Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “Передбачення руху цін акцій за допомогою рекурентних нейронних мереж”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - 23.00.00.000

Студент

ІПЗ-42 _____ /Андрій УСТІНСЬКИЙ/

Науковий керівник

к.т.н., ас._____/Максим ТКАЧЕНКО/

Консультант з питань нормоконтролю

_____/Тамара ЧАПОВСЬКА/

Допускається до захисту

Завідувач кафедри

д.т.н., проф. _____ /Олексій БИЧКОВ/

Київський національний університет імені Тараса Шевченка

Київ - 2021

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО

Зав. кафедри _програмних систем і технологій__

_____/Олексій БИЧКОВ/

(підпис) (прізвище та ініціали)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Устінському Андрію Андрійовичу

(прізвище, ім’я, по-батькові)

1. Тема бакалаврської роботи “Передбачення руху цін акцій за допомогою рекурентних нейронних мереж**”**

Керівник проекту (роботи) Ткаченко Максим Васильович, к.т.н., асистент_____ _____ затверджені наказом вищого навчального закладу від **“** _ ” _____2021 р. № __ _ ___

2. Строк подання студентом роботи 1 червня 2021 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій при розробці моделей машинного навчання

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз фондового ринку

2. Аналіз моделей машинного навчання для фондового ринку.

3. Вивчення роботи рекурентних нейронних мереж.

4. Розробка алгоритмічної моделі та програмної реалізації різних видів РНМ.

5. Порівняння методів аналізу цін акції та визначення самих ефективних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1 “Фондовий ринок та аналіз цін”	Ткаченко М.В.	22.01.2021	22.01.2021
Розділ 2 “Штучний інтелект і рекордна прибутковість інвестицій. РНМ”	Ткаченко М.В.	22.01.2021	22.01.2021
Розділ 3 “Моделі рекурентних нейронних мереж”	Ткаченко М.В.	22.01.2021	22.01.2021
Розділ 4 “Програмна реалізація”	Ткаченко М.В.	22.01.2021	22.01.2021

7. **Дата видачі завдання** 14 жовтня 2020 р.

Керівник _____ /Максим ТКАЧЕНКО/

Завдання прийняв до виконання _____ /Андрій УСТІНСЬКИЙ/

КАЛЕНДАРНИЙ ПЛАН

№ п/п	**Назви етапів бакалаврської роботи**	**Строк виконання етапів роботи**	**Примітка**
1	Уточнення постановки задачі	15.10.20 – 12.11.20	Виконано
2	Підбір і вивчення літератури	13.11.20 – 15.12.20	Виконано
3	Аналіз існуючих методів, концепцій, моделей та алгоритмічні вирішення задачі	16.12.20 – 12.01.21	Виконано
4	Розробка алгоритмічної моделі	13.01.21 – 22.02.21	Виконано
5	Програмна реалізація алгоритму	23.02.21 – 29.03.21	Виконано

Студент – бакалавр _____/Андрій УСТІНСЬКИЙ/

Керівник роботи _____/Максим ТКАЧЕНКО/

АНОТАЦІЯ

Дипломна робота: 65 с., 19 рис, 10 джерел, 1 додаток (15 с., 3 діаграми, 1 табл.).

Тема: Передбачення руху цін акцій за допомогою рекурентних нейронних мереж.

Об’єкт дослідження: біржові методології та технології штучного інтелекту.

Мета роботи: розробка AI моделі для аналізу та передбачення руху ціни на ринку.

Предмет дослідження: досліджуються методи передбачення на основі РНМ (Рекурентних нейронних мереж).

Результати дослідження:

Реалістичний прогноз поведінки акцій на основі використання лінійної алгебри та нейронних мереж.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RNN/ PHM - Recurrent Neural Networks/Рекурентна Нейронна Мережа

PSO-LS-SVM - particle swarm optimization with support vector machine(оптимізація рою частинок з підтримкою векторної машини)

LSSVM - support vector machine (Метод опорних векторів)

NN-BP - neural network with Back-Propagation Algorithm(нейронна мережа з алгоритмом зворотного поширення)

BRNN - bidirectional recurrent neural network (двонаправлена рекурентна нейронна мережа)

LSTM - long short-term memory (тривала короткочасна пам’ять)

GRU - Gated recurrent units (Вентильні рекурентні вузлии)

RSI - Relative strength index (Індекс відносної сили)

MACD - Moving Average Convergence/Divergence (сходження / розбіжність ковзних середніх)

CNN - convolutional neural network (Згорткова нейронна мережа)

НЛП - Нейролінгвістичне програмування

RMLP - Recurrent Multi-Layer Perceptron (Багатошаровий перцептрон, що повторюється)

ВСТУП

Актуальність роботи

На даний час на біржах торгують мільйони трейдерів, які кожен день створюють рівновагу на ринку. Всі вони повністю різні, у кожного своя стратегія і вид робити на біржі, але всіх їх об’єднює одна дуже важлива річ : отримувати завжди якомога більше прибутку з ринку. Один з варіантів, який може їм допомогти в цей час, це аналіз активів за допомогою штучного інтелекту.

Тому, від початку прогресу в розробці різних видів штучного інтелекту та нейронних мереж, велика кількість світових брокерських компаній, почали власні досліди в спробі, пристосувати алгоритми штучного інтелекту до бірж та надалі продовжують їх вдосконалювати.

Так як, алгоритмів штучного інтелекту, існує велика кількість, актуальністю цієї робити, є самостійно розробити всі можливо робочі варіанти рекурентної нейронної мережі, так як цей вид нейронних мереж найбільш підходящий до поставленої задачі та отримати най працездатні алгоритми.

Порівняння роботи з відомими розв’язаннями проблеми

Відомі алгоритми такі як : PSO-LS-SVM(particle swarm optimization with support vector machine), LSSVM(support vector machine) та NN-BP(neural network with Back-Propagation Algorithm), часто використовують при аналізі руху цін на біржі. Але для роботи з їх допомогою, створюють більш адаптовані версії до бірж, так як вони не повністю ідеально пристосовуються для передбачення руху ціни.

Більш підходящі алгоритми, для передбачення руху цін є алгоритми побудовані на основі РНМ (рекурентні нейронні мережі), вони краще працюють з ціновими графіками, так як вони не побудовані на регресійному аналізі чи звичайних нейронних мережах, а на нейронні мережі, які мають пам'ять і які при роботі з новими даними, ще й перепровіряють вже вивчені дані. Важливим є розв’язок проблеми, визначення найбільш підходящої моделі.

Мета і задачі дослідження

Метою дипломної роботи є розробка методу передбачення руху ціни на біржі, за допомогою історичних даних та знань розробки нейронних мереж отриманих під час виконання даної роботи.

Досліджуванні моделі повинна опрацювати та провести аналіз будь-якого активу, акції, індекса, валюти, тощо.

Визначення самого оптимального алгоритму, який найкраще може справитися з поставленою задачею – передбачення руху цін акцій.

Досягнення мети включає розв’язання таких **задач**:

1) огляд існуючих концепцій лінійної алгебри;
2) аналіз існуючих алгоритмів аналізу даних та передбачення.
3) вибір оптимального алгоритму та обгрунтування доцільності його використання;
4) реалізація алгоритмів РНМ та їх адаптація до роботи з біржою.
5) Порівняти між собою та з власною адаптацією алгоритму LSTM.

Предметом дослідження є біржеві ринки, розробка штучного інтелекту, алгоритми рекурентних нейронних мереж.

Методи дослідження

Головним дослідження для входження в сферу фінансів було, вивчення базових концепцій поведінки ціни. Вивчення фундаментального зв’язку з ціною, такий як вплив світових випадків на поведінку трейдерів та фондів. Вивчення технічної частини ринку : ситуації, паттерни, об’єми, тощо.

Наступний крок, це вивчення базових концепцій побудови штучного інтелекту: ймовірності, регресії, класифікації, ANN, RNN, тощо.

Практичне значення одержаних результатів

Одержана модель на основі Рекурентних нейронних мереж, адаптованих до роботи з біржами, яка дозволяє проводити аналіз будь-якого активу та отримати графік та передбачення його майбутньої ціни.

Особистий внесок студента

Основним результатом є: Модель, яка буде проводити передбачення руху ціни обраної акції, на основі найважливіших алгоритмів РНМ та особистої модифікації одного із алгоритмів та получати результат, для проведення інвестиційних рішень на основі нього.

РОЗДІЛ 1

ФОНДОВИЙ РИНОК ТА АНАЛІЗ ЦІН.

Фондовий ринок є публічним ринком, який існує для випуску, купівлі та продажу акцій, які торгуються на біржі або на інших площадках. Акції, представляють собою часткову власність (частинку) в компанії, а фондовий ринок - це площадка, де інвестори можуть купувати і продавати право на власність таких активів. Ефективне функціонування фондового ринку є дуже важливим процесом для економічного розвитку, оскільки він дає підприємству можливість швидкого доступу до капіталу для подальшого розвитку та працездатності.

1.1. Цілі фондового ринку - капітал та інвестиційний дохід

Фондовий ринок сприяє двом дуже важливим аспектам. Перший полягає в наданні коштів компаніям для фінансування та розширення свого бізнесу. Якщо компанія випускає один мільйон акцій, по стартовій ціні - 10 умовних одиниць за акцію, це дає компанії капітал в 10 мільйонів умовних одиниць, який вона може використовувати для розвитку свого бізнесу. Звичайно потрібно враховувати різні збори та плата інвестиційному банку за управління акціями. Така модель уникає будь-які борги, які можуть призвести до легкого підривання довіри заінтересованих лиць та відсотків при віддачі боргу.

Друга мета цієї моделі в фондовому ринку, є можливість для інвесторів прийняття участі у прибутках компаній в які вони інвестують. Інвестори можуть отримати прибуток від покупки акцій певної компанії одним з двох способів:

- Деякі акції виплачують регулярні дивіденди (певна сума грошей на акцію, якою хтось володіє).

- Продажа акцій з метою отримання прибутку, якщо ціна акцій збільшується від їх ціни покупки.

Наприклад, якщо інвестор купив акцію компанії за ціною 55 умовних одиниць за акцію, а ціна акції згодом підвищується до 83 умовних одиниць за акцію, інвестор може отримати 50% прибутку від початкової інвестиції, продавши свою акцію. Або він може зафіксувати певну частину інвестиції для подальшої утримання її.

1.2. Аналіз акцій - ринкова капіталізація, прибуток на акцію і фінансові коефіцієнти

Аналітики та інвестори які працюють у фондовому ринку можуть аналізувати безліч факторів, щоб вказати ймовірне майбутній напрямок ціни акцій. Деякі найбільш часто враховані змінні для аналізу запасів :

- Ринкова капіталізація - це загальна вартість всіх, в обігу акцій. Чим більш висока ринкова капіталізація, тим зазвичай вона вказує на те, що компанія стійкіша і фінансово стійка.
- Біржові регулюючі органи зобов'язані регулярно надавати звіти про доходи. Аналітики ретельно стежать за цими звітами, що публікуються щоквартально і щорічно, так як це є хорошим показником того, наскільки успішним є бізнес. Серед ключових факторів, проаналізованих в звітах про прибутки і збитки - EPS (прибуток компанії на акцію), який відображає прибуток компанії, поділену на все її акції в зверненні.
- P/E (Співвідношення ціни до прибутку): відношення ціни акцій компанії до її EPS. Більш високе відношення цього індекса вказує на те, що інвестори готові платити більш високі ціни за акції компанії, тому що передбачається ріст ціни більший за ціну покупки.
- Співвідношення позикових і власних коштів - це фундаментальний показник фінансової стійкості компанії, оскільки він показує, який відсоток операцій компанії фінансується за рахунок боргу в порівнянні з тим, який відсоток фінансується інвесторами в акціонерний капітал.

- ROE - Коефіцієнт чистої рентабельності. Коефіцієнт рентабельності власного капіталу вважається дуже важливим показником потенціалу зростання компанії, оскільки він показує чистий дохід компанії по відношенню до загальних інвестицій в акціонерний капітал компанії.
- Інші інвестори зазвичай використовують фінансові коефіцієнти які включають прибутковість активів (ROA), дивідендну прибутковість, співвідношення ціни до балансу (P/B), коефіцієнт поточної ліквідності і коефіцієнт оборотності запасів.
- Маржа прибутку - є кілька коефіцієнтів рентабельності, які можуть розглянути інвестори, включаючи маржу операційного прибутку і маржу чистого прибутку. Перевага аналізу рентабельності замість абсолютного показника в вираженні умовних одиниць полягає в тому, що він показує процентну рентабельність компанії. Наприклад, компанія може показати прибуток в розмірі 4 мільйонів умовних одиниць США, але якщо це призведе тільки до 3% прибутку, то значить значно знижений рівень доходів може поставити під загрозу прибутковость компанії.

1.3. Два основних підходи до інвестування на фондовому ринку

Аналітики та інвестори використовують безліч методів відбору акцій, але практично всі вони представляють собою ту чи іншу форму двох основних стратегій торгівлі акціями: інвестування в вартість або інвестування в рух ціни, як вверх так і вниз.

Інвестори зазвичай інвестують в стабільні компанії, які демонструють прибутковість протягом тривалого періоду часу і можуть пропонувати регулярні дивідендні доходи. Інвестори, які ведуть зростання, шукають компанії з виключно високим потенціалом зростання з розрахунком в максимальні оцінці вартості акцій. Вони зазвичай менше піклуються про доходи від дивідендів і більш схильні

ризикувати, вкладаючи капітал у відносно молоді компанії. Інвестори в зростанні часто віддають перевагу акціям технологічних компаній через їх високий потенціал зростання.

1.4. Висновки до розділу

Інвестиції є дуже давнім і важливим процесом ще більш затребуваним в цей час, де все оцифровується. Інвестиції це то що надає інвестуючі компанії капітал для подальшого розвитку. Інвестиції бувають двох видів : в довгий строк, та короткий. Більшість трейдерів працюють в короткий строк, так як це дає більшу можливість для заробітку, але й більший ризик. Інвестори можуть інвестувати в ріст ціни та в її падіння. Інвестування в падіння – це коли інвестор бере в компанії “в кридит” 10 акцій по ціні 59 умовних одиниць та зразу їх продає, потім ціна падає наприклад до 45 умовних одиниць і він купує 10 акцій вже по меншій ціні, та ввідіє їх назад компанії а різницю залишає собі. Звичайно для таких маніпуляцій, інвестору потрібно бути впевненому в такому сценарії, для такого повинен був бути проведений аналіз активу. Аналіз активу проводиться здебільшого на основі рівнів підтримки та опори, об’ємім, різних індекаторів та повендінкового фактору. Ці всі методи використовується дуже багато років, і до сих пір є актуальними, але метою цієї роботи є добавлення до цих методів – аналіз на основі РНМ.

Інвестори зазвичай інвестують в стабільні компанії, які демонструють прибутковість протягом тривалого періоду часу і можуть пропонувати регулярні дивідендні доходи. Інвестори в зростанні часто віддають перевагу акціям технологічних компаній через їх високий потенціал зростання.

РОЗДІЛ 2
ШТУЧНИЙ ІНТЕЛЕКТ І РЕКОРДНА ПРИБУТКОВІСТЬ ІНВЕСТИЦІЙ. РНМ

2.1. Рекурентні нейронні мережі

Рекурентними називають штучні нейромережі, в яких, поряд із прямими зв'язками, направленими від рецепторів (входів) мережі до її ефекторів (виходів), є зворотні, що мають діють в протилежному напрямку. На відміну від нейронних мереж прямого поширення, які створюють проекцію статичного типу поданих на стимули (входи векторів даних) у реакції (вихідні вектори), нейронні мережі, що повторюються є динамічними системами, що оперують з послідовностями вхідних даних, перетворюючи їх на послідовності реакцій. Поведінковий фактор рекурентних нейромереж показує стереотипи які були набуті при навчанні, що робить їх близькими до цілеспрямованих адаптивних динамічних систем, настроєних на досягнення цілей визначених заздалегідь. Але на відміну від останніх, програмування поведінки РНМ здійснюється шляхом навчання на реальних прикладах або випадкаї, що не потребує формального визначення якихось цілей. Вони здатні діяти в умовах невизначеності ефективно, зокрема, вирішувати задачі адаптивного керування поведінкою складних систем у нестаціонарному оточенні, приймати оперативні рішення в системах ситуаційного управління тощо. За архітектурою та здатністю адаптуватись до умов оточення РНМ нагадують справжню-живу нервову систему. Тому їх дослідження має не тільки загальнонаукове, але і прикладне значення для усвідомлення факту адаптації в живій природі, створення іноваційниї методів лікування нервових та діагностики психічних захворювань, розкриття механізмів збереження інформації в пам'яті, правильної інтерпретації даних нейрофізіології які є відомі.

Ще в 60-ті роки минулого століття почалося вивчення нейромереж із зворотними зв’язками. В перших дослідженнях з моделювання нервової системи вивчались в основному статистичні властивості потоків нервової активності, тому часто залишались поза увагою функції зворотних зв’язків. Вперше рекурентну організацію було використано Д. Хопфілдом , яка надала властивостей асоціативної пам’яті створеній ним нейронні мережі. Термін “рекурентна нейронна мережа” почав поширюватися, коли було запропоновано методи навчання нейронних мереж із зворотними зв’язками (наприкінці 80-х років) та визначено основні проблеми, що вимагали практичного застосування ти вирішення на шляху до їх реалізації. Найбільші складності викликали незадовільна результативність навчального процесу та висока ресурсоємність, а теж відсутність гарантії стабільності поведінки рекурентних нейронних мереж. За минулі два десятиріччя у вирішенні цих проблем досягнуто суттєвого прогресу. Створено експериментальні моделі рекурентних нейронних систем для адаптивного керування складними транспортними та промисловими об’єктами. Розроблено ефективніші методи навчання та тестування РНМ, розвинуто загальну теорію динамічних рекурентних нейронних мереж, запропоновано нові архітектурні рішення, які дозволяють поєднувати рекурентні структури з відомими нейронними парадигмами і одержувати нейронні мережі з новими властивостями. Успішним прикладом застосування рекурентної нейронної мережі для вирішення легкої технічної задачі стала продемонстрована фірмою Bouing експериментальна нейронна система керування гіперзвуковим літаком зі змінною геометрією крила.

