НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп'ютерні науки
(код і назва)

на тему: Застосування методики машинного навчання Q-Learning
_____
_____

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-31
(шифр групи)

Осіюк Микола Вікторович
(прізвище, ім'я, по батькові) (підпис)

Керівник ас., к.т.н. Свірін Павло Володимирович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний доцент к.е.н. Рощина Н. В.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Осіюк Микола Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи Застосування методики машинного навчання Q-Learning
_____
_____ ,
керівник роботи ас, к.т.н, Свірін Павло Володимирович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи

1. Постановка задачі, розгляд її актуальності

2. Дослідження предметної області

3.Аналіз сучасних прикладів застосування Q-Learning

4. Аналіз власних прикладів застосування Q-Learning
5. Зробити висновки щодо отриманих результатів

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

_____

_____

_____

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Осіюка Миколи Вікторовича на тему

«Застосування методики машинного навчання Q-Learning»

Дана дипломна робота була присвячена дослідженню прикладів застосування методики машинного навчання Q-Learning та її різновидів.

В роботі було розглянуто загальні відомості про машинне навчання, розглянуто основні складові методики Q-Learning. Було виконано аналіз сучасного використання модифікацій Q-Learning: Fuzzy Q-learning в хмарних контролерах і Deep Q-Learning бот для Atari 2600.

Також була створена тестова реалізація алгоритму яку протестували на 2 задачах: гра у Flappy bird і покупка/продаж bitcoin.

Загальний обсяг роботи: 70 сторінок, 26 зображень, 8 таблиць, 17 посилань

Ключові слова: машинне навчання, Q-Learning, агент, середовище, навчання із підкріпленням.

АНОТАЦИЯ

бакалаврской дипломной работы Осиюка Николая Викторовича на тему «Применение методики машинного обучения Q-Learning»

Данная дипломная работа была посвящена исследованию примеров применения методики машинного обучения Q-Learning и ее разновидностей.

В работе были рассмотрены общие сведения о машинном обучении, рассмотрены основные составляющие методики Q-Learning. Был выполнен анализ современного использования модификаций Q-Learning: Fuzzy Q-Learning в облачных контроллерах и Deep Q-Learning бот для Atari 2600.

Также была создана тестовая реализация алгоритма, которую протестировали на 2 задачах: игра в Flappy bird и покупка / продажа bitcoin.

Общий объем работы 70 страниц, 26 изображений, 8 таблиц, 17 ссылок.

Ключевые слова: машинное обучение, Q-Learning, агент, среда, обучение с подкреплением.

ABSTRACT

to the bachelor thesis by Osiiuk Mykola Viktorovych on “Applications for Q-Learning Algorithm in Machine Learning”

The paper considered general information about machine learning, the basic components of the methodology Q-Learning. The using of modern versions of Q-Learning was analyzed, such as Fuzzy Q-learning in a cloud controller and Deep Q-Learning bot for the Atari 2600.

Also created the test realization of algorithm. It was tested by 2 different tasks: the Flappy bird game and the buying/selling on bitcoin.

Includes: 70 pages, 26 illustrations, 8 tables and 17 references.

Keywords: machine learning, Q-Learning, agent, environment, reinforcement learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

QL – Q-Learning

RL – reinforce learning

DQN – Deep Q-Learning network

FQL – Fuzzy Q-Learning

ВСТУП

В сучасному суспільстві значну роль відіграють інформаційні технології. Фактично, в сьогоденні вони використовуються в будь-якій сфері людського життя. Поміж всіх усіх інформаційних технологій, в сьогоднішньому світі особливе місце займають технології машинного навчання.

Початком історії машинного навчання можна назвати працю Томаса Баєса “An Essay towards solving a Problem in the Doctrine of Chances”[1], яку опублікували через 2 роки після його смерті Річардом Прайсом.

