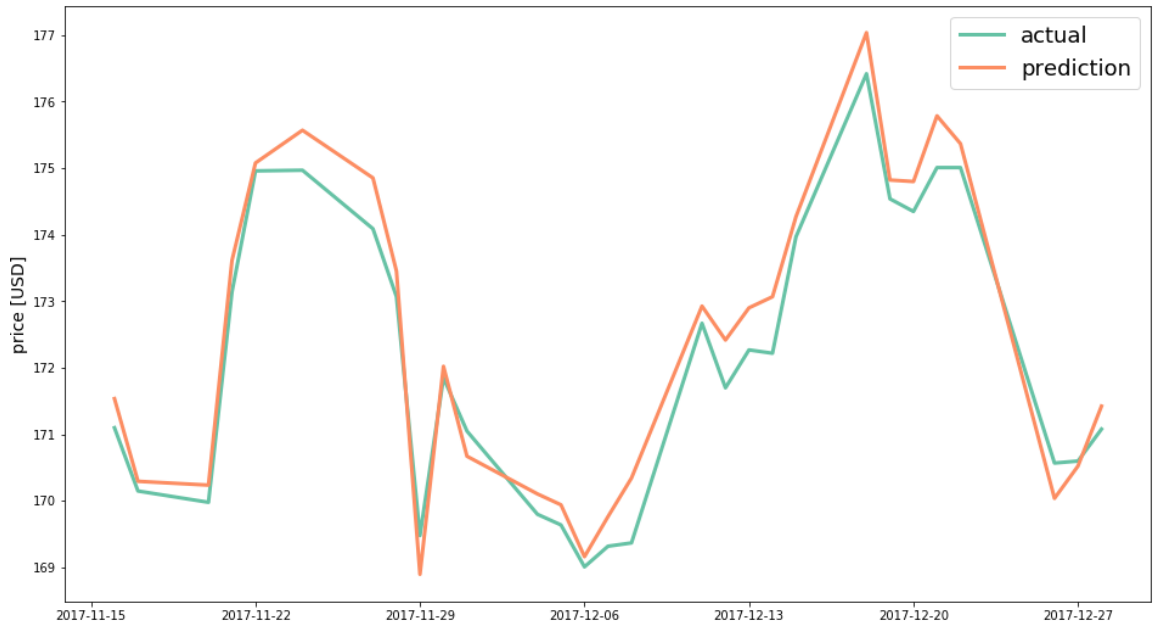


Рисунок 3.6 – Скорегований прогноз моделі 1-20

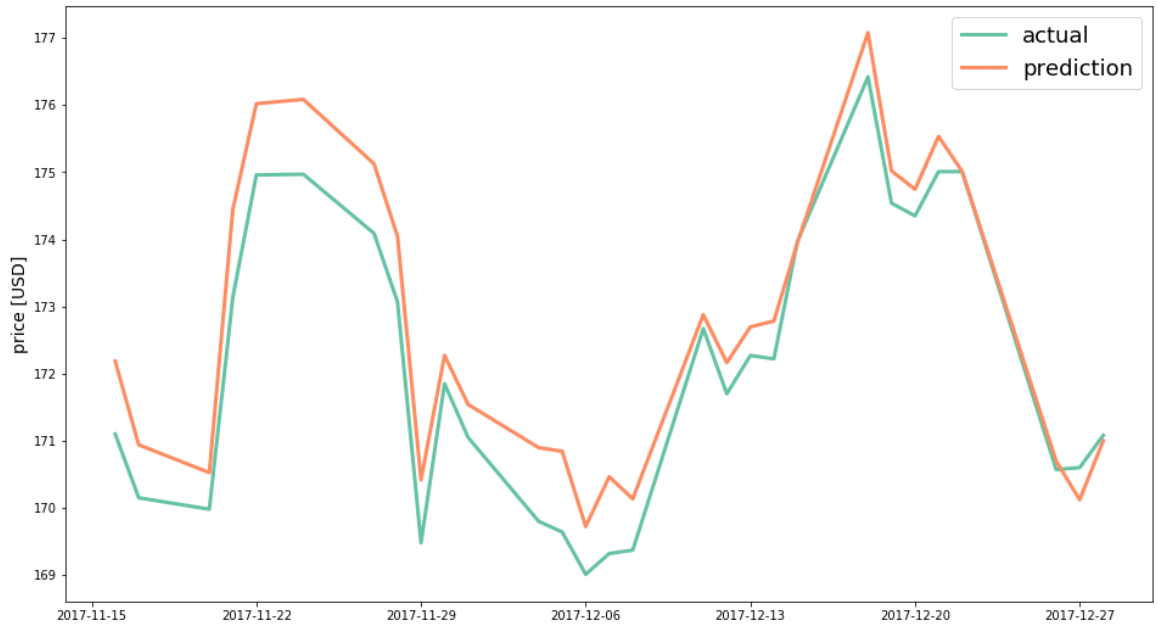


Рисунок 3.7 – Скорегований прогноз моделі 1-80

З цього видно, що неочікувано спостерігається майже ідеальна відповідність між реальними даними та прогнозованими даними, що вказує на те, що моделі по суті вивчають ціну в попередній день.

Щоб пояснити даний факт, обчислимо очікувані прибутки, які було спрогнозовано моделями та порівняємо їх з реальними прибутками, модель 1-20 (рис. 3.8) і модель 1-80 (рис. 3.9).

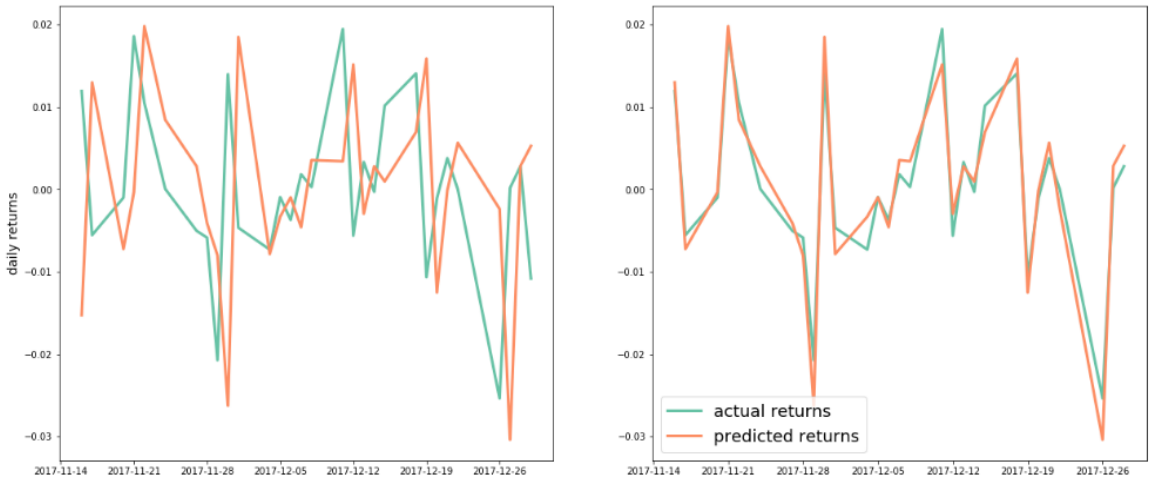


Рисунок 3.8 – Реальні та прогнозовані прибутки моделі 1-20

Рисунок 3.9 – Реальні та прогнозовані прибутки моделі 1-80

Дивлячись на реальні та прогнозовані прибутки, як в їх початковому вигляді, так і для корегувань на день, можна зробити наступне припущення: як видно з графіків наведених вище, реальна прибутковість і прогнозована прибутковість не пов’язані між собою. Тільки після застосування 1-денної зміни до прогнозованих даних, отримуємо висококорельовані результати моделі 1-20 (рис. 3.10) і моделі 1-80 (рис. 3.11), які нагадують результати реальних цін.