2.1.1. Загальні відомості про РНМ

« Рекурентні нейромережа, пам’ятає минуле і на її рішення впливає те, що вона дізналася з минулого».

Рекурентна нейронна мережа (RNN - recurrent neural networks) - це тип штучної нейромережі, яка використовує дані часових рядів або послідовні дані. Ці алгоритми глибокого навчання зазвичай використовуються для часових порядкових

проблем, таких як обробка природної мови (nlp), переклад мови, розпізнавання мови та субтитри до зображень; вони включені в такі популярні програми, як Google Translate, голосовий пошук та Siri. Різниця між рекурентними та звичайними НМ:

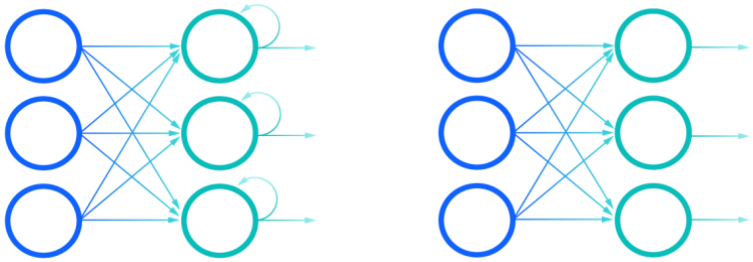


Рис. 2.1. Нейронні мережі, що повторюються та нейронні мережі прямого зв'язку

Як згорткові так і прямі нейромережі (CNN), періодичні нейромережі використовують дані для навчання (рис. 1). Вони відрізняються своєю “пам’яттю”, оскільки щоб впливати на поточний вхід і вихід, вони беруть інформацію з минулих входів. У той час як традиційні глибинні НМ можуть допустити, що виходи та входи не залежать один від одного, вихід рекурентних нейромереж залежить від попередніх елементів у даній послідовності. Хоча майбутні події також могли б допомогти у визначенні результату даної послідовності, однонаправлені РНМ не можуть враховувати у своїх прогнозах події майбутнього.

Іншою відмітною характеристикою періодичних періодичних мереж є те, що вони діляться на кожному рівні мережі своїми параметрами:

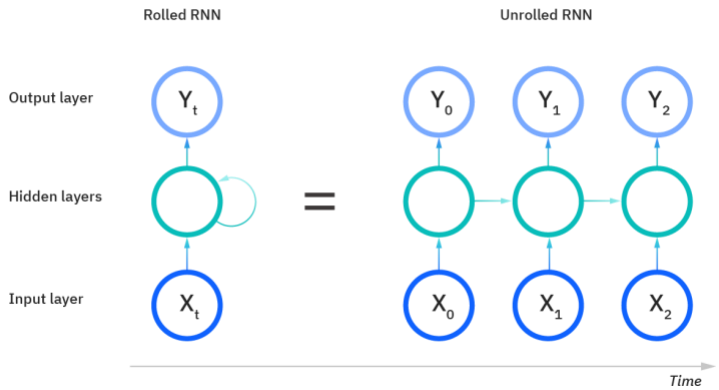


Рис. 2.2. Порівняння періодичних нейронних мереж (ліворуч) та зворотних нейронних мереж (праворуч)

Хоча мережі прямої передачі даних мають різну вагу на кожному вузлі, періодичні НМ мають однаковий вагомий параметр у кожному рівні мережі. Тим не менш, ці ваги все ще коригуються в процесі зворотного розповсюдження та градієнтного спуску для полегшення (підкріплення) навчання.

Рекурентні нейромережі, використовують алгоритм зворотного розповсюдження через BPTT (Backpropagation through time) для визначення градієнтів, який дещо відрізняється від традиційного зворотного поширення, оскільки є специфічним для даних послідовностей. Принципи BPTT такі ж, як традиційне зворотне розповсюдження, коли модель тренується сама, обчислюючи помилки від вихідного рівня до вихідного рівня. Ці розрахунки дозволяють нам правильно відрегулювати та підігнати параметри моделі. BPTT відрізняється від традиційного підходу тим, що BPTT робить підсум помилок на кожному часовому кроці, тоді як мережі прямого пересилання не враховують підсумовування помилок, оскільки вони не ділять параметри на кожному рівні нейронної мережі.

Через цей процес РНН, як правило, перетинаються з двома проблемами, такі як “вибухові градієнти” та “зникаючі градієнти”. Ці проблеми зумовлюються розміром градієнта, який є нахилом функції втрат уздовж кривої похибки. Коли градієнт занадто малих розмірів, він буде продовжувати зменшуватися, оновлюючи всі вагові параметри, доки вони не стануть незначних розмірів - тобто 0. Алгоритм більше не навчається, коли це відбувається. Вибухові градієнти можуть з’явитися, коли градієнт занадто великих розмірів і тоді він починає створювати нестабільну модель. У цьому випадку ваги моделі стануть занадто великими і вони на кінець будуть представлені як NaN (Not a Number). Одним із вирішень цих проблем є

зменшення кількості прихованих шарів у нейромережі, усуваючи певні складності в моделі RNN.

2.1.2 Типи рекуррентних нейронних мереж

Мережі прямої передачі даних відображають - один вхід на один вихід і хоча можна візуалізувати повторювані нейронні мережі таким чином на вищенаведених діаграмах, вони насправді не мають цього обмеження. Натомість їх вхідні та вихідні дані можуть відрізнятися за довжиною і різні типи RNN використовуються для різних випадків використання, таких як генерація музики, класифікація настроїв та машинний переклад. Різні типи RNN зазвичай виражаються за допомогою таких діаграм :

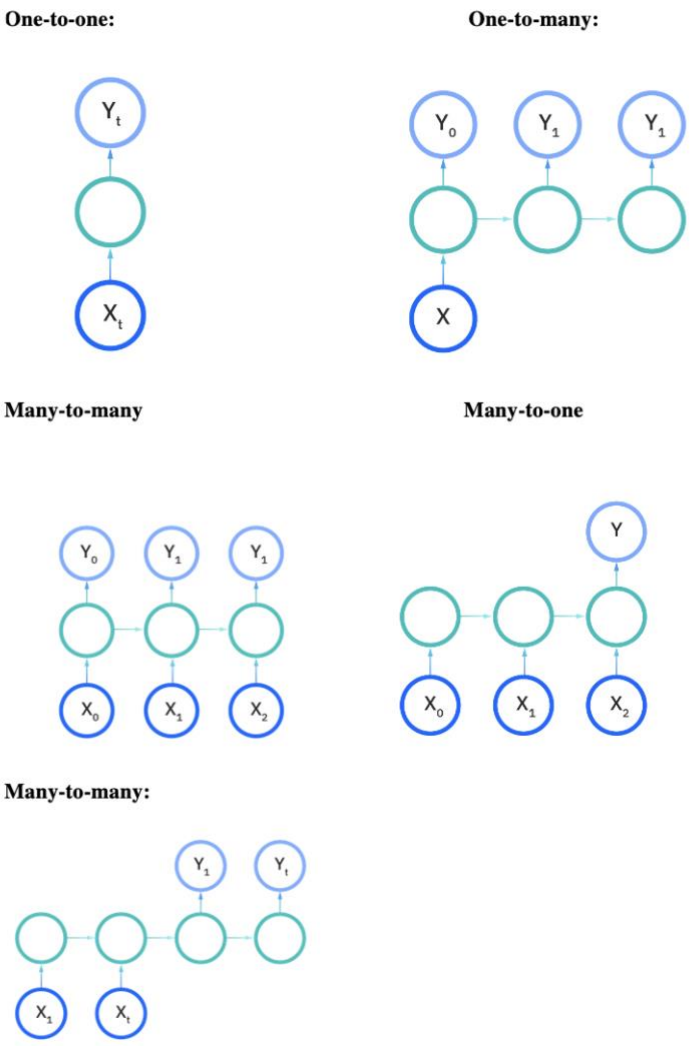


Рис. 2.3. Типи рекуррентних нейронних мереж.

2.1.3 Загальні функції активації

Функція активації визначає, чи слід активувати нейрон. Нелінійні функції зазвичай перетворюють вихідні дані даного нейрона в значення від 0 до 1 або -1 до 1. Деякі з найбільш часто використовуваних функцій визначаються наступним чином :

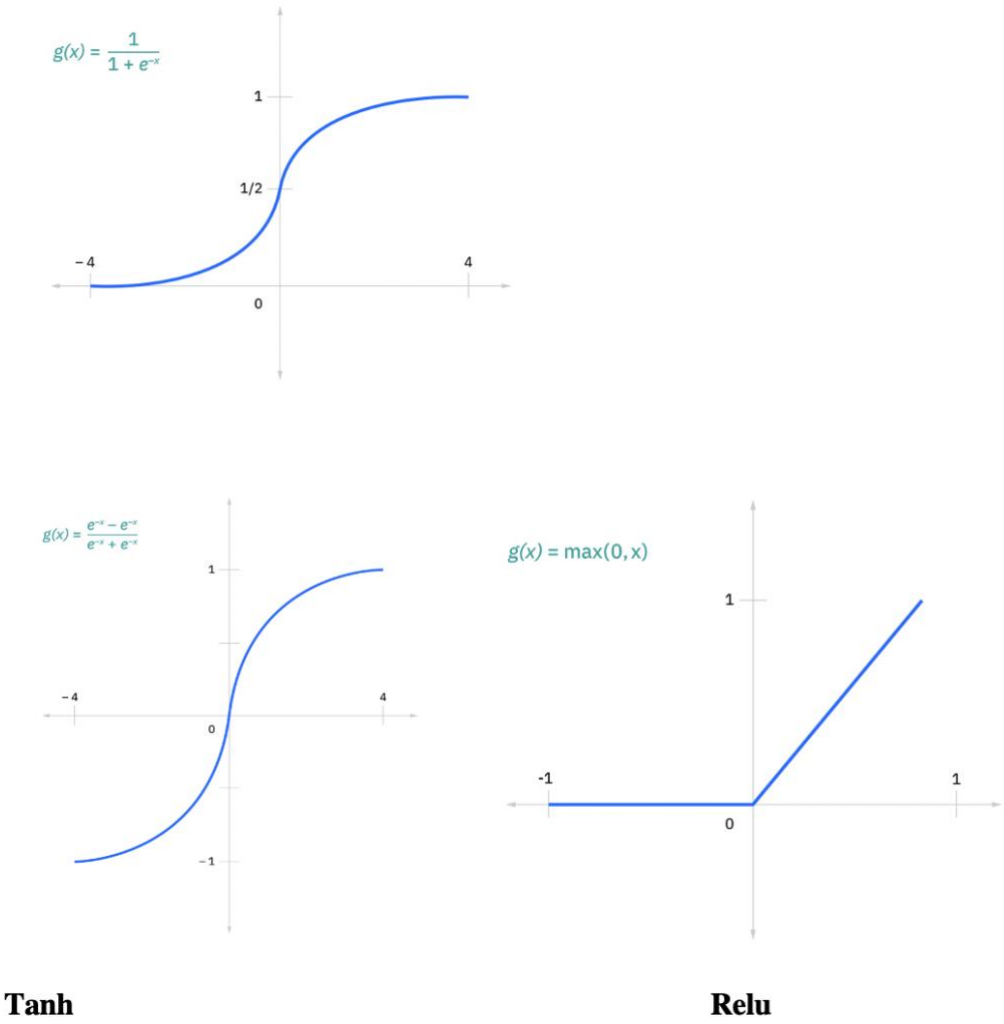


Рис. 2.4. Функції активації

2.1.4. Варіантні архітектури RNN

2.1.4.1. Глибокі RNN

Одним з найперспективніших сучасних технологій машинного навчання є застосування глибинних нейронних мереж, в основі яких лежить застосування глибокого навчання.

Глибоке навчання – це набір алгоритмів машинного навчання, які дозволяють створювати моделі з високим рівнем абстракції у вихідних даних, використовуючи архітектури нейронних мереж, що містять нелінійні перетворення сигналу.

Ми можемо збільшити глибину у трьох можливих місцях типового РНМ як наведено на рис. 5

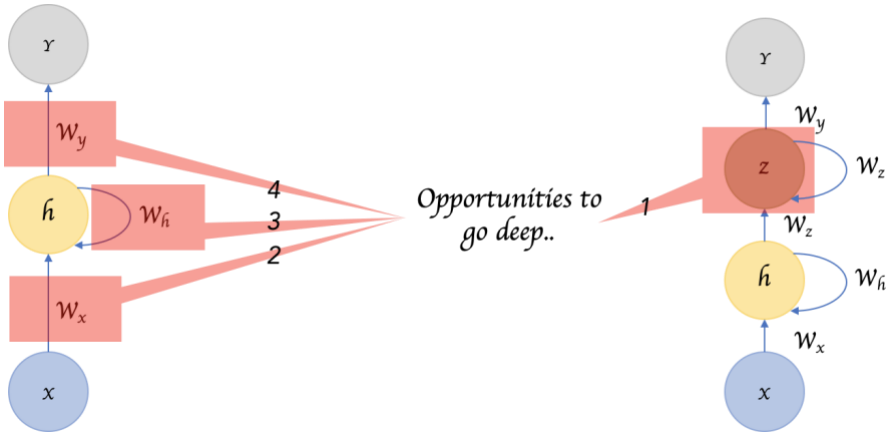


Рис. 2.5. 4 місця де можна збільшити глибину

Чотири можливі способи додати глибини :

1. Додавання прихованих станів один на інший, подаючи вихід одного до наступного.
2. Збільшення глибини від прихованого до прихованого переходу.
3. Збільшення глибини від прихованого до вихідного переходу
4. Додавання додаткових нелінійних прихованих шарів між входом до прихованого стану.

.

2.1.4.2. Двонаправлені RNN

Двонаправлені рекурентні нейронні мережі (BRNN - bidirectional recurrent neural network) - це варіант архітектури мережі RNN. У той час як односпрямовані RNN можуть витягуватися лише з попередніх входів для прогнозування поточного стану, двонаправлені RNN втягують майбутні дані для підвищення їх точності:

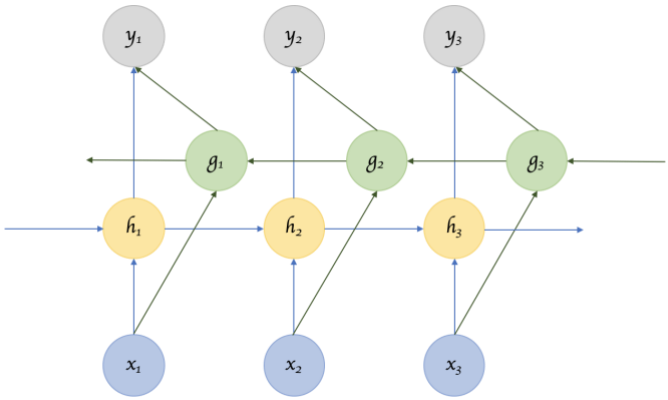


Рис. 2.6. Двонаправлені RNN

Так потрібен час для того, щоб побачити всі вхідні дані, вся операція стане дорогою. А у таких випадках, як розпізнавання мови, очікування, поки буде вимовлено ціле речення, може призвести до менш вагомого випадку використання. Тому цей метод, як зазначено раніше, найменш використовується. Але тоді як для завдань НЛП (Нейролінгвістичне програмування), де вхідні дані, як правило, доступні, скоріше за все, може бути розглянуто цілі речення за один раз. Крім того, залежно від програми, якщо чутливість до найближчих сусідів вища, ніж вхідні дані, що віддаляються, може бути змодельований варіант, який розглядає лише обмежене майбутнє або минуле.

2.1.4.3. Рекурсивні нейронні мережі

Рекурсивна нейронна мережа подібна до того, що переходи неодноразово застосовуються до входів, але не обов'язково в послідовному порядку. Рекурсивні нейронні мережі є більш загальною формою РНМ. Він може працювати на будь-якій ієрархічній деревоподібній структурі. Розбір вузлів введення, об'єднання

вузлів нащадків у батьківські вузли та поєднання їх з іншими дочірніми або батьківськими вузлами для створення деревоподібної структури. Повторювані нейронні мережі займаються чимось приблизним, але структура там суворо лінійна, тобто ваги накладаються на перший вхідний вузол, потім на другий, третій тощо:

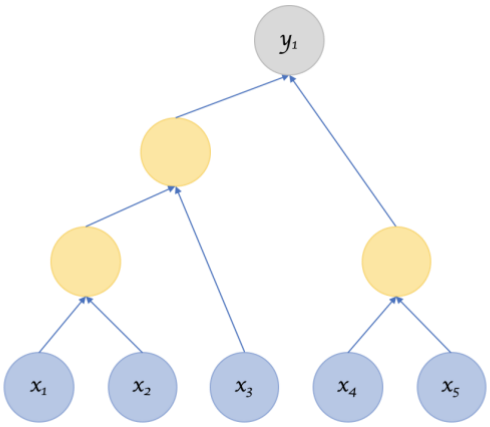


Рис. 2.7. Рекурсивна нейронна мережа

Але з цього випливає питання, що стосуються структури. Якщо структура фіксована, як у періодичних нейронних мережах, то процес навчання, зворотного зв'язку тощо має сенс, оскільки вони схожі на звичайну нейронну мережу. Але якщо структура не є фіксованою, Сохер в своєму документі зазначив [6], що в такі архітектурі сенсу ніякого немає під визначинням структури та запровадженні навчальних даних.