Після цього були відкриті ланцюги Маркова і Метод найменших квадратів, але справжнім початком розвитку машинного навчання вважається 1950 рік. Саме тоді, у жовтневому журналі “Mind” Алан Тюрінг опублікував статтю “COMPUTING MACHINERY AND INTELLIGENCE”[2] в які запропонував машини, які можуть вчитися і ставати інтелектуальними. Це було передбаченням генетичних алгоритмів.

Перша нейромережева машина появилась в 1951 році під назвою SNARC “Stochastic neural analog reinforcement calculator”[3]

Згодом, в 1957 році, Френк Розенблат придумує персептрон, який був реалізований в 1960 році у вигляді нейрокомп’ютера Марк-1. Після цього відкриття дослідження пришвидшились. Були відкриті алгоритми “найближчого сусіда” і автоматичного розподілу. Так же Марвіном Мінським та Сеймуром Папертом в 1969 була написана книга “Perceptrons” в якій були описані обмеження персептронів і нейронних мереж.

В 80-х роках дослідження пожвавились. Саме ці роки були відкриті рекурентні нейронні мережі і алгоритм зворотного поширення помилки. Так же, в кінці цього десятиліття, а саме в 1989 році був винайдений алгоритм Q-Leaning, який значно покращив практичність навчання із підкріпленням.

І хоча QL був відкритий аж в 1989 році, він залишається актуальним і в сьогоденні. Прикладом цього є нова варіація алгоритму QL, створена Google DeepMind, яка була названа як Deep Q-learning або Deep reinforcement learning.

Робота складається з п’яти розділів:

- В першому розділі описуються завдання роботи та її мета. Також аналізується її актуальність.
- В другому розділі проводиться аналіз предметної області, описуються машинне навчання, Q-learning і його варіанти.
- В третьому розділі проводиться аналіз прикладів застосування модифікацій методики Q-learning
- В четвертому розділі було проведено тестування однієї реалізації QL алгоритму на двох задачах: гра в Flappy Bird і покупка/продаж bitcoin
- В п’ятому розділі проводиться функціонально-вартісний аналіз

1 РОЗГЛЯД ЗАДАЧІ

1.1 Задачі дипломної роботи

Задачами дипломної роботи є:

1. Дослідити та вивчити основні принципи роботи методики машинного навчання Q-Learning
2. Проаналізувати існуючі приклади застосування
3. Створити тестовий приклад реалізації методу QL
4. Розробити вхідний набір даних для оцінки методу
5. Зробити висновки щодо отриманих результатів

1.2 Ціль дипломної роботи

Основною метою дипломної роботи є аналіз прикладів застосування методики машинного навчання Q-Learning.

Машинне навчання (англ. Machine learning) – це підрозділ штучного інтелекту, в якому вивчають методи побудови алгоритми, здатні навчатися. Ці алгоритми навчаються за допомогою великої кількості даних, і після цього становляться здатні виконувати завдання. Фактично, в машинному навчанні розглядають узагальнені алгоритми, які опрацьовують дані без написання специфічного коду під кожне завдання.

Q-Learning – це метод машинного навчання який відноситься то методів навчання з підкріпленням (англ. reinforcement learning). В його основі лежить те, що алгоритм формує функцію корисності Q. В наступних ітераціях це дозволяє алгоритму вибирати дії, спираючись на попередній досвід із взаємодією з середовищем.

1.3 Актуальність

Машинне навчання появилося ще в середині XX століття а сам Q-Learning вперше було представлено в 1989 році. З того часу в цій області були і успіхи і застої. Але в останні роки пройшов великий скачок в розвитку машинного навчання. Все почалось із 2012 року, коли команда Google brain написала нейронну мережу, яка розпізнавала зображення котів із зображень, взятих з відеороликів YouTube[4]. В 2013 DeepMind опублікувала результати із створення НМ яка грала в ігри Atari. Як видно із результатів (Рис. 1.1) DQN в 22 із 49 іграх перевершила людський результат, а в 43 побила спеціалізовані алгоритми.