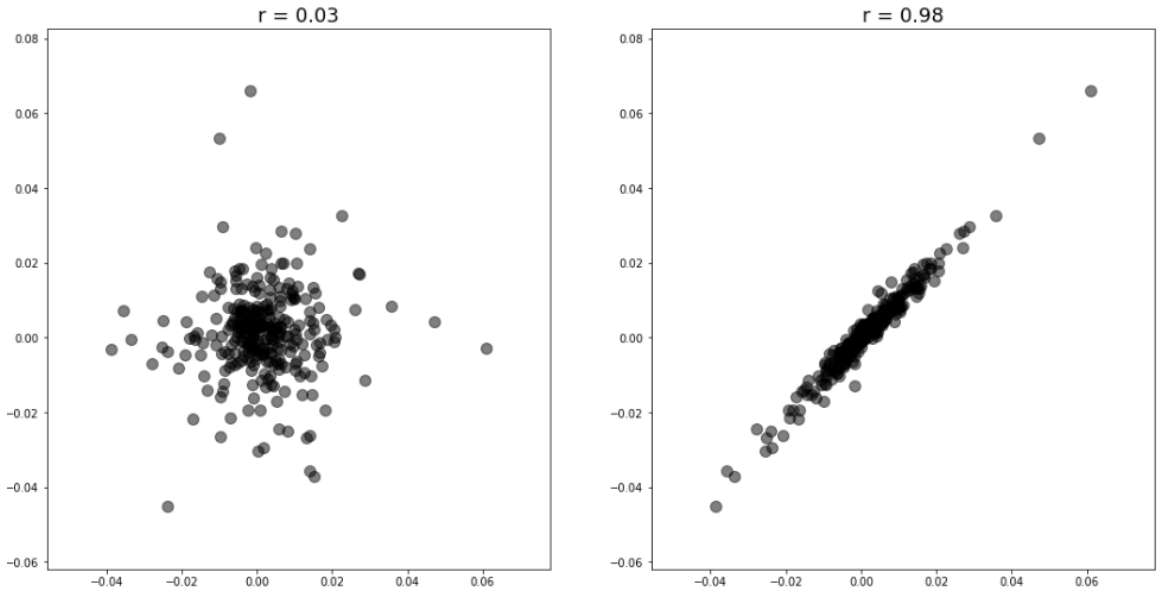


Рисунок 3.10 – Актуальна та здвигнута кореляція результатів прогнозування моделі 1-20

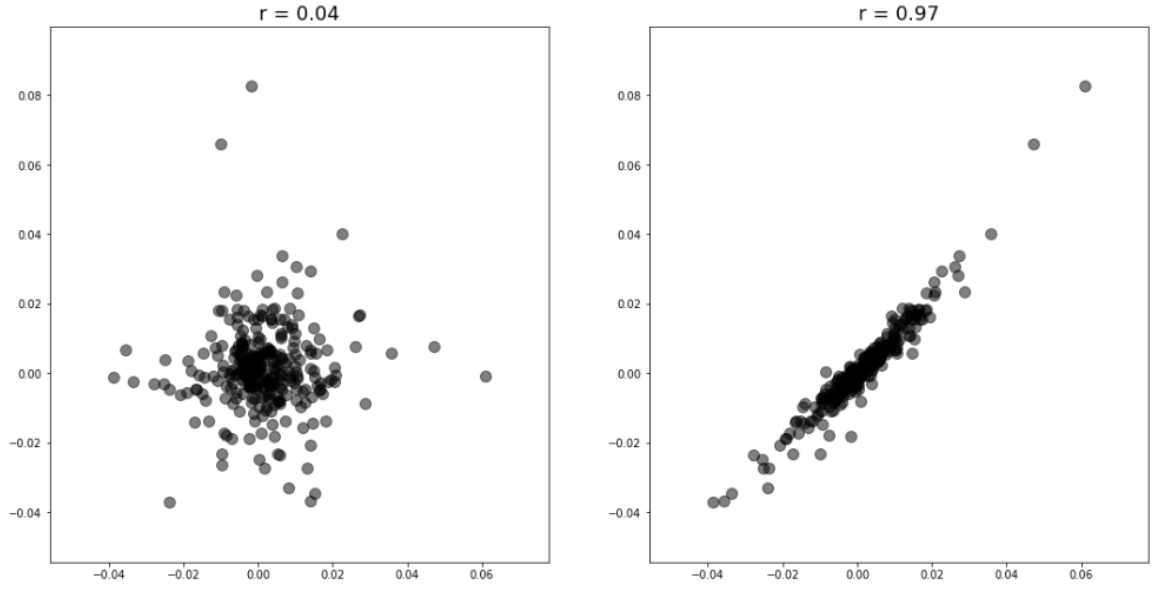


Рисунок 3.11 – Актуальна та здвигнута кореляція результатів прогнозування моделі 1-80

Висновки до розділу 3

Ціль цього дослідження полягала у тому, щоб розглянути множину прикладів прогнозувань цін на фондовому ринку з використанням глибоких нейронних мереж, які використовують підхід аналогічний тому, який використовувався в даній роботі: реалізація нейронної мережі LSTM з

використанням історичних даних про ціни для прогнозування майбутніх результатів. Показано чому дані моделі не завжди придатні для реальної торгівлі.