2.1.4.4. Енкодер - Декодер

Кодер-декодер або послідовність-послідовність RNN часто використовуються в роботі перекладу. Основна ідея полягає в тому, що існує два RNN, один кодер, який постійно оновлює свій прихований стан і видає остаточний єдиний вивід "Контекст". Потім він подається в декодер, який переводить цей контекст у послідовність виходів. Інша ключова відмінність у цій схемі полягає в

тому, що довжина вхідної послідовності та довжина вихідної послідовності не обов'язково повинні бути однаковими :

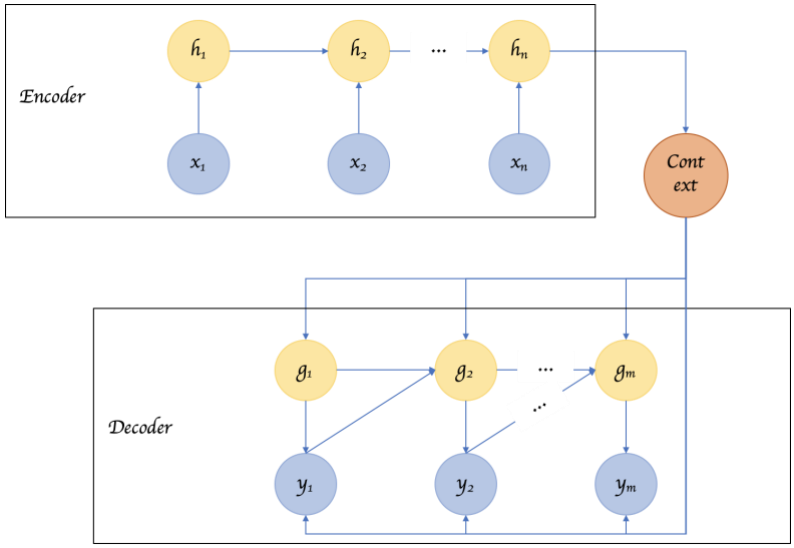


Рис. 2.8. Кодер декодера або послідовність до послідовності RNN

2.1.4.5. Довга короткострокова пам'ять

LSTM (long short-term memory - довга короткострокова пам'ять) - тип PHM, здатний навчатися довгостроковим залежностям. LSTM були представлені в роботі [Hochreiter & Schmidhuber (1997)], згодом вдосконалені і популяризували іншими дослідниками, до сих пір широко застосовуються і добре справляються з багатьма завданнями.

LSTM спеціально розроблені для усунення проблеми довгострокової залежності. Їх спеціалізація - запам'ятовування інформації протягом тривалих періодів часу, тому їх практично не потрібно навчати.

Всі рекурентні нейронні мережі мають форму ланцюжка повторюваних модулів нейронної мережі. У стандартних РНС цей повторюваний модуль має просту структуру, наприклад, один шар tanh (рис. 4).

2.1.4.5.1 Принцип роботи LSTM мережі

У LSTM зменшує або збільшує кількість інформації в стані осередки, в залежності від потреб. Для цього використовуються ретельно настроєнні структури, так звані гейти.

Гейт - це «ворота», пропускають або не пропускають інформацію. Гейти складаються з сигмовидної шару нейронної мережі і операції поточечного множення.

На виході сигмовидної шару видаються числа від нуля до одиниці, визначаючи, скільки відсотків кожної одиниці інформації пропустити далі. Значення «0» означає «не пропустити нічого», значення «1» - «пропустити все» (рис.1)

2.1.4.6. Закриті рекурентні одиниці (GRU):

Цей варіант RNN подібний до LSTM, оскільки він також працює для вирішення проблеми короткочасної пам'яті моделей RNN. Замість того, щоб використовувати "стан комірки" для регулювання інформації, він використовує приховані стани і замість трьох шлюзів у нього є два - шлюз скидання та шлюз оновлення. Подібно до входів у LSTM, ворота скидання та оновлення контролюють, скільки та яку інформацію зберігати.

2.2. Штучний інтелект і рекордна прибутковість інвестицій

Ідея використання комп'ютерів для торгівлі акціями не нова. Її аналог - алгоритмічна торгівля або чорні ящики - використовується вже більше десяти років і неухильно набирає популярність. У 2012 році алгоритмічна торгівля вже займала 85% ринку а зараз цей відсоток і зовсім перевісив за 90%.

Якщо цей тренд збережеться, 90% торгівлі буде вестися через комп'ютерні програми. Алгоритмічна торгівля сьогодні рухається в бік високочастотної HFT(high-frequency trading) - торгівлі, в якій акції купуються і продаються за частки секунди. Алгоритм швидко виявляє і використовує розбіжність, прибуток стає все менше і менше, але обсяг торгів не скорочується.

Дослідження Eurekahedge (провідний незалежний постачальник даних у світі новин, індексів та баз даних хедж-фондів) про 23 хедж-фондів [1], що використовують штучний інтелект, показало, що вони демонструють набагато кращі результати, ніж ті, що управляються людьми.

За останні шість років ці фонди домоглися річної прибутковості в 8,44% в порівнянні зі звичайними фондами, показники яких склали від 1,62% до 2,62% [1]. Автори дослідження пов'язують домінування штучного інтелекту в галузі з тим, що він постійно проводить повторне тестування, а не просто накопичує дані. Це також може бути пов'язано з недоліками традиційних квантових підходів і застосуванням торгівельних моделей, побудованих з використанням неприбуткових бектестов на історичних даних, які не здатні приносити прибуток в режимі реального часу.

Штучний інтелект нескінченно обробляє величезні масиви даних, включаючи книги, твіти, новини, фінансові показники і навіть розважальні телевізійні програми. Так він вчиться розуміти глобальні тренди і постійно вдосконалює свої передбачення про фінансові ринки.

Хедж-фонди вже давно наймають на роботу математиків, що розробляють статистичні моделі і використовують історичні дані для створення торгових

алгоритмів, які передбачають можливості ринку, але штучний інтелект робить це швидше і постійно вдосконалюється.

Ось чому фінансові гіганти, такі як Goldman Sachs, що запустив торговельну платформу Kensho на базі штучного інтелекту в 2014 році, переходять на роботизовані системи, що пророкують ринкові тренди і продають значно краще людей.

2.2.1. Перевага нейронних мереж над інвесторами.

Заробити більше середнього на фондовому ринку майже неможливо - навіть найталановитіші інвестори на Уолл-Стріт не відрізняються сталістю за даними Bloomberg (американська компанія, провайдер фінансової інформації)[2]. Трейдери і менеджери хедж-фондів не витримують конкуренції, однак їх проблема полягає в тому, що вони просто люди, в той час як всі рішення, які приймають роботи, засновані лише на даних і статистиці.

«Люди завжди залишаються упередженими й емоційними, незалежно від того, усвідомлюють вони це чи ні, - в інтерв'ю Bloomberg говорив Бабак Ходжат, співзасновник фінансового стартапу Sentient і один з розробників Siri в Apple. – «Усім відомо, що люди роблять помилки. На мою думку, набагато страшніше покладатися на здогадки й інтуїцію, а не на дані та статистику ».

Системи, на зразок тієї, що розробляє компанія Sentient (https://www.sentient.io/) може аналізувати величезні обсяги інформації, що включають ринкові дані, обсяги торгів, коливання цін, інтернет-заявки для всіх компаній, дані соцмереж, новини і відео на YouTube. Мета - домогтися того, щоб алгоритм становив оптимальний інвестиційний портфель на основі наявних знань і регулярно оптимізував його, виходячи з очікуваних нових даних за кожен місяць. Кількість подібних проектів в останні роки значно зросла. За деякими оцінками, у фінансовій сфері кількість компаній, що працюють з штучним інтелектом, досягає 1500.

Наприклад, фонд Medallion в Renaissance Technologies, що використовує кількісні методи аналізу фондового ринку, може похвалитися одними з кращих показників в інвестиційній історії. За 20 років фонд зміг повернути + 35% в річному вираженні. Це означає, що якщо б ви вклали $ 10 тис в 1997 році, сьогодні у вас на руках було б уже $ 4,04 млн [3].

Bridgewater Associates найняли команду, яка повинна побудувати автономну AI-систему під керівництвом Девіда Ферручо, в минулому розробив для IBM комп'ютер Watson, який переміг в інтелектуальної телевікторині Jeopardy [4].

Aidyia Limited, керуючий активами в Гонконзі, запустили хедж-фонд, повністю управляється штучним інтелектом. Він може читати новини на декількох мовах, аналізувати економічні дані, виявляти сумнівні шаблони, прогнозувати ринкові тенденції і після цього інвестувати.

Деякі компанії використовують штучний інтелект для забезпечення прибутковості через алгоритмічну торгівлю. Фонд Sentinent Technologies, всього за кілька хвилин може зімітувати 1800 торгових днів, зіштовхуючи трильйони віртуальних трейдерів між собою [5].

2.3 Висновок до розділу

Безліч багатообіцяючих хедж-фондів в усьому світі вже давно використовують машинне навчання для алгоритмічної торгівлі, тому що це виключає будь-які прояви ірраціональних почуттів, таких як страх і жадібність. Таку модель торгівлі, можна використовувати не тільки для того щоб замінити трейдерів, але й щоб використати змодельований аналіз для допомоги при аналізі загальновживаними методами. Для розробки таких моделей, найкраще підходять алгоритми PHM, які аналізують дані враховуючи послідовність, та пам’ятаючи як дані змінювалися, на що не спроможні інші алгоритми машинного навчання.

РОЗДІЛ 3

МОДЕЛІ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

3.1. Структура та принцип дії рекурентних нейромереж

Будову РНМ схематично зображено на рис. 3. Її виходами та входами належать шари ефекторних та рецепторних нейронів або відрізків передачі інформації. Поміж цими шарами розташовано один або декілька шарів прихованих нейронів. Входи нейронів кожного шару мають прямий зв'язок з виходами нейронів попереднього шару та можуть мати зворотний зв'язок з виходами нейронів поточного та подальших шарів. Зворотні зв'язки, здебільшого, мають елементи затримки, що надають нейромережі властивостей тимчасової оперативної пам'яті.

Нейрони різних шарів можуть бути схожими/однаковими або відрізнятись за типом функцій активації та характером нейронних парадигм. Завдяки наявності затриманих зворотних зв'язків, рекурентні мережі є динамічними системами, поведінка яких має зовнішній склад, що відповідає спостережуваним значенням виходу та входу та приховану інформацію, яка характеризує внутрішній стан нейронної мережі. Ці складові поведінки представляють такими рівняннями входу-виходу(1) та рівнянням стану(2):

$$Y(t) = F[Z(t), X(t)], \quad (1)$$

$$Z(t) = F[Z(t_0), X_{t_0}^{t}], \quad (2)$$

де Y(t) , Z(t), X(t) – значення: виходу, стану та входу динамічної системи відповідно, в момент t ;

$X_{t_0}^{t}$– реалізація стимулу на вході системи в інтервалі часу (t_0, t]

Поняття стану динамічної системи віддзеркалює її попередню поведінку. У випадку РНМ це поняття може мати доволі різні значення, залежно від тривалості інтервалу спостереження (t_0, t]. При t - $t_0 \to 0$ вектор Z(t) визначає поточний стан нейронної мережі як сукупність значень реакції нейронів. При t $- t_0 \to \infty$ Z(t)

представляє певний глобальний стан, який находиться в архітектурі нейронної мережі та значеннях ваги зв'язків між нейронами. Глобальний стан є змістом довготривалої пам’яті, сформовані при навчанні нейронної мережі, тоді як теперішній – змісту її тимчасовій оперативної пам’яті, що визначає фактор поведінки нейронної мережі в даний час роботи. Поділ на 2 стани: поточний та глобальний має сенс, коли навчальний та тестувальний процеси нейронної мережі повністю окремі. В майже всіх випадках такий поділ існує, але наприклад, у задачах адаптивного керування, рекурентна мережа має безперервно коригувати зв’язками між нейронами, відповідно до змін поведінки керуючого об’єкта. Так, що потрібно дотримуватись першого вказаного визначення і представляти стан РНМ сукупністю поточних та затриманих значень реакцій нейронів, які в даний момент діють на певні входи нейронів мережі.

3.2. Архітектура рекурентних нейронних мереж.

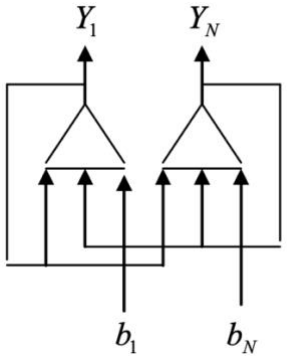
За архітектурою РН можна поділити на три типи:

1) відкриті РНМ;

2) рекурентні перцептрони;

3) ядерні РНМ.

Відкритими є рекурентні нейронні мережі, що не мають схованих нейронів. Прикладом цього є нейромережа Хопфілда, в якій міститься тільки один шар зв’язаних між собою нейронів . Її структуру представлено нижче, де величини y , b та s позначають відповідно входи, пороги нейронів та постсинаптичні потенціали, а wij – вагу зв’язків між нейронами.



$$y_i(t) = sign\left[s_i(t) - b_i(t)\right],$$

$$s_i(t) = \sum_{i=1}^{N} w_{ij} y_i(t-1).$$

Рис. 3.1. Нейромережа Хопфілда

Виходи нейронів можуть приймати такі значення : +1 або –1. замість знакової функції активації, іноді використовують сигмоїду в мережі Хопфілда. За формулою Хопфілда обчислюються вага зв’язків між нейронами:

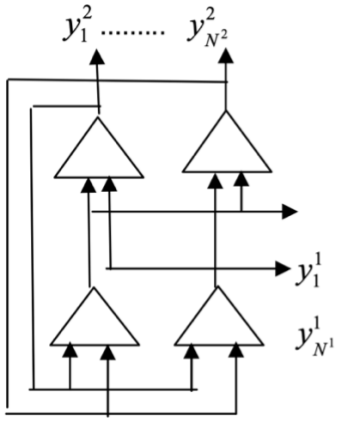
$$W_{ij} = N^{-1} \sum_{m=1}^{M} y_i^m y_j^m$$

де N є число нейронів мережі, а M – кількість векторів послідовності для навчання.

Ці вектори визначають атрактори (стійкі стани) нейронної мережі. Якщо її стан не збігається з жодним з стійких станів то відбувається процес конвергенції,

тобто нейронна мережа послідовно змінює свій стан, поки не досягне найближчого стійкого стану. Конвергенція нагадує процес асоціативного пошуку зразком, заданим у вигляді дефолтного стану нейронної мережі. Здатність до конвергенції надає змогу використовувати мережу Хопфілда як асоціативну пам’ять для відновлення перемішаних або частково пошкоджених даних.

На рис. 10 зображено двосторонню асоціативну пам’ять, що є прикладом відкритої двошарової РНМ. Її “стійкими станами” є пари векторів, що представляють реакції обох шарів.



$$y_i^2(t) = sign\left[\sum_{j=1}^{N^2} w_{i,j} y_j^1(t-1)\right] =$$

$$= sign\left[\sum_{j=1}^{N^2} w_{i,j} sign\left(\sum_{k=1}^{N^1} w_{j,k} y_k^1(t-2)\right)\right].$$

Рис. 3.2. Двостороння асоціативна пам’ять

Навчання відкритих НМ здійснюється шляхом розрахунків значень вагових коефіцієнтів на основі аналітичного рішення рівняння їхнього стабільного стану. Таке рішення існує тільки, якщо к-сть атракторів не буде більшою за число нейронів у мережі. Найвідомішим типом РНМ є багатошаровий рекурентний перцептрон (RMLP - Recurrent Multi-Layer Perceptron). На рис. 11 дано схему рекурентного перцептрона, призначеного для апроксимації часових залежностей, який має тільки один ефекторний нейрон та один схований шар нейронів.

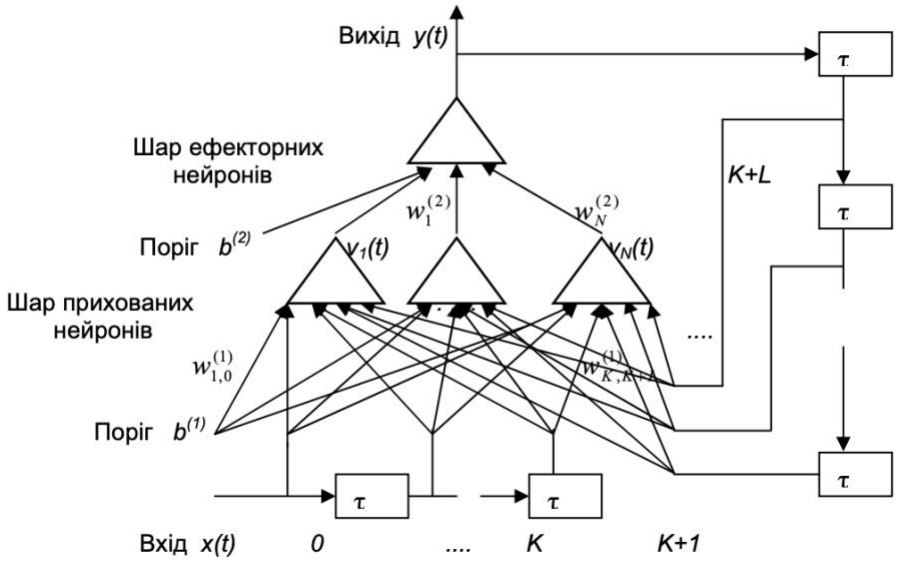


Рис. 3.3. Архітектура мережі RMLP

Мережі RMLP звичайно використовують лінійну функцію активації для ефекторних нейронів, а для прихованих – сигмоїдну. Прихований шар складається з N нейронів, на входи яких надходять сигнали з затриманння, що подаються на вхід мережі, затримані сигнали з виходів мережі, а також поріг $b^{(1)}$. Така нейромережа здійснює відображення

$$y(t+1) = F[b^{(1)}, b^{(2)}, x(t), x(t-\tau), \dots, x(t-K\tau), y(t-\tau), \dots, y(t-L\tau)]$$

де τ – крок затримки, K та L – кількість затримок сигналів на виході та вході даної мережі.

На рис. 12 представлено РНМ Ельмана, яка має змогу працювати з багатовимірними векторними даними і має певну кількість зовнішніх виходів та входів, відповідно (N та M). Щоб надати мережі необхідні властивості динаміки, у зворотні зв’язки включено елементи затримки. Зворотні зв’язки, які були названі “контекстним шаром”, разом із зовнішніми входами утворюють певний вхід для прихованого шару в нейронах.