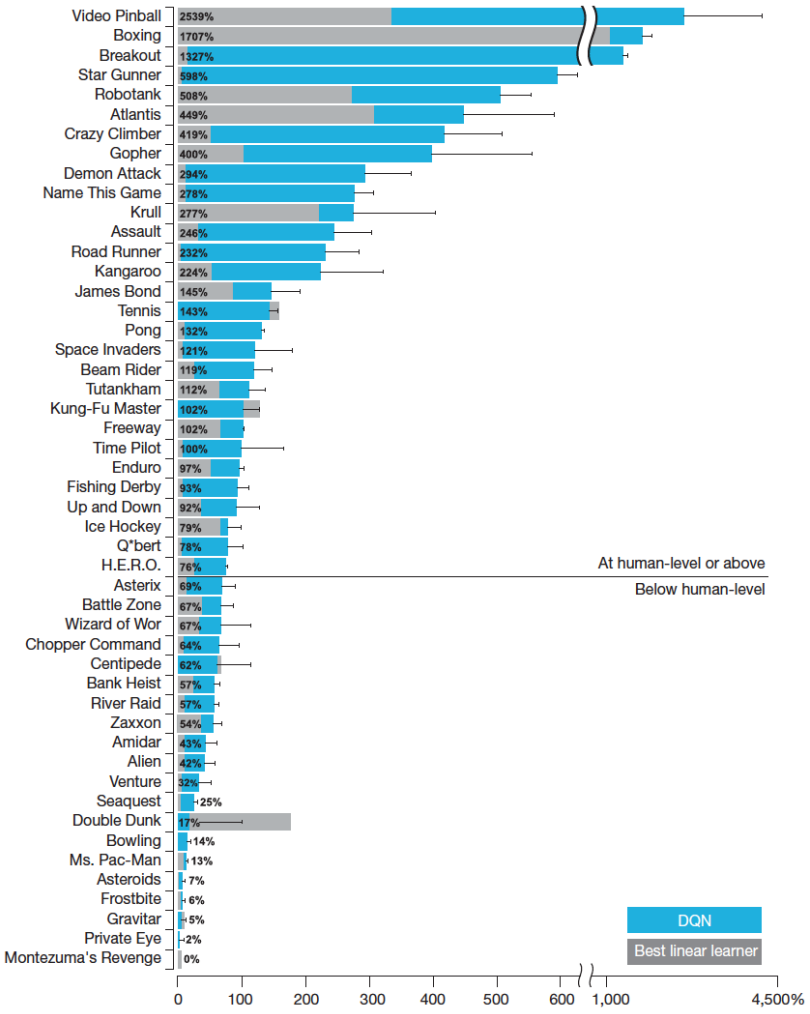


Рисунок 1.1 – Порівняння Deep Q-Learning агента із іншими методами [7]

Ще одною значною перемогою в машинному навчанні є AlphaGo. В 2015 року вона обіграла трикратного чемпіона Європи Фана Хуея[5], що досі вважалося неможливим. В березні 2016 року AlphaGo, з рахунком 4:1, перемогла Лі Седоля, одного із кращих гравців в Го [6].

На сьогодні машинне навчання тільки продовжує поширюватися і розвиватися. В 2015 році Google випустили TensorFlow[8] що дозволило практично будь-кому створювати свої нейронні мережі на Python, а в 2016 TensorFlow вийшов із бети і добавив експериментальні api для Java, Go і C.

1.4 Висновки

В даному розділі було сформульована ціль даної дипломної роботи та були приведені її цілі та короткий опис теми.

Також було доведено актуальність даної роботи на основі недавніх досягнень в даній області.

2 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Машинне навчання

Машинне навчання (Machine Learning) — обширний підрозділ штучного інтелекту, який вивчає методи побудови алгоритмів, здатних навчатися. Машинне навчання знаходиться на стику математичної статистики, методів оптимізації та класичних математичних дисциплін, але має і власну специфіку, пов'язану з проблемами обчислювальної ефективності та перенавчання. Багато методів індуктивного навчання розроблялися як альтернатива класичним статистичним підходам. Багато методів тісно пов'язані з вилученням інформації та інтелектуальним аналізом даних.