Мережа спроможна ефективно навчатися. Але в результаті використовується стратегія, в якій прогнозування значення близького до попереднього, виявляється успішним з точки зору мінімізації середньої абсолютної помилки.

Однак незалежно від того, наскільки точні результати прогнозування з точки зору помилки втрат – на практиці результати точкового прогнозування, які заснованих тільки на історичних даних про ціни, як показано в роботі, залишаються важко досяжними та не особливо корисні для торгівлі.

Потенційно існують більш складні підходи для впровадження нейронних мереж LSTM для прогнозування цін. Використання великої кількості даних, а також оптимізація мережної архітектури та параметрів. Однак є більше можливостей для включення даних і функцій, які виходять за рамки тільки історичних цін. Можна сказати, що минулі результати не являються показником майбутніх результатів.

ВИСНОВКИ

Нейронні мережі отримали найбільше розповсюдження в області прогнозування динамічних систем, вони успішно застосовуються для рішення цілого ряду задач інтелектуального аналізу даних. Разом з тим, для багатьох галузей вивчення можливостей використання нейронних мереж знаходиться в експериментальній стадії нейромережні технології не повинні розглядатися як універсальний засіб вирішення подібних задач. Їх використання виправдано в тих областях, в яких існує значна кількість однотипних прикладів, що відображають приховані взаємозв'язки.

Для застосування технології глибокого навчання для вирішення практичних задач пов'язаних з аналізом і прогнозуванням часових рядів, в даній роботі було вирішено наступні задачі:

1. Проведено аналіз предметної області в області глибокого навчання, прогнозування часових рядів;

2. Проаналізовано сутність комплексу задач з прогнозування часових рядів і виділення їх компонент;

3. Побудовано нейромережну модель LSTM;

4. Охарактеризувано вхідні, постійні, проміжні та вихідні дані нейромережної моделі LSTM;

5. Реалізовано обраний варіант нейромережної моделі LSTM;

6. Проведено навчання та тестування нейромережної моделі LSTM;

7. Досліджено результати прогнозування майбутніх значень часових рядів за допомогою нейромережної моделі LSTM.

Використання різноманітної архітектури мереж, випадковість вибору початкових синоптичних коефіцієнтів і використання інших мережний параметрів привели до того, що прогнози різних нейромежер навчених однаково на одних і тих прикладах відрізнялися. Середня абсолютна помилка самої ефективної моделі після 50 епох навчання становить 0,00893, що є

задовільним результатом для моделей подібного роду. Виходячи з отриманих прогнозів і оцінки побудованої моделі, можна зробити висновок, що штучні нейронні мережі являються хорошим інструментом для прогнозування часових рядів.

Таким чином, можна стверджувати, що була досягнута ціль і успішно вирішені всі завдання роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Forecasting: principles and practice [Электронный ресурс]: / Электрон. дан. - Neural network models - https://www.otexts.org/fpp/9/3 (дата обращения 04.04.2016).

2. Leondes T. Neural Network Systems Techniques and Applications Vol. 7 / C.T. Leondes – San Diego: Academic Press, 2010 – 438 p.

3. Анализ временных рядов [Электронный ресурс]: / Электрон. дан. - Идентификация модели временных рядов - http://www.statsoft.ru/home/textbook/modules/sttimser.html (дата обращения 26.04.2016).

4. Архитектура нейронной сети / Сайт «POZNAYKA» [Электронный ресурс]. / URL: http://poznayka.org/s78608t1.html

5. Архитектура нейронной сети / Сайт «ИНТУИТ» [Электронный ресурс]. / URL: https://www.intuit.ru/studies/courses/6/6/lecture/178

6. Архитектура нейронной сети. / Н. П. Посмаков [Электронный ресурс]. / URL: http://www.tpinauka.ru/2017/06/Posmakov.pdf

7. Барсегян А. Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод – СПб.: БХВ-Петербург., 2007. – 384 с.