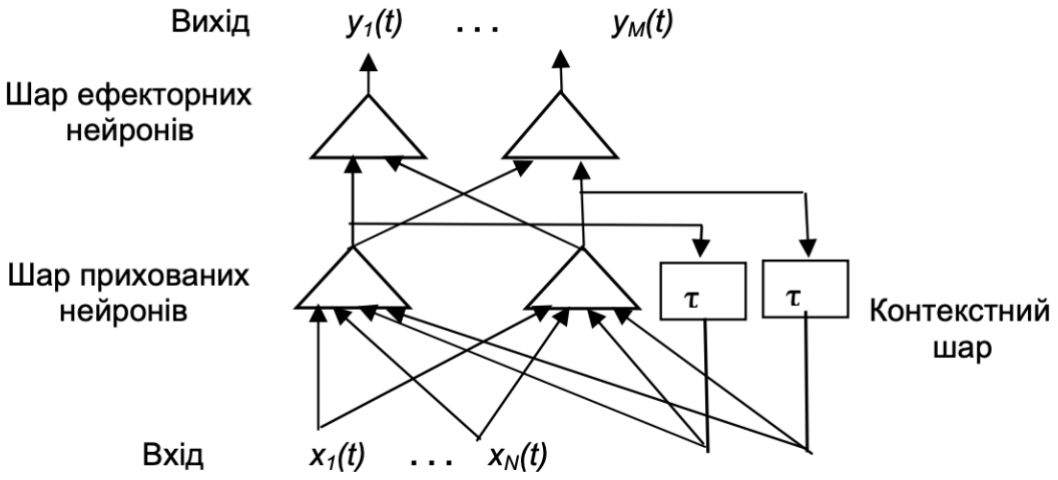


Рис. 3.4. Архітектура мережі Ельмана

RMLP та мережа Ельмана є базовими моделями, які стали основою для розробки ядерних РНМ, в яких застосовуються багато різних способів перетворення та маніпуляції вхідними даними з метою поліпшення роботи рекурентного шару нейронів. Ці рекурентні нейронні мережі, містять шар радіально-базисних нейронів, який може самостійно виконувати обробку даних на вході рекурентного шару або виконувати функції контекстного шару. Використання методів мінімізації розмірності векторів на вході рекурентного шару за допомогою адаптивної послідовності лінійних проекцій є дуже популярнию практикою. Це забезпечує використання всієї інформації яка на даний момент є в наявності про вирішувану задачу та характер вхідних даних для отримання її рішення.

Типовою можна вважати РНМ, що має L шарів по N^1 нейронів у кожному, тобто загалом $N_\Sigma = \sum_{l=1}^{L} N^l$ нейронів. На синапси (входи) кожного нейрона надходять реакції всіх нейронів попереднього шару (прямі зв’язки), теж можуть надходити затримані реакції від нейронів поточного та подальших шарів (зворотні зв’язки). Загальне число зв’язків (або синапсів) N^* включає також зовнішні рецептори (входи) нейронної мережі. Її стан можна представити N^* -вимірним

вектором $Z(t)$: $\{z_n^l(t-\tau)\}_{n=1,l=0}^{N^l,L}$, де $z_n^L(t)$ – значення реакції n -го нейрона l-го шару, $\tau \in \Theta$ – величина затримки реакції. Його компоненти $z_n^L(t)$ опридідяють реакції ефекторних нейронів або виходи нейронної мережі, а $z_n^0(t)$ – стимули, які діють на її входи. Значення реакція поточно значення довільного нейрона l-го шару можна представити як

$$z_n^l(t) = f_n^l[s_n^l(t)],$$

$$[s_n^l(t)] = \sum_{k=1}^{L}\sum_{j=1}^{N^k}\sum_{\tau\in\Theta} w_{n,j}^l z_j^k(t-\tau) + \sum_{k=0}^{N^{l-1}} w_{n,k}^l z_k^{l-1}(t),$$

де f_n^l (.)– активаційна функція нейрона;

$s_n^l(t)$ – величина постсинаптичного потенціалу (ПСП);

$w_{n,j}^l$– вага зв'язку між виходом j -го нейрона (шар, до якого належить j -й нейрон, може бути будь-яким) та входом n -го нейрона l-го шару;

$\Theta_{n,j}$– множина затримок зв'язків між входом n -го та виходом j -го нейрона.

Перша складова ПСП визначає внесок зворотних зв'язків нейронної мережі, а друга – прямих зв'язків з нейронами шару попереднього стану. Величину порога віднесено до прямих зв'язків. Її представляє складова $w_{n,0}^l z_0^{l-1}(t)$, яку можна асоціювати з впливом додаткових зовнішніх входів НМ.

Реакція $z_n^l(t)$, що оприділяє поточний стан даного нейрона, є сталою при такій умові:

$$\frac{\partial z_n^l(t)}{\partial t} = \frac{\left(\frac{\partial f_n^l}{\partial s_n^l}\right)\partial s_n^l(t)}{\partial t} = 0$$

Оскільки $f_n^l(.)$ – монотонна функція, її похідна є позитивною величиною:

$\frac{\partial f_n^l}{\partial s_n^l} = {f_{i,s}^l}' = \lambda_i^l > 0, [s_i^l] < \infty$

Тому умову (4) можна представити як $\frac{\partial s_n^l(t)}{\partial t} = 0$ або як

$$[s_n^l(t)] = \sum_{k=1}^{L} \sum_{j=1}^{N^k} \sum_{\tau \in \Theta} w_{n,j}^l z_j^k (t - \tau) + \sum_{k=0}^{N^{l-1}} w_{n,k}^l z_k^{l-1}(t),$$

Для бінарної функції активації f(s) = sign(s) умова набуває вигляду

$$sign\left(s_n^i(t)\right) = const,$$

тобто при такому випадку суттєвим є лише стабільність знаку ПСП нейрона.

Спрацювання умови для даного нейрона означає синхронізацію активності всіх нейронів, що мають зв'язки з входами даного нейрона. При цьому реакції нейронів попереднього шару, які надходять через зв'язки прямого типу, нейтралізуються реакціями тих нейронів, що надходять через зв'язки зворотнього типу. Виконання умови одночасно для всіх нейронів РНМ відповідає стану динамічного атрактора, привернутого дією зовнішнього стимулу $z^0(t)$. При збуренні зовнішнього стимулу, тобто коли поточне значення $z^0(t)$. відрізняється від значення яке було очікувано, НМ може нейтралізувати збурення і продовжити перебування у стані динамічного атрактора. Робастність (стійкість до збурень) зумовлена характером міжнейронних зв'язків РНМ, які повторюють стереотипи поведінки, засвоєні нею при навчанні. Здатність відтворювати поведінку, яка була раніше засвоєа, дає можливість застосовувати РНМ для вирішення багатьох задач, пов’язаних з відновленням та обробкою пошкоджений та зібраних даних у реальному часі.

3.2.1. Відкриті рекурентні нейронні мережі

Відкриті РНМ поділяються на два типи: статичні мережі, наприклад такі, як мережа Хопфілда або двостороння асоціативна пам'ять, або динамічні, що які звязані зворотніми зв'язками з елементами затримки, які надають їм динамічні властивосі.

На рис. 13 схематично зображено відкриту динамічну РНМ, яка має N^1 нейронів і відповідно N^1 бінарних виходів. Кількість входів нейронної мережі N^0 може відрізнятись від числа нейронів. Можна вважати, що сигнали на входах теж є бінарними. Кількість затриманих виходів N^τ може відрізнятись від числа нейронів ($N^\tau \le N^1$) Також можна вважати, що величини затримки сигналів τ однакові

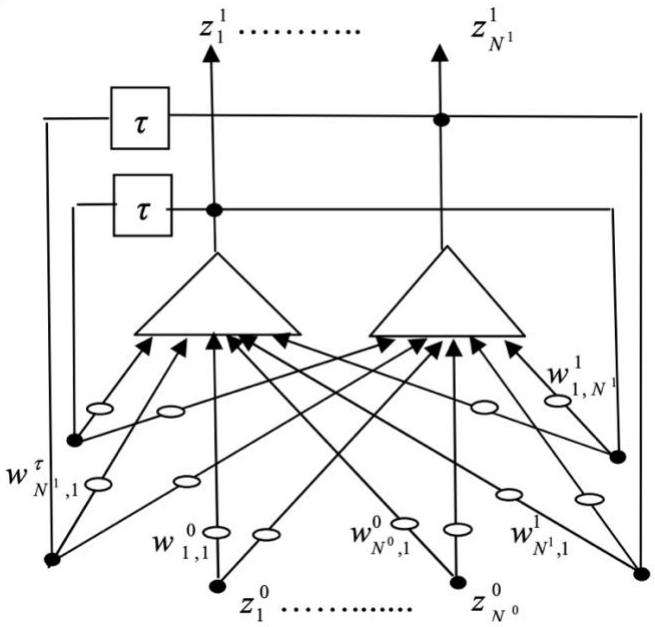


Рис. 3.5. Відкрита динамічна рекурентна нейронна мережа

Сукупність затриманих та прямих сигналів, що одночасно діють на входи всіх нейронів, визначає:

N^*– вимірний вектор стану динамічної нейронної мережі $Z^*(t) = \{z_n^*\}_{n=1}^{N^*}$, де $N^* = N^1 + N^\tau + N^0$. Розглядаючи послідовності станів у дискретні моменти часу …t-

1,t,t+1…, поточне значення реакції нейронної мережі представимо таким вектором :

$$Z^1(t+1) = F[S^1(t)] = \{f(s_n^1(\mathrm{t})\}_{n=1}^{N^1}$$

де f (.) – функція активаційна нейрона;

$S^1(t)$ – вектор ПСП (постсинаптичних потенціалів):

$$S^1(t) = W^{11}Z^1(t) + W^{1\tau}Z^1(\mathrm{t}-\tau) + W^{10}Z^0(t)$$

Де $Z^1(t)$, $Z^0(t-\tau)$ – вектори затриманих та поточних значень реакції нейронів;

$Z^0(t)$ - вектор входу нейронної мережі;

W^{11}, $W^{1\tau}$ та W^{10} – матриці ваги зв'язків між виходами та входами нейронів, затриманими виходами та входами, а також зовнішніми входами динамічної РНМ. Умовою перебування нейронної мережі в стані атрактора є виконання рівності:

$$Z^1(t) = Z^1(t+1) = F[S^1(t)] = \{f(s_n^1(\mathrm{t})\}_{n=1}^{N^1}$$

Для статичної нейронної мережі це означає, що значення виходу, а відповідно і перестає залежати від часу постсинаптичний потенціал нейронів. Це неможливо для рекурентної нейронної мережі, в якої є затримані зв'язки зворотнього типу. Її динамічні атрактори є послідовностями атракторних станів, які є реакціями на певні послідовності зовнішніх стимулів. Тоді m -й вектор даної послідовності як R^{*m}.

Компонентами його є 3 вектори: $R^{1,m}$ - поточних значень виходів нейронів, $R^{1,m-1}$ – попереднього (m−)1 -го атрактора, який відповідає затриманим виходам нейронів, $R^{0,m}$ – поточних значень зовнішнього стимулу.

Враховуючи монотонність функції активації, умову можна записати як

$R^{1,m1} = \Lambda[W^{11}R^{1,m} + W^{1\tau}R^{1,m-1} + W^{1,0}R^{0,m}]$,

де Λ – деяка позитивно визначена діагональна матриця $N^1 * N^1$ $(\lambda_{ii} \geq 0)$.

Якщо вектори $R^{1,m}$, $R^{1,m-1}$ та $R^{0,m}$ відомі, то це рівняння можна вирішити відносно матриць

W^{11}, $W^{1\tau}$ та $W^{1,0}$. Для цього представимо симетричну статичну нейронну мережу, що має N^* нейронів. Вектор стану такої нейромережі Z^*(t) має три компоненти Z^1(t), $Z^1(\mathrm{t}-\tau)$, $Z^0(t)$. ПСП уявної НМ можна представити рівнянням матричного типу:

$$S^*(t)=\begin{vmatrix} S^1(t) \\ S^1(t-\tau) \\ S^0(t) \end{vmatrix}=\begin{vmatrix} W^{11} & W^{1\tau} & W^{10} \\ W^{\tau 1} & W^{\tau\tau} & W^{\tau 0} \\ W^{01} & W^{0\tau} & W^{00} \end{vmatrix}\begin{vmatrix} Z^1(t) \\ Z^1(t-\tau) \\ Z^0(t) \end{vmatrix}=\begin{vmatrix} W^{11}Z^1(t)+W^{1\tau}Z^1(t-\tau)+W^{1\tau}Z^0(t) \\ W^{\tau 1}Z^1(t)+W^{\tau\tau}Z^1(t-\tau)+W^{\tau 0}Z^0(t) \\ W^{01}Z^1(t)+W^{0\tau}Z^1(t-\tau)+W^{0\tau}Z^0(t) \end{vmatrix}.$$

Можна допустити, що вектори її атракторних станів є тотожними тим, що утворюють динамічні атрактори в динамічній РНМ. Якщо розглянути динамічні атрактори як послідовності векторів, які буду змінювати один одного в дискретні моменти часу...t $-\tau$,t,t $+\tau$... , можна встановити відповідність між станами динамічної НМ та її атракторами: $Z^1(\mathrm{t}) \to R^{1,m}$, $Z^1(\mathrm{t}-\tau) \to R^{1,m}$. Тепер послідовність атракторних станів можна представити у матричному вигляді:

$$\Re^*=\begin{vmatrix} R^{1,1}\ldots & R^{1,m}\ldots & R^{1,M} \\ R^{1,0}\ldots & R^{1,m-1}\ldots & R^{1,M-1} \\ R^{0,1}\ldots & R^{0,m}\ldots. & R^{0,M} \end{vmatrix},$$

стовпчиками - вектори статичних атракторів уявної нейронної мережі. Користуючись аналогією з рівнянням, запишемо узагальнене рівняння атракторного стану уявної НМ:

$$\Re^* = \Lambda W^* \Re^*$$

Вирішуючи його відносно W^* одержимо

$$W^* = \Lambda^{-1}\Re^*(\Re^*)^+$$

Де $(\Re^*)^+$ – псевдообернена матриця $\Re^*$. Оскільки вектори $R^{1,m}$ та $R^{0,m}$ є бінарними, то можна вважати $\Lambda = l$

Для обчислення матриці W^* застосовують псевдоінверсне правило:

$$w_{i,j}^{*m+1} = w_{i,j}^{*m} + (r_i^{m+1} - s_i^{m+1})(r_j^{m+1} - s_j^{m+1})/d^{m+1}$$

$$s_i^{m+1} = \sum_{k=1}^{N^*} w_{ik}^{*m} r_k^{m+1}, \quad d^{m+1} = \sum_{K=1}^{N^*} r_{iK}^{m+1} (s_k^{m+1} - s_k^{m+1})$$

Де r_k^{m+1} – компонента вектора R^{*m+1}. Матриця $W^* = \Re^* (\Re^*)^+$ є проекційною в лінійному просторі $\Im$, напнутому на M векторів з $\Re^*$. Вона має такі властивості:

$$W^* = (w^*)^2,\ w_{i,i}^* = \sum_{j=1}^{N^*} (w_{i,j}^*)^2, Tr W^* = \sum_{i=1}^{N^*} w_{i,i}^* = M,$$

$$\underset{w_{i.i}^*}{\rightarrow} = M/N^*,\ M/N^*[1 - M/N^*] \geq \left(w_{i,j}^*\right)^2 \geq, M/(N^*(N^* - 1)[1 - M/N^*],\ i \neq j$$

Величина елементів які не є діагональними знаходиться в межах між верхньою оцінкою, яка відповідає розрідженій матриці, більшість елементів якої мають значення, приблизні до нуля та нижньою, що відповідає майже рівномірному розподілу її елементів. Діагональні елементи визначають вагу зворотних позитивних зв'язків нейронів. При збільшенні відношення M/N^* для нейронів, вони втрачають чутливість до зовнішніх змін, які можуть стати причиною появи хибних атракторів та зупинки процесу конвергенції.

При відсутності затримок зворотних зв’язків (τ = 0) розрахована за формулами симетрична НМ відповідає моделі асоціативної пам’яті, керованої порогом, яка має M < N^1 головних атракторів, представлених парами векторів $\{R^{1,m}, R^{0,m}\}_{m=1}^{M}$. Якщо така нейронна мережа попадає в нестабільний початковий стан, то відбувається процес послідовних змін стану в бік ближнього головного атрактора (конвергенції). Цей процес може не досягти головного атрактора через зупинку в локальному (хибному) атракторі. З підвищенням співвідношення M N - ймовірність такої зупинки зростає. При M / N > 0,14 у мережі Хопфілда процес конвергенції зупиняється. А якщо M / N > 0,25 при обчисленні зв'язків за псевдоінверсним правилом зупинка настає. Застосування до такої НМ метод рознасичення синаптичної матриці дозволяє підняти цей рівень до M / N≈ 0,7 . При наявності затриманих зворотних зв'язків у матриці W^* залишаються блоки W^{11}, $W^{1\tau}$ та W^{10} , які відповідають реальним зв’язкам. Процес конвергенції обмежений

першим кроком переходу нейронів в наступний з поточного стану у такій нейронній мережі. Це обмеження не заважає існувати таким динамічним атракторам РНМ, для яких наступний стан є іншим атрактором. Динамічний атрактор може початися з любого вектора послідовності, представленою матрицею $\Re^*$ і закінчуватись останнім її вектором $R^{*,m}$. Якщо початковий стан не є атрактором, ситуація може бути більш складною, так як на відміну від симетричної статичної НМ, стан якої монотонно наближається до найбільш близького атрактора при конвергенції. Поведінка динамічної РНМ може бути не монотонною поза динамічним атрактором.

3.2.2. Процес конвергенції в статичній рекурентній нейромережі.