Найбільш теоретичні розділи машинного навчання об'єднані в окремий напрям, теорію обчислювального навчання.

Машинне навчання – не тільки математична, а й практична, інженерна дисципліна. Чиста теорія, як правило, не призводить відразу до методів і алгоритмів, які можуть застосовуватися на практиці. Щоб змусити їх добре працювати, доводиться винаходити додаткові механізми, що компенсують невідповідність зроблених в теорії припущень до умов реальних завдань. Практично жодне дослідження в машинному навчанні не обходиться без експерименту на модельних або реальних даних, що підтверджує практичну працездатність методу.

2.1.1 Навчання із учителем

У навчання з учителем (Supervised learning) ідея полягає в тому, що існує деяка залежність між відповідями і об'єктами, але вона невідома. Відома тільки кінцева сукупність прецедентів - пар «об'єкт, відповідь», яка названа навчальної вибіркою. На основі цих даних потрібно відновити залежність, тобто побудувати алгоритм, здатний для будь-якого об'єкта видати досить точну

відповідь. Для вимірювання точності відповідей певним чином вводиться функціонал якості.

Під учителем розуміється або сама навчальна вибірка, або той, хто вказав на заданих об'єктах правильні відповіді. (Рис. 2.1)

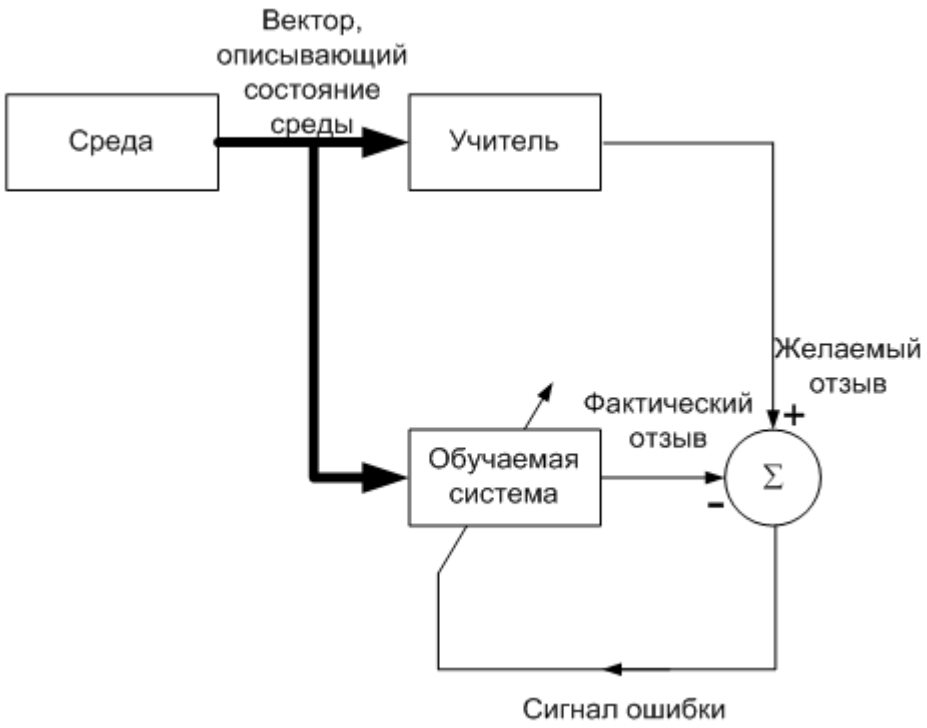


Рисунок 2.1 – Навчання із учителем[9]

Функціонал якості зазвичай визначається як середня помилка відповідей, виданих алгоритмом, по всіх об'єктах вибірки.

2.1.2 Навчання без вчителя

Навчання без вчителя (Unsupervised learning) вирішує задачі, у яких відомий тільки опис множини об’єктів. І потрібно знайти внутрішні взаємозв’язки, залежності які існують між об’єктами. Прикладами таких задач є Кластеризація і візуалізація даних.