8. Введение в нейронные сети / Сайт «Apsheronsk.bozo.ru» [Электронный ресурс]. / URL: http://apsheronsk.bozo.ru/Neural/Lec1.html

9. Глубокое обучение погружение в мир нейронных сетей. / С. Николенко, А. Кадурин, Е. Архангельская / СПБ, 2018. 480с.

10. Интеллектуальные информационные системы и технологии: учебное пособие / Ю.Ю. Громов, О.Г. Иванова, В.В. Алексеев [и др.] / Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2013. 244 с.

11. Искусственные нейронные сети. Теория и практика. / В. В. Борисов, В. В. Круглов / Горячая линия-Телеком, 2002. - 382 с.

12. Как обучаются нейронные сети. / Д. Е. Хинтон. / В мире науки - 1992-N11-N12-с.103-107.

13. Логическое программирование: учебное пособие. / О. П. Солдатова, И. В. Лёзина / Самара: СНЦ РАН, 2010. 81 с., ил.

14. Математическое моделирование и анализ нелинейных систем. / Ю. А. Бычков, Ю.М. Соловьева, Е. Б. Щербаков [и др.] / Изд-во СПБГЭТУ «ЛЭТИ», 2015. 354 с.

15. Машинное обучение / Сайт «Habr» [Электронный ресурс]. / URL: https://habr.com/company/wunderfund/blog/331310

16. Методы оптимизации: курс лекций / А. С. Михайлов / Красноярск: СибГТУ, 2012. – 81с.

17. Нейронные сети. / Я. Еремин / [Электронный ресурс]. / URL: http://elanina.narod.ru/lanina/ind/neiro/index.html

18. Основы ИНС / Сайт «NEURALNET» [Электронный ресурс]. / URL: https://neuralnet.info/book

19. Подкорытова О. Анализ временных рядов: Учебное пособие / О.А. Подкорытова, М.В. Соколов – М.: Юрайт, 2016 – 266 с.

20. Прогнозирование с помощью нейронных сетей / М. В. Ульянова [Электронный ресурс]. / URL: http://www.scienceforum.ru/2018/pdf/5783.pdf

21. Садовникова Н. Анализ временных рядов и прогнозирование: Учебное пособие / Н.А. Садовникова, Р.А. Шмойлова – М.: Синергия, 2016 – 152 с.

22. Суслов В.И. Эконометрия. Часть III Эконометрия – I: Анализ временных рядов: Учебное пособие / В.И. Суслов, Н.М. Ибрагимов, Л.П. Талышева, А.А. Цыплаков – Новосибирск: СО РАН, 2005. – 744 с.

23. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен – М.: Вильямс, 2015. – 1312 с.

24. Файншмидт В. Дифференциальное и интегральное исчисление функций одного аргумента: Учебное пособие / В. Файншмидт – СПб.: БХВ Петербург, 2006 – 224 с.

25. Хайкин С. Нейронные сети. Полный курс / С. Хайкин – М.: Вильямс, 2016. – 1104 с.

26. Ярушкина Н. Интеллектуальный анализ временных рядов: Учебное пособие / Н.Г. Ярушкина, Т.В. Афанасьева, И.Г. Перфильева – М.: Инфра-М, 2015 – 160 с.

27. Deng, L.; Yu, D. (2014). “Deep Learning: Methods and Applications” (PDF). Foundations and Trends in Signal Processing. 7 (3—4): 1—199.

28. Bengio, Yoshua (2009). “Learning Deep Architectures for AI” (PDF). Foundations and Trends in Machine Learning. 2 (1): 1—127.

29. Schmidhuber, J. (2015). “Deep Learning in Neural Networks: An Overview”. Neural Networks. 61: 85—117.

30. Glauner, P. (2015). Deep Convolutional Neural Networks for Smile Recognition (MSc Thesis). Imperial College London, Department of Computing.

31. Song, Lee, Neural Information Processing, 2013.

32. Olshausen, B. A. (1996). “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. Nature. 381 (6583): 607—609.