Ітерація конвергенції в статичній НМ представляє послідовність нелінійних та лінійних перетворень вектора стану. Повертаючись до формул і вважаючи, що ітерація конвергенції здійснюється за одиницю часу, получається:

$$Z^1(t+1) = F[S^1(t)] = \{f(s_t\ (\mathrm{t})\}_{n=1}^{N^1}$$

$$s_t\ (\mathrm{t}) = \sum_{j=1}^{N} w_{i,j}^1 z_j^1(t-1) + \sum_{j=1}^{N} w_{i,j}^0 z_j^0(t)$$

Ітерація включає лінійну операцію обчислення значення ПСП та його перетворення в реакцію НМ нелінійною функцією активації, яка буде представленою ступеневим рядом:

$$f[s_i] = a_1 s_i + a_3 s_i^3 + \ldots . a_{2p+1} s_i^{2p+1} + \ldots . = s_i \sum_{p=0}^{\infty} a_{2p+1} s_i^{2p} .$$

Оскільки ця функція є непарною і монотонною, то коефіцієнти ряду містять різноманітні знаки, а їх абсолютні значення зменшуються дуже швидко. Це можна бачити на прикладі гіперболічного тангенса:

$$thx = x - \tfrac{1}{3}x^3 + \tfrac{2}{15}x^5 - \tfrac{17}{315}x^7 + \ldots .$$

При розгляді ітерації конвергенції для компоненти вектора ПСП:

$$s_i(t) = \sum_{j=1}^{N} w_{i,j}^1 z_j^1(t-1) + \sum_{j=1}^{N} w_{i,j}^0 z_j^0(t) =$$

$$= \sum_{j=1}^{N} w_{i,j}^1 \{[\sum_{k=1}^{N} w_{j,k}^1 z_j^1(t-2) + \sum_{j=1}^{N} w_{j,k}^1 z_k^0(t-1)] \sum_{p=0}^{\infty} a_{2p+1} s_k^{2p}\} + \sum_{j=1}^{N} w_{i,j}^0 z_j^0(t).$$

Переставляючи члени і враховуючи, що для матриць, розрахованих за псевдоінверсними правилами, має місце співвідношення $\sum_{J=1}^{N} w_{i,j} w_{j,k} = w_{j,k}$,одержимо

$$s_i(t) = a_1 \sum_{j=1}^{N} w_{i,j}^1 z_j^1(t-2) + \sum_{j=1}^{N^0} w_{i,j}^0 [z_j^0(t) + a_1 z_j^0(t-1)] + \sum_{j=1}^{N} w_{j,k}^1 [\sum_{p=0}^{\infty} a_{2p+1} s_k^{2p}].$$

Перші дві складові є проекціями попередньої реакції $Z^1(t-2)$ та накопиченого значення зовнішнього стимулу $Z^0(t) + a_i Z^0(t)$ в лінійний простір матриці W^1. Останій склад визначає проекцію суми старших членів ряду, які були утворині при нелінійному перетворенні попереднього значення постсинаптичного потенціалу. Якщо представити ортогональний розклад матриці W^1, наслідки даної проекції стають зрозумілішими:

$$W^1 = YDY^T,$$

де Y – матриця L × N , L ≤ N, стовпчиками якої є власні вектори, а D – діагональна матриця, що представляє спектр матриці W^1. Власні вектори Y утворюють базис лінійного простору, в який проектується вектор Z(t):

$$S(t) = W^1 Z(t) = Y\Psi_1(t),$$

Де $\Psi_1(t) = DY^T Z(t)$ – спектр поточного значення постсинаптичного потенціалу.

При перетворенні ПСП нелінійно, виникають комбінаційні компоненти, частина яких є ортогональними до лінійного простору w^1 . Наступне лінійне проектування залишає комбінаційні компоненти та звільнює спектр ПСП від цих компонентів, які віддзеркалюють спектральний склад попереднього постсинаптичного потенціалу. Послаблюються найменші та підсилюються найбільші складові завдяки нелінійному перетворенню в цьому залишку. Конвергенції диференціація складових спектра ПСП зростає при наступних ітераціях, поки не залишиться одна компонента, яка визначатиме атракторний стан нейронної мережі.

У спектрі проекції будуть сильнішати складові, які відповідають найбільшим власним значенням матриці W якщо власні числа матриці W відрізняються від 0 або 1. Проекція вектора стану може віддалятись від оригіналу, заданого зовнішнім стимулом і наближатись до найбільшого власного вектора матриці W, у цьому випадку. Тому нейронна мережа може втратити свої асоціативні властивості при порушенні проекційності матриці W

3.2.3. Ітеративне навчання рекурентних нейромереж

Для РНМ, в якої є приховані нейрони, не може бути заздалегідь визначено вектори атракторних станів. Тому значення ваги зв’язків між нейронами у таких нейронних мережах обчислюють , шляхом поступового коригування параметрів НМ, ітеративно в напрямку антиградієнта похибки. Найбільш ефективним є Backpropagation algorithm (метод зворотного поширення похибки), відповідно який ефективно використовується для навчання багатошарових нейронних мереж. Застосування цього методу до РНМ ускладнюється тим, що будь-яка корекція параметрів нейроннї мережі викликає зміни попередніх значень реакції нейронів. Розроблено метод зворотного поширення похибки в часі для подолання цих ускладнень, за яким послідовність операцій над даними в час розгортається у вигляді багатошарової нейронної мережі прямого поширення, до якої для кожного

кроку затримки в часі додають новий шар нейронів. Навчання такої НМ проводять за критерієм мінімуму величини похибки, яку визначають як

$$E = \frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N^L}(d_n^m - z_n^{L,m})^2,$$

Де d_n^m – очікувана реакція нейронної мережі;

$z_n^{l.m}$ – дійсне значення реакції;

N^L– число ефекторів (виходів) мережі;

M – розмір (число векторів) епохи навчання. Коригування вагових коефіцієнтів здійснюється за дельта-правилом:

$$(w_{p,q}^l)_{new} = (w_{p,q}^l)_{old} - \mu\,\partial E / \partial w_{p,q}^l,$$

де μ – коефіцієнт швидкості навчання.

$$\frac{\partial E}{\partial w_{p,q}^l} = \frac{\partial E}{\partial z_p^l}\frac{\partial z_p^l}{\partial w_{p,q}^l} = -\sum_{m=1}^{M} q_p^{l,m}\frac{\partial z_p^{l,m}}{\partial w_{p,q}^l},$$

$$q_p^{l,m} = \sum_{n=1}^{N^L}(d_n^m - z_n^{Lm})\frac{\partial z_n^{L,m}}{\partial z_p^{l,m}}.$$

Тут $z_p^{l,m}$ – при надходженні m -го вектора послідовності навчання це значення реакції p -го нейрона l-го шару мережі:

$$z_p^{l,m} = f_p^l\left(\sum_{j=0}^{N_p} w_{p,j}^l z_j^m\right),$$

де N_p – множина нейронів, що мають зв’язки з входами цього нейрона. Індекс зв’язку j = 0 позначає вхід порога, який можна враховувати як постійна величина. Решта зв’язків поділяється на прямі зв’язки, які ідуть від попереднього шару та зворотні зв’язки, які ідуть від нейронів наступних шарів, тобто тих, що більш близькі до виходу мережі. Їхні значення відповідають попереднім елементам послідовності навчання, так як реакції останніх вказаних приходять із затримкою.

Така різниця між зворотними та прямими зв'язками являється серйозним ускладненням процес обчислень за формулою. Потрібно включати залежність поточної реакції даного нейрона від затриманих значень реакції попередніх нейронів, у випадку зворотних зв'язків, якщо для прямих зв'язків діє звичайне ланцюгове правило,. Наприклад, якщо $w_{p,q}^{l}$ – вага затриманого на r кроків зворотного зв'язку входу p -го нейрона з виходом q -го нейрона l+1-го шару, похідна обчислюється як

$$\partial z_p^{l,m} / \partial w_{p,q}^{l} = f_p^{l\prime} \partial s_p^{l,m} / \partial w_{p,q}^{l} = f_p^{l\prime} z_q^{l+1,m-r},$$

$$\lambda_p^{l,m} = g_p^{l\prime}(s_p^{l,m}) / g s_p^{l,m}, \qquad s_p^{l,m} = \sum_{i=0}^{N_p^l} w_{p,i}^{l} z_i^{l},$$

де Np – множина входів p -го нейрона, яка включає як зворотні, так і прямі звязки.

Коригування ваги зв'язків використовуючи попередню формулу приводить до зміни попередніх значень ПСП, а відповідно і минулих значень нейронних реакцій z_j^{m-r}. Застосовують послідовну схему зображену на рис. 14 корекції ваги прямих (W t) та зворотних (W u) зв'язків, щоб врахувати ці зміни.

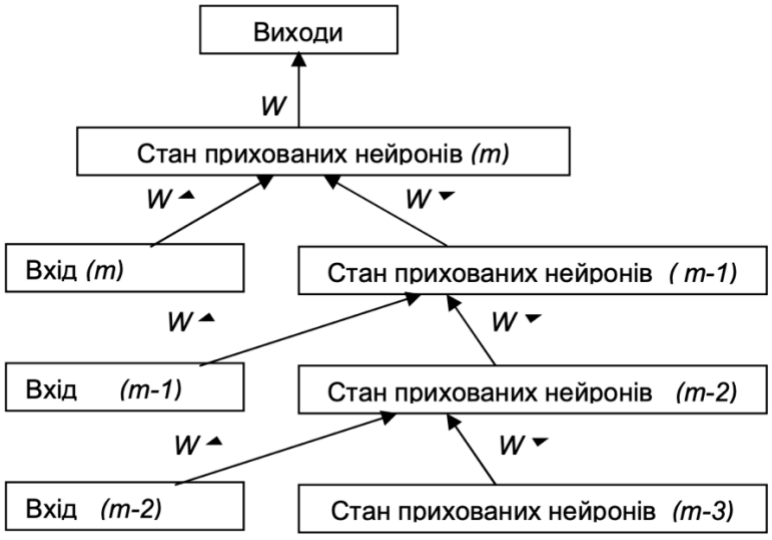


Рис. 3.6. Схема корекції ваги прямих (Wt) та зворотних (Wu) зв’язків за методом зворотного поширення в часі

При обчисленні за цією моделлю потрібно запамятовувати значення виходів нейронів мережі для кожного з M елементів послідовності навчання. На практиці здебільшого використовуються скорочена процедура, за якою враховується тільки останні 10–20 елементів навчальної послідовності, а решту відкидають. Під час проходження експериментів, таке обмеження не має ніякого впливу на результати навчання, в майже всіх випадках.

Метод зворотного поширення (Backpropagation algorithm) в часі звичайно застосовують для навчання в кумулятивному (пакетному) режимі. Множину даних послідовності для навчання поділяють на послідовні епохи. Корекція ваг даних здійснюється раз за епоху (epoque), розмір якої обирається набагато меншим тим даним які містяться в навчальній послідовності. Розміри епохи можуть бути обмежені величиною від 20ти до 50ти векторів, що відповідає скороченому навчанню з відсіканням. Для кожної епохи послідовно для кожного моменту часу обчислюють значення локальних градієнтів похибки. Одержані значення градієнтів сумують у межах даної епохи і тільки після її завершення, проходить процес коригування ваги зв’язків. Ця процедура буде повторюватися поки не будуть пройдені всі епохи. Навчання по епохах повторюють багато разів на всій послідовності для навчання, доки похибка реакції нейронної мережі не буде зменшиною до задовільного рівня.

Дуже важливим є некумулятивний метод рекурентного навчання в реальному часі, що не потребує поділу послідовності для навчання на окремі епохи. Цей метод являє собою обчислення градієнта похибки для всіх елементів послідовності навчання при відсіканні решти її елементів, крім попереднього. Застосування цього методу сполучають з використанням елементів затримки сигналів як на вході нейронної мережі, так і в зворотних її зв’язках. Це надає можливість використовувати дані послідовності для навчальної в повному обсязі і наблизитись

до режиму кумулятивного навчання з відсіканням. При використанні РНМ для адаптивної обробки потоків даних у реальному часі застосовують метод безперервного рекурентного навчання, за яким модифікацію параметрів нейронні мережі, без повторів проводять послідовно для кожного елемента вхідного потоку даних. Процес модифікації ніколи не буде завершеним, а його результат буде оцінено за розміром похибки на виході нейронної мережі. Застосовують для моделювання нестаціонарних процесів, таких як: процесів мовлення, коли не існує можливості зупинитись для повторення фрагментів даних поданих при вході. Загалом градієнтні методи навчання РНМ характеризуються відсутністю гарантії збіжності цього процесу та повільною швидкістю при навчанні. Тому пошук задовільних результатів потребує значних зусиль. Для покращення ефективності такого пошуку існує декілька евристик:

- принцип поступовості у збільшенні розмірів рекурентного шару нейронної мережі та обсягу послідовності для навчання. За даним принципом додавання нових елементів може бути можливе тільки якщо буле досягнуто певний прогрес на попередній стадії навчання.
- принцип, під назвою “підсилення вчителя”, полягає в тому, що при навчанні за методом backpropagtion (зворотного поширення) в часі всі минулі реакції мережі, які надходять по зворотних зв’язках, замінюють очікуваними значеннями цих реакцій. Врахованим буде лише значення похибки, яке було одержане для останньої реакції нейронної мережі. Це може іноді дозволити значно зменшити час тривалості навчального процесу через уникнення появи проміжних похибок.

3.2.4. Проблема обчислювальної складності навчання РНМ

Основною проблемою, що зупиняє практичне застосування більшості РНМ, є висока складність в процесі їх навчання. Виключенням є відкриті рекурентні нейронні мережі, але їх застосування обмежено задачами, в яких множина динамічних образів є сталою. Обчислювальна складність навчання за методом

зворотного поширення в часі становить від O(N^3) до O(N^4), що значно перевищує оцінку O(N^2) для статичних нейронних мереж. Найбільша витрата ресурсів припадає на коригування зворотних зв'язків нейронної мережі, тому для вирішення цієї проблеми потрібно суттєвог удосконалити архітектуру та методи коригування зворотних зв'язків РНМ або взагалі під час процесу навчання відмовитися від їх модифікацій. Дослідження з удосконалення архітектури привели до створення гібридних РНМ із застосуванням нейропарадигм радіально-базисних функцій, мапи Кохонена та нечіткої логіки. Деякі з цих моделей показали вражаючі результати: тривалість навчання складала лише декілька десятків епох. Більша ефективність цих нейронних мереж можна пояснити тим, що вони мають можливість кращого пристосування до даної задачі та її характеру. При збільшенні розмірності вирішуваної задачі або зростанні коефіцієнту невизначеності даних при вході їхні переваги втрачаються. Другий можливей шлях покращення процесу навчання базується на приближенні нелінійної динаміки РНМ послідовністю лінійних моделей, реалізованих за допомогою фільтра Калмана або методів динамічного програмування. Як і гібридні нейронні мережі, такі методи апроксимації нелінійної динаміки ефективні лише для відносно простих рекурентних нейронних мереж 22 ISSN 1028-9763. Математичні машини і системи, 2009, № 3 при невеликому відхиленні прогнозованих або аналізованих процесів від послідовностей, використаних під час навчання. Вирішенням цієї проблеми може бути відмова від коригування зворотних зв'язків при навчанні РНМ. Цю ідею покладено в основу запропонованої Джагером ESN (Echo State Network) :

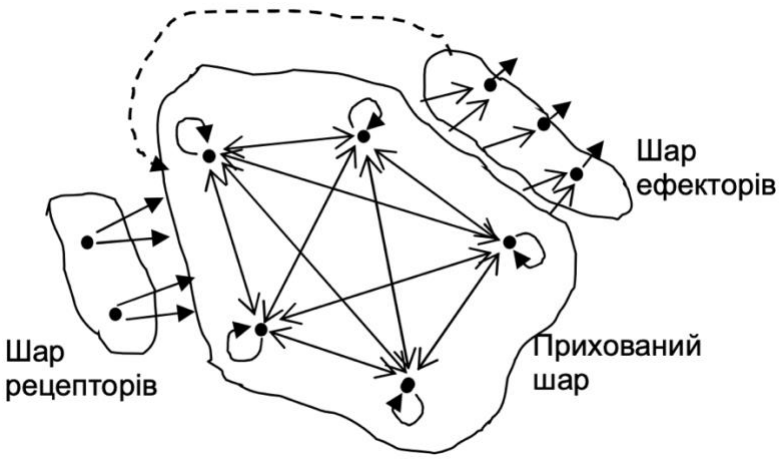


Рис. 3.7. Нейромережа Echo State Nenwork

При навчанні цієї рекурентної нейронної мережі модифікацію проходять лише прямі зв’язки нейронів з ефекторами в прихованому шарі. Зворотні та прямі зв’язки між нейронами прихованого шару не буде змінено. Деякі моделі Echo State Nenwork можуть мати також постійні зворотні зв’язки між виходами ефекторів та входами нейронів прихованого шару.

При створенні мережі встановлюються зв’язки прихованого шару які мають постійні рандомні значення. Реакції ефекторних нейронів мережі ESN віддзеркалюють лінійні композиції динамічних реакцій нейронів прихованого шару оскільки під час навчання можуть модифіковуватися лише прямі зв’язки з ефекторними нейронами. Останні створюють динамічну мозаїку, яка показує динаміку процесу на вході нейронної мережі. З цього походить причина розглядати прихований шар ESN мережі, як деякий контейнер, що містить частки динамічних образів, з яких формуються реакції НМ. Тому мережу ESN часто називають рідинною, або резервуарною (Liquit State Machine).

Можливість застосування до них звичайних методів навчання, розроблених для мереж прямого поширення, зокрема, перцептронів є головною перевагою ESN мережі. Але при цьому існує проблема швидкого збільшення спектрального радіуса синаптичної матриці ефекторних нейронів, тобто зростання відстані без ніяких меж між максимальними та мінімальними значеннями ваги зв’язків. Ця проблема

пов’язана в неможливості усунення похибок реакції (в повному обсязі) ефекторів за рахунок перерозподілу ваги зв’язків з нейронами прихованого шару, реакції яких у навчальному процесі не змінюються. Існують два варіанти до її вирішення:

1) створення мультифункціональних рекурентних нейронних мереж, контекстний шар яких представляє множину нелінійних динамічних модулів;

2) декореляція реакцій контекстного шару нейронів з використанням методу, похожого до методу “conjugate gradients” (узгоджених градієнтів).

Обидва методи(підходи) були створені на припущенні про консервативність контекстного шару, властивості якого дуже повільно змінюються, в порівнянні з реакціями нейронної мережі на зміни зовнішніх стимулів, яке відповідає придставленню про довготермінову пам’ять як контейнер або резервуар, що містить динамічні компоненти, з яких формується суцільна поведінка нервової системи.

3.3. Висновок до розділу

Зараз очевидно те, що не існує універсального шляху вирішення проблеми при розв’язуванні певної задачі за допомогою РНМ. Вибір архітектури рекурентної нейронної мережі та методів її навчання залежить від обсягу та характеру вирішуваної прикладної задачі. Для нескладних прикладних задач автоматичного керування можуть застосовуватись гібридні РНМ, які дозволяють найбільш повно використовувати апріорні дані про вирішувану задачу для поліпшення роботи нейронної мережі. Для швидшого процесу навчання таких нейронних мереж можна застосувати методи динамічної лінійної апроксимації, зокрема, фільтр Калмана. Перспективним є використання відкритих РНМ, навчання яких здійснюється неітеративно, шляхом розрахунку.

Коренем проблеми ресурсоємності навчання є рівень складності моделі оточення, яку будує нейронна мережа у процесі навчання. Для звичайних статичних нейронних мереж оточення і його модель є статичними замкненими системами.

Найбільш радикальне вирішення проблеми ресурсоємності обіцяє концепція ESN резервуарної РНМ, за якою зв’язки нейронів рекурентного шару при навчанні не змінюються, а модифікуються лише прямі зв’язки з шаром ефекторів.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1. Реалізація нейронних мереж

На сьогоднішній день існує ряд готових рішень, створених великими корпораціями і ака-деміческой групами. Ці рішення істотно спрощують процес розробки та навчання искусст-венного інтелекту, надаючи різний рівень абстракцій. Можна виділити низькорівневі фреймворки, які реалізують математичну логіку, дозволяючи користувачеві самому реалізови-вать логічні одиниці нейронної мережі. Також су-ществуют фреймворки більш високого рівня, кото-які надають готові абстракції для більшої зручності і швидкості розробки системи.

1. TensorFlow - програмна бібліотека для глибокого навчання, розроблена Google для ре-ня завдань побудови і тренування нейронних мереж. На даний момент є однією з найбільш широко застосовуваних бібліотек як для наукових досліджень, так і для комерційних проектів. Є продовженням закритого проекту DistBelief, який

створювався Google для внутрен-него використання. Дана бібліотека реалізована на Python, також є офіційні реалізації для C ++, Haskell, Java і Go. Однак для інших Відомими мов програмування мають не-офіційні обгортки, розроблені пользовате-лямі.TensorFlow заснована на графах операцій, ко-торие оперують з тензорними обчисленнями. Та-ким чином бібліотека являє собою низько-рівневий інструментарій без прив'язки до конкрет-вим об'єктів нейронної мережі. Продуктивність даної бібліотеки полягає в використанні символьного підходу до вичісленіям [7].

2. Theano - бібліотека, яка використовується для розробки систем машинного навчання як сама по собі, так і в якості обчислювального бекенда для більш високорівневих бібліотек, наприклад, Lasagne, Keras або Blocks [8]. Theano розробляється з 2007 року, головним чином, групою MILA з Уні-версітета Монреаля і названа на честь давньогрецької жінки-філософа і математика Феано. Основними принципами є: інтеграція з numpy, прозрач-ве використання різних обчислювальних вуст-влаштування (зазвичай GPU), динамічна генерація опти-мізірованного C-кода.Theano, як і TensorFlow, ис-помагає символьний підхід до вичісленіям.

3. Gluon - програмна бібліотека для реа-лізації нейронних мереж, розроблена спільно компаніями Microsofti Amazon.Ее характерними особливостями є швидке прототіпірова-ня, оптимізація для роботи в хмарних службах, розпаралелювання обчислень і упор на оптимізатора-цію LSTM-модулів і рекурентних-сетей [9]. Также Gluonподдерживает роботу з вбраними і кван-Това даними, що є великим пре-майном при роботі з природним язиком.Gluon орієнтований для роботи з хмарними сис-тема і володіє для цього широким API. У свою чергу це дозволяє підтримувати складні мето-ди, наприклад, динамічні графи і гнучкі струк-тури, без необхідності розбиратися в конкретних деталях і оптимізувати всі вручну.

4. Cognitive Toolkit (CNTK) - безкоштовна про-програмних бібліотека з відкритим вихідним кодом, розроблена для глибокого машинного навчання компанією Microsoft [10]. На сьогоднішній день є однією з найбільш популярних

бібліотек для по-будови нейронних мереж і головним конкурентом вищеописаного TensorFlow. З основних особ-ностей і відмінностей можна виділити:-швидкість. CNTK в цілому працює швидше, ніж TensorFlow, а в рекурентних мережах дає аж до п'яти-і десятикратного виграшу в виробляй-ності; -Структура API. CNTK має гнучкий і мощ-ний API для C ++ і пропонує як низькорівневі, так і прості у використанні високорівневі Python API на основі парадигми функціонального програмування; -масштабіруемость. CNTK легко масштаби-ється і в разі обчислювально вимогливих завдань може виконуватися хоч на тисячах графич-ських процесорів; -скорінг. У CNTK є продуктивний Eval API для C ++, .NET, Java і Python, для спрощена-ня інтеграції нейронних мереж в свої додатки; -расширяемость. CNTK легко розширюється завдяки можливості використання Python для оп-ределенном власних шарів і процедур обученія.-вбудовані модулі зчитування. У CNTK є невимогливі до пам'яті вбудовані средст-ва читання даних.

4.2. Реалізація коду

4.2.1 Алгоритм

1. Отримуємо та записуємо вхідні історичні дані активу.
2. Встановлюємо в таблиці дати даних як індекс.
3. Копіюємо отриману таблицю для подальшої роботи.
4. Отримуємо в окремі таблиці дату та закриття денної ціни активу
5. Створюємо масиви для тренування з пропорцією 70/30 та валідації моделі.
6. Всі дані для роботи нормалізуємо.
7. Добавляємо в робочу таблицю даних , розраховані показники RSI і MACD(рис. 17, рис. 19).

- Індекс відносної міцності (RSI): Індикатор технічного імпульсу, який порівнює величину останніх прибутків з останніми втратами в спробі визначити перекуплені та перепродані умови активу. Формула для обчислення показника відносної міцності така.

RSI = 100- [100 / (1 + RS)]

Де RS = Середня значення з x денних верхніх закриттів / середнє значення x денних нижніх закриттів.

- Рухаюча середня конвергенція / розбіжність (MACD): Ця функція обчислює різницю між коротко- та довгостроковими ковзними середніми для поля. Формули для обчислення MACD та його сигнал наступні.

  MACD = [0.075*EMA of Closing prices] – [0.15*EMA of closing prices] Signal Line = 0.2*EMA of MACD

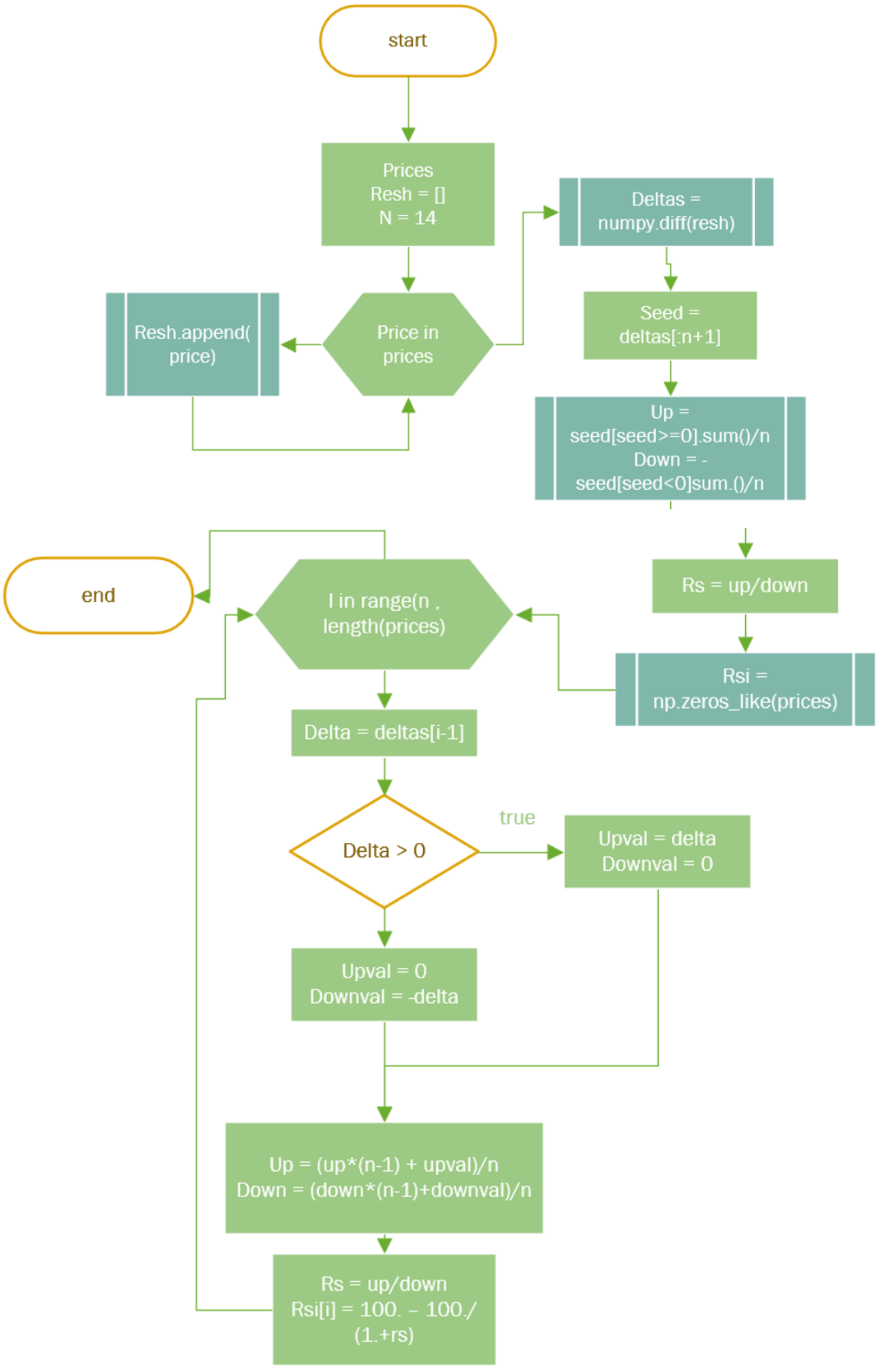


Рис. 4.1. Блок-схема алгоритму розрахунку RSI

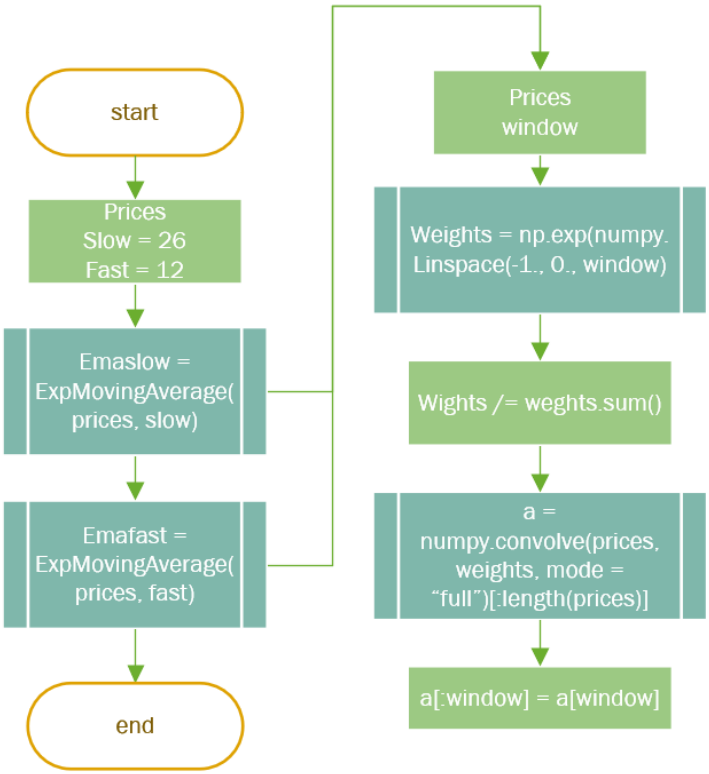


Рис. 4.2. Блок-схема алгоритму розрахунку MACD

8. Створюємо моделі рекурентного машинного навчання на основі моделей : 6-вузлової одношарової моделі, 32 вузлової одношарова моделі, 32 вузлової багатошарової моделі, багатошарової GRU моделі, багатошарої GRU моделі з алгоритмом відсіву, рекурентного алгоритму Facebook – prophet.
9. Сворюємо модель машинного навчання на основі 2х шарів нейронів LSTM та третього шара з нейроном виходу з функцією активації ReLu (рис. 12)
10. Настроюєм(навчаєм) моделі на основі даних для тренування
11. Отримуєм передбачені дані на заданий період з розрахунків модель
12. Розраховуєм середньо квадратичну похибку
13. Будуємо графік руху ціни з розрахованою майбутньою ціною

4.2.2 Роз’яснення алгоритму

Історичні дані, використані для коду, це дані часових рядів за 2006–2017 рр. для вибраних акцій. Ці дані можна знайти на Kaggle. Метою є підготовка моделі на основі даних акцій з 2006 по 2016 рік, а потім використання цієї моделі для прогнозування цін на 2017 рік.

Нижче наведено приклад руху цін акції JPM. Для тренування було використано дані найвищої ціни дня. Дані позначені синім, це дані для тренування нейронної мережі, а дані позначені зеленим, це дані які потрібно передбачити.

Рис. 4.3. Графік руху ціни акції AAPL

Розглядаючи методи машинного навчання, перераховані у вступі, використовуючи sklearn, ці методи, як правило, розглядають змінні X та Y, де x можна сприймати як «причина» а Y як наслідок.

Однак у даних часових рядів ціна акції в кожному з попередніх рядків корисна для прогнозування цін, знайдених у наступних рядках.

Щоб скористатися перевагами взаємозв'язку між цінами акцій з часом, можуть бути використані моделі RNN та LSTM.

4.2.2.1 Попередня обробка даних

Для того, щоб почати працювати з історичними даними, спочатку потрібно виконати попередню обробку даних :

training_set = all_data[:'2016'].iloc[:,1:2].values

test_set = all_data['2017':].iloc[:,1:2].values

Потім потрібне масштабування даних за допомого MinMaxScaler:

sc = MinMaxScaler(feature_range=(0,1))

training_set_scaled = sc.fit_transform(training_set)

Детальний процес обробки початкових даних показаний в фрагменті 1

```
def prepareSets(self, all_data):

    training_set, test_set = self.createSets(all_data);

    training_set_scaled = self.scale(training_set);

    for i in range(60, 2768):
        self.X_train.append(training_set_scaled[i - 60:i, 0])
        self.Y_train.append(training_set_scaled[i, 0])

    self.X_train, self.Y_train = np.array(self.X_train), np.array(self.Y_train)

    self.X_train = np.reshape(self.X_train, (self.X_train.shape[0],
                                             self.X_train.shape[1], 1))

    total_data = pd.concat(
        (all_data["High"][:'2016'], all_data["High"]['2017':]), axis=0)
    inputs = total_data[len(total_data) - len(test_set) - 60:].values
    inputs = inputs.reshape(-1, 1)
    inputs = self.scaler.transform(inputs)

    for i in range(60, 311):
        self.X_test.append(inputs[i - 60:i, 0])

    self.X_test = np.array(self.X_test)
    self.X_test = np.reshape(self.X_test, (self.X_test.shape[0],
                                           self.X_test.shape[1], 1))
```

Фрагмент коду 1.

4.2.2.2. RNN, LSTM та GRU

Для початку потрібно розробити дійсно просту одношарову рекурентну мережу з лише невеликою кількістю вузлів. Нижче показана 6-вузлова одношарова модель.

```
def createSimpleRnnReduced(self, dm):

    model = Sequential()
    model.add(SimpleRNN(6))
    model.add(Dense(1))

    model.compile(optimizer='rmsprop', loss='mean_squared_error')

    model.fit(dm.X_train, dm.Y_train, epochs=100, batch_size=150)

    scaled_preds = model.predict(dm.X_test)
    test_preds = dm.scaler.inverse_transform(scaled_preds)

    accuracy = Accuracy()

    rms = accuracy.calcRMS(dm, test_preds);

    return test_preds, rms
```

Фрагмент коду 2.

Часто модель можна вдосконалити, додавши більше вузлів. Зазвичай це другий крок до вдосконалення моделі глибокого навчання. У наведеному нижче коді можна побачити, що додаткові вузли додані до 32 вузлів, але це всеодно одношарова модель.

```
def createSimpleRnn(self, dm):

    model = Sequential()
    model.add(SimpleRNN(32))
    model.add(Dense(1))

    model.compile(optimizer='rmsprop', loss='mean_squared_error')

    model.fit(dm.X_train, dm.Y_train, epochs=100, batch_size=150)

    scaled_preds = model.predict(dm.X_test)
    test_preds = dm.scaler.inverse_transform(scaled_preds)

    accuracy = Accuracy()

    rms = accuracy.calcRMS(dm, test_preds);

    return test_preds, rms
```

Фрагмент коду 3.

У побудові цих періодичних моделей часто складають кілька шарів разом, щоб поліпшити їх здатність прогнозувати. Моделі з меншою кількістю шарів, як правило, недооцінюють дані, тоді як кожен наступний шар наближає до ‘overfiting’.

«Набір повторюваних шарів - це класичний спосіб побудови більш потужних рекурентних мереж: наприклад, те, що в даний час забезпечує алгоритм Google Translate, - це стек із семи великих шарів LSTM» - Франсуа Шолле

Нижче наведений приклад такої мережі, з більшою кількістю шарів.

```
def createRnn(self, dm):

    model = Sequential()
    model.add(SimpleRNN(32, return_sequences=True))
    model.add(SimpleRNN(32, return_sequences=True))
    model.add(SimpleRNN(32, return_sequences=True))
    model.add(SimpleRNN(32))
    model.add(Dense(1))

    model.compile(optimizer='rmsprop', loss='mean_squared_error')

    model.fit(dm.X_train, dm.Y_train, epochs=100, batch_size=150)

    scaled_preds = model.predict(dm.X_test)
    test_preds = dm.scaler.inverse_transform(scaled_preds)

    accuracy = Accuracy()

    rms = accuracy.calcRMS(dm, test_preds);

    return test_preds, rms
```

Фрагмент коду 4.

Модель SimpleRNN, як правило, вважається корисною лише тоді, коли остання точка даних містить необхідну інформацію для прогнозування наступної точки даних.

У більшості випадків використання рівнів LSTM або GRU перевершує SimpleRNNlayers у цих багатошарових моделях, оскільки ці шари можуть використовувати більш довгострокові залежності, вирішуючи те, що відоме як проблема зникаючого градієнта. Ці більш довершені шари краще зберігають застарілу інформацію для використання у майбутніх прогнозах.

На практиці шари GRU та LSTM часто досягають подібних результатів, хоча шари LSTM можуть трохи перевершити. Однак шари GRU, як правило, є кращими, оскільки вони менш обчислювальні. Нижче наведено приклад GRU моделі:

```
def createGRU(self, dm):

    regressorGRU = Sequential()
    regressorGRU.add(GRU(units=50, return_sequences=True,
                         input_shape=(dm.X_train.shape[1], 1), activation='tanh'))
    regressorGRU.add(GRU(units=50, return_sequences=True, activation='tanh'))
    regressorGRU.add(GRU(units=50, return_sequences=True, activation='tanh'))
    regressorGRU.add(GRU(units=50, activation='tanh'))
    regressorGRU.add(Dense(units=1))

    regressorGRU.compile(
        optimizer=SGD(
            lr=0.01,
            decay=1e-7,
            momentum=0.9,
            nesterov=False),
        loss='mean_squared_error')
    regressorGRU.fit(dm.X_train, dm.Y_train, epochs=50, batch_size=150)

    GRU_predicted_stock_price = regressorGRU.predict(dm.X_test)
    GRU_predicted_stock_price = dm.scaler.inverse_transform(GRU_predicted_stock_price)

    accuracy = Accuracy()

    rms = accuracy.calcRMS(dm, GRU_predicted_stock_price);

    return GRU_predicted_stock_price, rms
```

Фрагмент коду 5.

Складні багатошарові нейронні мережі мають тенденцію до “overfiting”. Одним із загальноприйнятих методів боротьби із “overfiting”є додавання відсіву. Відсів - це метод випадкового скидання ваг у різних частинах мережі, щоб зменшити можливість мережі переобладнюватись.

```
self.model = Sequential()
self.model.add(LSTM(units=50, return_sequences=True, input_shape=(dm.X_train.shape[1],1)))
self.model.add(LSTM(units=50))
self.model.add(Dense(1, activation='relu'))

self.model.compile(loss='mean_squared_error', optimizer='adam')
self.model.fit(dm.X_train, dm.Y_train, epochs=1, batch_size=1, verbose=2)
```

Фрагмент коду 6. (модель LSTM)

Рекомендованою стратегією використання відсіву в LSTM є використання лише в щільних шарах, що слідують за шарами LSTM. Хоча не рекомендується відсівання в шарах LSTM, оскільки під час підбору даних часових рядів кожен вузол може містити інформацію, яку не бажано скидати. У багатьох випадках випадання будь-якої потужності не призводить до вдосконалення моделей часових рядів.

В фрагменті 6, наведено приклад GRU моделі з алгоритмом відсіву.

```
def createGruWithDropOut(self, dm):

    regressorGRU = Sequential()
    regressorGRU.add(GRU(units=50, return_sequences=True,
                         input_shape=(dm.X_train.shape[1], 1), activation='tanh'))
    regressorGRU.add(Dropout(0.2))

    regressorGRU.add(GRU(units=50, return_sequences=True, activation='tanh'))
    regressorGRU.add(Dropout(0.2))

    regressorGRU.add(GRU(units=50, return_sequences=True, activation='tanh'))
    regressorGRU.add(Dropout(0.2))

    regressorGRU.add(GRU(units=50, activation='tanh'))
    regressorGRU.add(Dropout(0.2))

    regressorGRU.add(Dense(units=1))

    regressorGRU.compile( optimizer=SGD( lr=0.01, decay=1e-7, momentum=0.9,
                                         nesterov=False), loss='mean_squared_error')
    regressorGRU.fit(dm.X_train, dm.Y_train, epochs=50, batch_size=150)

    GRU_predicted_stock_price = regressorGRU.predict(dm.X_test)
    GRU_predicted_stock_price = dm.scaler.inverse_transform(GRU_predicted_stock_price)

    accuracy = Accuracy()

    rms = accuracy.calcRMS(dm, GRU_predicted_stock_price);

    return GRU_predicted_stock_price, rms
```

Фрагмент коду 7.

В якості остаточної моделі буде запропонована модель пророка Facebook, яка використовує ряд традиційних компонентів даних акцій.

У документації перелічені такі параметри:

• щогодинні, щоденні або щотижневі спостереження з історією щонайменше декількох місяців.с

• сильні множинні сезонні "людські масштаби": день тижня та час року

• важливі події (свята), які відбуваються через нерегулярні проміжки часу, які відомі заздалегідь.

• Кількість відсутніх спостережень або великих відхилень.

• історичні зміни тенденцій, наприклад, внаслідок випуску продукту або змін журналу.

• тенденції, що являють собою нелінійні криві зростання.

Нижче наведено приклад розробки такої моделі.

```
def createProphet(self, all_data,final_train_idx=2768, pred_periods=250):

    train_data = all_data[:final_train_idx].reset_index()[['Date', 'High']]
    train_data.columns = ['ds', 'y']

    prophet_model = Prophet()
    prophet_model.fit(train_data)

    test_dates = prophet_model.make_future_dataframe(periods=pred_periods)
    forecast_prices = prophet_model.predict(test_dates)

    return forecast_prices
```

Фрагмент коду 8.

4.3 Висновки до розділу

Тестування всіх алгоритмів були проведені неодноразово, на різних даних акцій. Після написання програми та отриманих результатів, досить складно визначитись, який алгоритм найбільш надійніший і може допомогти в проведенні інвестицій, так як не має універсального алгоритму, але можна точно назвати трьох лідерів в даному дослідженні. Враховуючи, що поведінка активів у всіх максимально різна, тому і зв'язки між алгоритмом, функцією активації, оптимізатором і не тільки, потрібно підбирати власноруч до кожного.

Нижче наведені результати роботи алгоритму. З того, що можна побачити, найочевидніше є те, що алгоритм від компанії facebook, не годиться для цієї задачі. Всі інші алгоритми справляються непогано, але лідерство займають алгоритми: адаптований lstm до біржових ринків, gru з відсіюванням та одношарова RNN з 32 вузлами. Проблема багатошарової моделі RNN, заключається якраз в зникаючому

градієнті. Так, як дані з шару в шар втрачають свою важливість, вона і показує гірший результат чим одношарова модель.

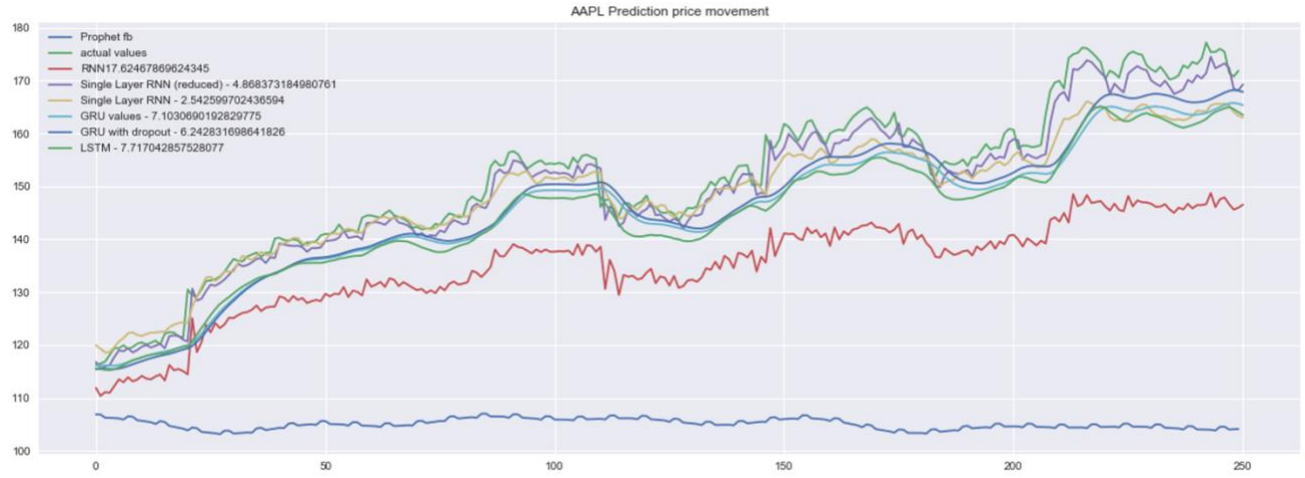


Рис. 4.4. Результат роботи запропонованої моделі

ВИСНОВКИ

Робота в сфері інвестицій дуже давній процес і є ще більш затребуваним в цей час, де все оцифровується. Час не стоїть на місці і все постійно покращується, так і розробку штучного інтелекту притягнули до роботи з біржами, для спрощення роботи з ними. Звичайно для маніпуляцій з ціною, інвестору потрібно бути впевненому по якому сценарію піде ціна, для такого повинен був бути проведений аналіз активу. Аналіз активу проводиться здебільшого на основі рівнів підтримки та опори, обємім, різних індекатор та повендінкового фактору. Ці всі методи використовується дуже багато років, і до сих пір є актуальними, але метою цієї роботи є добавлення до цих методів – аналіз на основі РНМ.

Безліч багатообіцяючих хедж-фондів в усьому світі вже давно використовують машинне навчання для алгоритмічної торгівлі, тому що це виключає будь-які прояви ірраціональних почуттів, таких як страх і жадібність. Таку модель торгівлі,

можна використовувати не тільки для того щоб замінити трейдерів, але й щоб використати змодельований аналіз для допомоги при аналізі загальновживаними методами.

Під час виконання даної роботи було дуже детально розглянуто процес, закономірності та механіку роботи бірж та поведінки цін. Було розглянуто процес створення нейронних мереж та їх адаптацію до ринків.

Виконаний аналіз показує перспективність застосування нейронних мереж в області дослідження таких аспектів бірж :

- Дослідження закономірностей поведінки ціни.
- Одержання передбаченої моделі майбутнього руху ціни.
- Спрощення аналізу історичних даних, через отримання узагальнених числових значень.

Під час виконання роботи було розроблено ПЗ, на основі алгоритмів RNN, LSTM, GRU та різних їх модефікацій. Тестування всіх алгоритмів були проведені неодноразово, на різних даних акцій. Після написання програми, та отриманих результатів, досить складно визначитись, який алгоритм найбільш надійніший і може допомогти в проведенні інвестицій. Нижче наведені результати роботи алгоритму. З того, що можна побачити, найочевидніше є те, що алгоритм від компанії facebook, не годиться для цієї задачі. Всі інші алгоритми справляються непогано, але лідерство займають алгоритми: адаптований lstm до біржових ринків, gru з відсіювання та одношарова RNN з 32 вузлами.

Зараз очевидно те, що не існує універсального шляху вирішення проблеми при розв’язуванні певної задачі за допомогою РНМ. Вибір архітектури рекурентної нейронної мережі та методів її навчання залежить від обсягу та характеру вирішуваної прикладної задачі.

Розроблену модель, можна використовувати в повсякденному аналізі цін акцій, так як показані результати є більш чим задовільні і на основі них можна дійти

до висновку, що рекурентні нейронні мережі в роботі з біржовими ринками є дуже корисним інструментом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://www.investopedia.com/terms/q/quantfund.asp#ixzz4YszizhII

2. https://www.bloomberg.com/news/articles/2017-02-06/silicon-valley-hedge-fund-takes-on-wall-street-with-ai-trader

3. https://www.nanalyze.com/2017/02/artificial-intelligence-stock-trading/

4. https://www.theguardian.com/technology/2016/dec/22/bridgewater-associates-ai-artificial-intelligence-management

5. https://www.forbes.com/sites/forbestechcouncil/2017/04/17/its-time-to-embrace-ais-superior-prediction-powers/?sh=7df92e6b68dd

6. https://nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf

7. https://www.tensorflow.org/

8. https://habr.com/ru/company/ods/blog/323272/

9. https://gluon.mxnet.io/

10. https://docs.microsoft.com/en-us/cognitive-toolkit/

ДОДАТКИ

Додаток 1

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Передбачення руху цін акцій за допомогою рекурентних нейронних мереж

Software Architecture Document (SAD)

Виконав: Устінський Андрій Андрійович

Зміст

Список діаграм

Список таблиць

1. Вступ

1.1 Мета

Метою є розробка ПЗ на основі рекурентних нейронних мереж який буде передбачувати руху ціни на біржі, за допомогою історичних даних. Модель повинна опрацювати та провести аналіз будь-якого активу, акції, індекса, валюти, тощо. Визначення самого оптимального алгоритму, який найкраще може справитися з поставленою задачею – передбачення руху цін акцій.

1.2 Сфера застосування

На даний час на біржах торгують мільйони трейдерів, які кожен день створюють рівновагу на ринку. Всі вони повністю різні, у кожного своя стратегія і вид робити на біржі, але всіх їх об’єднює одна дуже важлива річ : отримувати

завжди якомога більше прибутку з ринку. Один з варіантів, який може їм допомогти в цей час, це аналіз активів за допомогою штучного інтелекту.

Так як, алгоритмів штучного інтелекту, існує велика кількість, актуальністю цієї робити, є самостійно розробити всі можливо робочі варіанти рекурентної нейронної мережі, так як цей вид нейронних мереж найбільш підходящий до поставленої задачі та отримати най працездатніший алгоритм.

1.3 Визначення, скорочення та абревіатури

1. UML - Уніфікована мова моделювання
2. PHM – рекурентна нейронна мережа.
3. LSTM - long short-term memory
4. GRU - Вентильні рекурентні вузлии
5. RSI - Індекс відносної сили
6. MACD - сходження / розбіжність ковзних середніх
7. UI/UX - User interface / User Experience

1.4 Посилання

1. Технічний аналіз - Джек Д. Швагер.
2. Один хороший трейд - Майк Беллафіоре
3. Искусственный интеллект с примерами на Python - Пратик Джоши
4. http://www.codeforge.com/s/0/pso-lssvm
5. https://habr.com/ru/company/iticapital/blog/274821/
6. https://metro.co.uk/2019/05/06/can-we-trust-machines-to-predict-the-stock market-with-100-accuracy-9325480/
7. http://www.ai-machine-learning.com/artificialintelligencestockmarket.php https://conceptosclaros.com/que-es-regresion-logistica/
8. https://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html
9. https://tproger.ru/translations/6-step-for-building-machine-learning-projects/
10. https://proglib.io/p/ml-3months

11. https://towardsdatascience.com/neural-networks-to-predict-the-market-c4861b649371
12. https://www.rankiapro.com/inteligencia-artificial-oportunidad-inversion/
13. https://habr.com/ru/company/iticapital/blog/330884/
14. https://habr.com/ru/company/iticapital/blog/274821/
15. https://www.quandl.com/
16. https://finance.yahoo.com/
17. https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
18. https://www.sciencedirect.com/topics/engineering/recurrent-neural-network
19. https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce
20. https://www.ibm.com/cloud/learn/recurrent-neural-networks
21. http://irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/metrolog_2016_77_11.pdf
22. Омату Сигеру, Халид Марзуки, Юсоф Рубия Нейроуправление и его приложения. – М.: ИПРЖР, 2001. – 272 с.
23. Нейронные сети в системах автоматизации / В.И. Архангельский, И.Н. Богаенко, Г.Г. Грабовский и др. – Киев: Техника, 1999. – 363 с. 14. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2004. – 343 с.
24. http://journals.nupp.edu.ua/sunz/article/view/1214
25. http://www.immsp.kiev.ua/publications/articles/2009/2009_3/Reznik_03_2009.pdf

2. Архітектурне уявлення

Цей документ представляє архітектуру як ряд поглядів; використання подання випадку, логічного подання, подання процесу та подання розгортання. У цьому документі немає окремої точки зору реалізації. Це погляди на базову модель уніфікованої мови моделювання (UML), розроблену з використанням Rational Rose. За можливості, ми використовуємо існуючі технології замість того, щоб винаходити колесо. Працездатність систему в реальних ситуаціях це пріоритету №1.

3. Архітектурні цілі та обмеження

Досягнення мети включає розв’язання таких задач:

1) огляд існуючих концепцій лінійної алгебри;
2) аналіз існуючих алгоритмів аналізу даних та передбачення.
3) вибір релевантного алгоритму та обгрунтування доцільності його використання;
4) реалізація алгоритмів РНМ та їх адаптація до роботи з біржою.
5) Порівняти між собою та з власною адаптацією алгоритму LSTM.

Ціллю є реальний продукт, який зможе бути корисним в повсякденній роботі трейдера на біржі. Під корисністю мається наувазі - можливість збільшити прибутки.

Найважливішим є розробка моделі яка буде працювати локально і буде доступна тільки обмеженому колі користувачів, так як для користування нею потрібно буде деякі вміння в програмування, такі як встановлення потрібних для роботи бібліотек та запуск з середовища розробки.

В подальшому для більш широко розповсюдження, буде потрібно розділи ПЗ на клієнтську та серверну частину. Клієнтська частина буде максимально простою і зручною, а всі розрахунки будуть проводитися на потужному віддаленому сервері.

Основними обмеженнями є в обчислювальній здатносі, так як такі розрахунки дуже в ній потребуються та нестабільна ситуація на ринках активів, де дуже часто стаються непередбачувані ситуації.

4.Use-Case View

Use-Case View - опис виду використання архітектури програмного забезпечення. Перегляд випадків використання є важливим вкладом у вибір набору сценаріїв використання, які є фокусом ітерації. Він описує набір сценаріїв

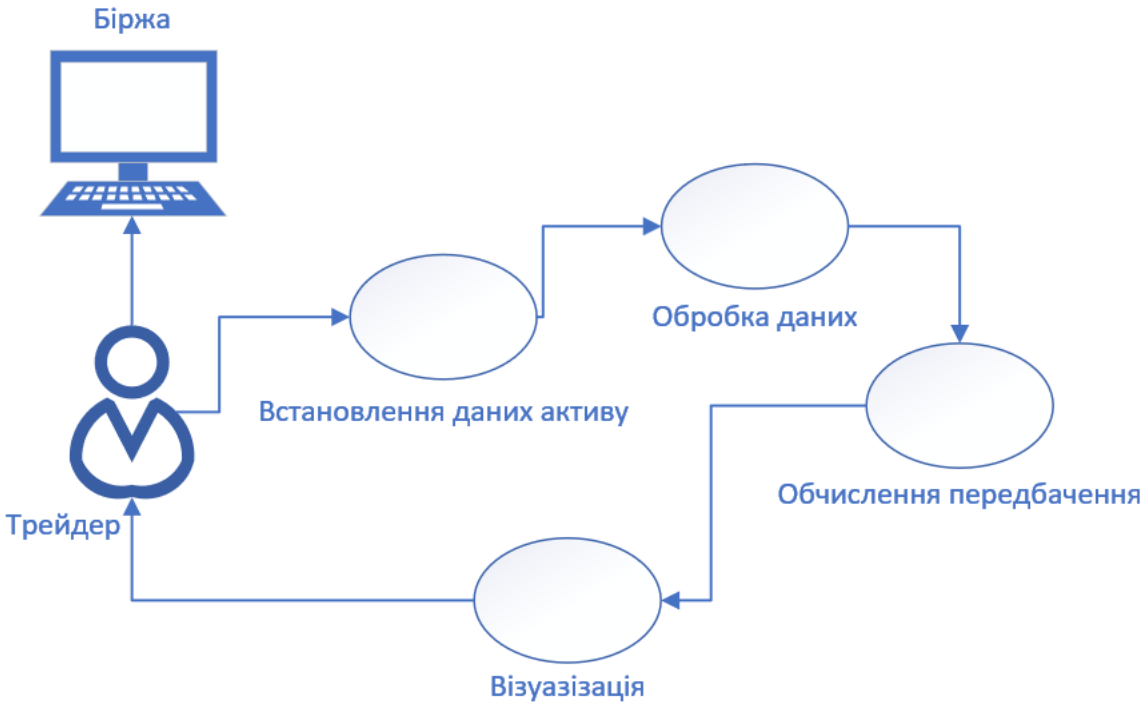
використання, які представляють деякі важливі центральні функції. Він також описує набір сценаріїв використання, які мають значне архітектурне охоплення (що реалізує багато архітектурних елементів) або які підкреслюють або ілюструють конкретний, тонкий момент архітектури.

Випадками використання представленого програмного забезпечення є:

- Встановлення даних активу
- Обробка даних
- Обчислення передбачення
- Візуалізація

Ці випадки використання ініціюються трейдером, після того як він зайшов на біржу, проаналізував актив, скачав дані активу та передав їх ПЗ для обчислення передбачення.

4.1 Архітектурно-значущі Use Cases



Діаграма 1. Архітектурно значущі випадки використання(Use Cases)

4.1.1 Встановлення даних активу

Короткий опис: Цей use case дозволяє трейдеру після того як отримав дані активу для навчання нейронної мережі, встановити їх в робочу директорію ./data/ з якої в подальшому розроблене ПЗ буде витягувати дані.

Date	Open	High	Low	Close	Volume	Name
2006-01-03	10.34	10.68	10.32	10.68	201853036	AAPL
2006-01-04	10.73	10.85	10.64	10.71	155225609	AAPL
2006-01-05	10.69	10.7	10.54	10.63	112396081	AAPL
2006-01-06	10.75	10.96	10.65	10.9	176139334	AAPL
2006-01-09	10.96	11.03	10.82	10.86	168861224	AAPL
2006-01-10	10.89	11.7	10.83	11.55	570088246	AAPL
2006-01-11	11.98	12.11	11.8	11.99	373548882	AAPL

Таблиця 1. Дані для тренування нейромережі активу AAPL

Дані подаються в такому вигляді : дата - ціна відкриття – вища ціна – нижча ціна – ціна закриття – проторгований об’єм – назва активу.

4.1.2 Обробка даних

Короткий опис: Цей use case дозволяє обробити дані для подальшої роботи рекурентної мережі. Програма вибирає задані користувачем дані для навчання, далі ділить їх та на: дані для навчання, тестування та передбачення, далі дані масштабуються.

4.1.3 Обчислення передбачення

Короткий опис: Цей use case дозволяє програмі використати дані для машинного навчання та отримати результати у вигляді передбачень за допомогою таких видів РНН : одношарова скорочена РНМ, одношарова РНМ, GRU модель, GRU модель з алгоритмом відсіву, LSTM модель, Prophet модель.

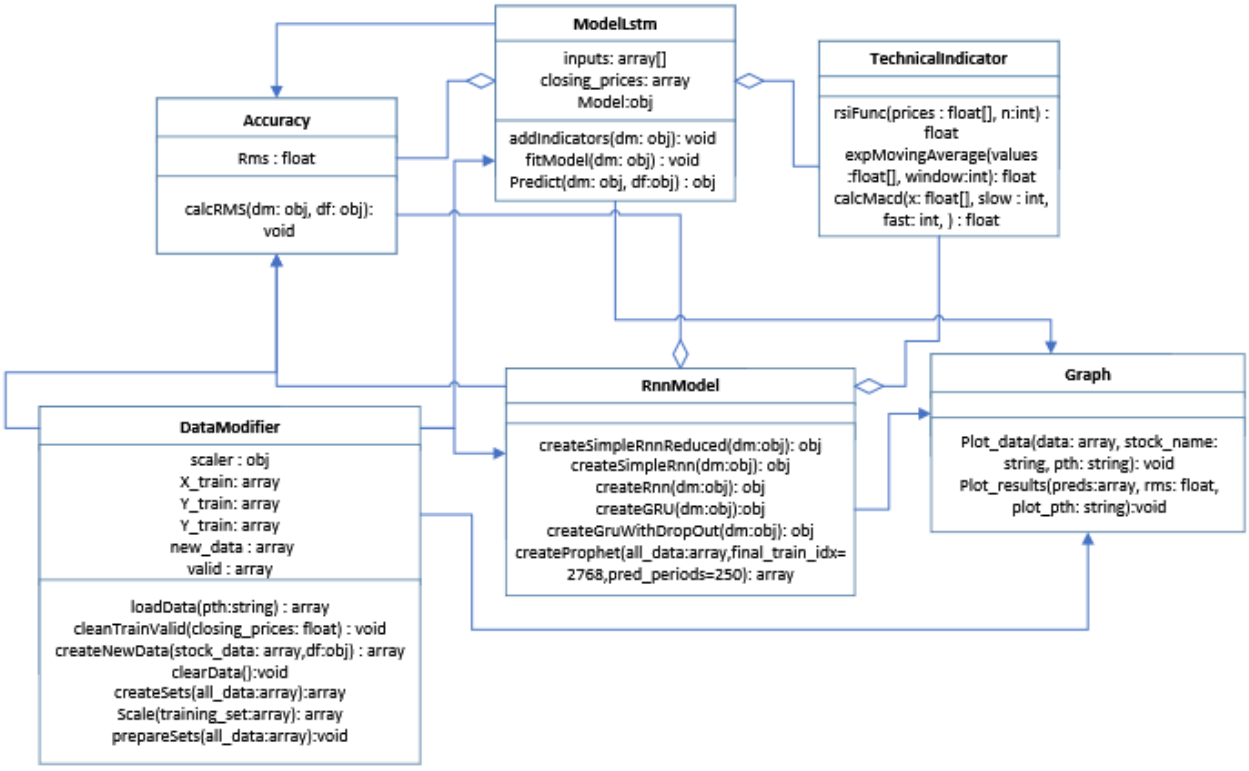
4.1.4 Візуалізація

Короткий опис: Цей use case дозволяє програмі подати результати у вигляді графіку.

5. Логічний погляд

Опис логічного погляду архітектури. Описуються найважливіші класи, їх організація в пакетах послуг та підсистемах, а також організація цих підсистем у рівні. Також описуються найважливіші варіанти реалізації, наприклад, динамічні аспекти архітектури. Діаграми класів можуть бути включені для ілюстрації взаємозв'язку між архітектурно значущими класами, підсистемами, пакетами та рівнями.

5.1 Діаграма класів програми



Діаграма 2. Діаграма класів застосунку.

Клас DataModifier:

Головний вузол для обробки даних

Метод loadData():

Метод для завантаження даних з директорії ./data/ в форматі csv для перетворення їх в робочу зміну.

Метод createNewData():

Метод для створення робочого масиву даних на основі вибраних даних.

Метод createSets():

Метод для створення масиву для тренування та тестування нейронної мережі.

Метод prepareSets():

Метод для масштабування робочих даних.

Метод Scale():

Метод який масштабує вхідні дані.

Метод clearData():

Так, як під час тренування всіх моделей програма використовує один об’єкт класу DataModifier, після кожної ітерації, попередні дані потрібно обнуляти.

Клас ModelLstm:

Клас для роботи з рекурентною моделлю LSTM (long short-term memory).

Метод addIndicators():

Метод для добавлення даних індекаторів RSI та MACD в робочий масив даних.

Метод fitModel():

Метод для тренування моделі LSTM

Метод predict():

Метод для передбачення руху цін на основі тренованої моделі.

Клас RnnModel:

Клас для запуску розроблених мелей РНН.

Метод createSimpleRnnReduced():

Метод для роботи з одношаровою скороченою РНН.

Метод createSimpleRnn():

Метод для роботи з одношаровою РНН.

Метод createRnn():

Метод для роботи з повною РНН.

Метод createGRU():

Метод для роботи з моделлю GRU.

Метод createGruWithDropOut():

Метод для роботи з моделею GRU з алгоритмом відсіву.

Метод createProphet():

Метод для роботи з моделлю Prophet.

Клас TechnicalIndicator:

Клас для роботи з технічними індекаторами.

Метод rsiFunc():

Метод для розрахунку індекатора RSI.

Метод expMovingAverage():

Метод для розрахунку індекатора EMA.

Метод calcMacd():

Метод для розрахунку індекатора MACD, який використовує для розрахунку EMA.

Клас Accuracy:

Клас для визначення тоності роботи алогоритму.

Метод calcRMS():

Метод для розрахунку середнього хвадратичного відхилення алгоритму.

Клас Graph:

Клас призначений для виведення результатів в графічні формі.

Метод Plot_data():

Метод, який відповідає за відображення руху ціни актива (дані які використовуються для тренування).

Метод Plot_results():

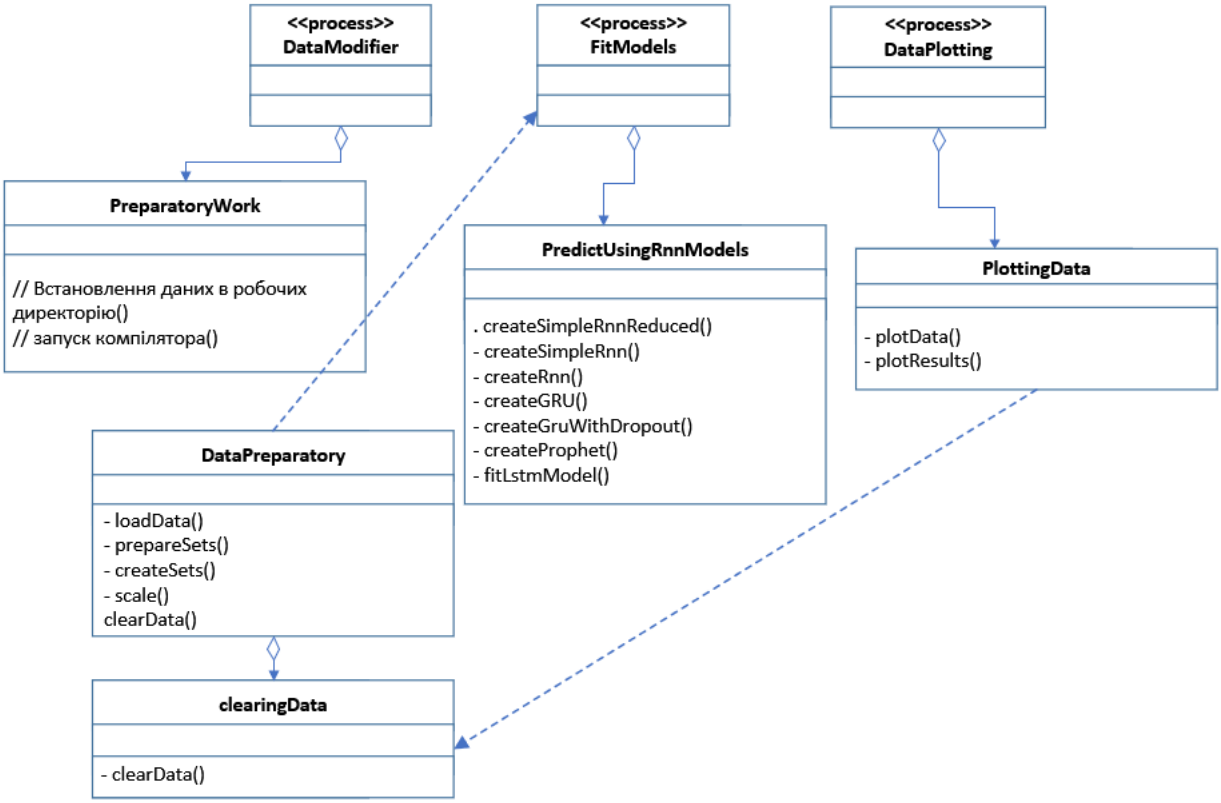
Метод, який відповідає за відображення результатів всіх алгоритмів та даних які потрібно було прописати.

6. Вид процесу

Опис процесу подання архітектури. Описує завдання (процеси та потоки), що беруть участь у виконанні системи, їх взаємодію та конфігурацію. Також описується розподіл об’єктів і класів на завдання.

Модель процесу ілюструє як проходять процеси від обробка даних, до тренування мереж та чистки змінних.

6.1 Processes



Діаграма 3. Діаграма процесів.

Основний процес програми починається з вузла '__main__' де створюється екземпляр класу dataModifier. Дальше вся програма працює в циклі, який завершується тоді коли закінчуються активи для аналізу в робочі директорії. Кожна ітерація проходить через такі процеси: PreparatoryWork -> DataPreparatory -> PredictUsingRnnModels -> PlottingData -> clearingData.

7. **Deployment View** (Перегляд розгортання)

Опис виду розгортання архітектури описує різні фізичні вузли для найбільш типових конфігурацій платформи. Також описується розподіл завдань (з подання процесу) на фізичні вузли.

7.1 Настільний ПК

Настілький ПК або ноутбук, це основні інструменти трейдера. В даній версії ПЗ, воно знаходиться у кожного трейдера локально і локально вони з ним інтерактують.

7.2 Сервер

На даній стадії ПЗ вона повністю націлина на локальну роботу. Але так як робота з нейронними мережами є доволі трудозатратним, потрібно немало обчислювальної здатності. Тому в подальших версіях потрібно перенести ПЗ на віддалений сервер з потужною обчислювальною здатністю, що надасть більшу швидкість при проведені розрахунків та можливість працювати більшому колу трейдерів.

7.3 Система виставлення рахунків

Поки ПЗ знаходиться в бета версії для обмеженого кола користувачів, воно є бесплатним, але з коли програмне забезпечення буде готовим, потрібно буле розробити система виставлення рахунків, яка буде генерувати рахунки відносно кількості використання.

8. Розмір та продуктивність

Обрана архітектура програмного забезпечення повинна підтримує ключові вимоги до розміру та часу.

Система повинна підтримувати до1000 одночасних користувачів які будуть робити запити на отримання бажаних даних. Робота з сервером буде обмеженна в запитах, тобто трейдер може робити запити не частіши ніж в певну одиниці часу. Таке обмеження є обов’язковим для того щоб система підтримувала більшу кількість користувачів та не була перегруженою.

Система повинна мати можливість виконати 80% усіх транзакцій протягом 10 хвилин (в звязку з обчислювальною складністю роботи нейронних мереж).
Клієнтська частина вимагатиме менше 50 Мб дискового простору та 64 Мб оперативної пам'яті.

Вибрана архітектура повинна підтримувати вимоги до розміру та часу шляхом реалізації архітектури клієнт-сервер. Клієнтська частина реалізована на локальних ПК на робочих місцях трейдера. Компоненти повинні бути розроблені для того, щоб забезпечити мінімальні вимоги до диска та пам'яті для клієнтської частини ПК.

9. Якість

Архітектура програмного забезпечення підтримує вимоги до якості.Інтерфейс робочого столу повинен відповідати всім канонам сучасного UI/UX дизайну.Коли симтема буде працювати онлайн, вона буде доступною в заложносі від відкриття робочих сесій на біржових ринках, 5 днів на тиждень. Час простою не повинен перевищувати 4%.

Кожна функція системи повинна мати вбудовану онлайн-довідку для користувача. Інтернет-довідка повинна містити покрокові інструкції щодо використання Системи. Інтернет-довідка повинна містити визначення термінів та скорочень.