2.1.3 Часткове навчання

Часткове навчання (semi-supervised learning) використовує при навчанні як розмічені, так і розділеного дані. Зазвичай використовується невелика кількість розмічених і значний обсяг нерозмічену даних. Часткове навчання є компромісом між навчанням без учителя (без будь-яких розмічених навчальних даних) і навчанням з учителем (з повністю розмічені набором навчання). Було відмічено, що якщо використовувати непомічені дані з невеликою кількістю розмічених то це може забезпечити значний приріст якості навчання. Під якістю навчання мається на увазі якийсь функціонал якості, наприклад, середньоквадратична помилка.

Збір розмічених даних для завдання навчання часто вимагає, щоб кваліфікований експерт вручну класифікував об'єкти навчання. Витрати, пов'язані з процесом розмітки, можуть бути настільки великими, що створення повного набору може бути неможливим, в той час як збір нерозмічених даних порівняно недорогий. У подібних ситуаціях цінність часткового навчання складно переоцінити.
Приклад прикладної задачі - автоматична рубрикація великої кількості питань за умови, що деякі з них вже віднесені до якихось рубрик.

2.1.5 Навчання із підкріпленням

Навчання із підкріпленням (reinforcement learning) вивчає, як агент повинен діяти в середовищі, щоб максимізувати деякий довготривалий виграш. Алгоритми з частковим навчанням намагаються знайти стратегію, зробити так, щоби в кожному стані навколишнього середовища агент вибирав найкращу дію. В економіці і теорії ігор навчання з підкріпленням розглядається в якості інтерпретації того, як може встановитися рівновага.

При навчанні з підкріпленням, на відміну від навчання з учителем, не надаються вірні пари "вхідні дані-відповідь", а прийняття майже оптимальних рішень (що дають локальний екстремум) не обмежується явно. Навчання з

підкріпленням намагається знайти компроміс між дослідженням невивчених областей і застосуванням наявних знань.

Формально найпростіша модель навчання з підкріпленням складається з:

1. безлічі станів оточення S;
2. безлічі дій A;
3. безлічі "виграшів".

Загалом навчання із підкріпленням працює таким чином (Рис. 2.2):

1. На агента приходить стан.
2. Агент вибирає дію
3. Середовище посилає агенту нагороду і наступний стан
4. Агент опрацьовує нагороду і коректує свої дії

2.2 Q-Learning

2.2.1 Вступ

Q-learning– метод застосовується в штучному інтелекті при агентному підході. Відноситься до навчання із підкріпленням (reinforcement learning). На основі отриманої від середовища винагороди агент формує функцію корисності Q, що надалі дає йому можливість уже не випадково вибирати стратегію поведінки, а враховувати досвід попереднього взаємодії. Одна з переваг Q-навчання - то, що воно в змозі порівняти очікувану корисність доступних дій, не формуючи моделі навколишнього середовища.

Використовується для ситуацій, які можна показати у вигляді марковського процесу прийняття рішень.

2.2.2 Задача про багаторукого бандита

Розглянемо наступну задачу навчання. Нехай доводиться багаторазово здійснювати вибір однієї з n різних альтернатив (варіантів дій). Кожен вибір

тягне за собою отримання певної винагороди. значення якого виходить з використанням деякого стаціонарного розподілу ймовірностей, що залежить від обраного дії. Метою послідовності дій є максимізація очікуваного повного винагороди за заданий період часу.

Це вихідна форма подання задачі про багаторукого бандита. На відміну від ігрового автомата або «однорукого бандита» ми розглянемо автомат, в якому n ручок замість однієї. Кожен вибір дії уподібнюється впливу на одну з ручок автомата в ході гри, а винагородами є виграші в даній грі. У серії ігор потрібно максимізувати виграш шляхом вибору кращих важелів.

У розглянутій задачі про багаторукого бандита кожній дії поставлено у відповідність очікувану або середню винагород, яка призначається при виборі цієї дії; будемо називати це цінністю даної дії. Якби цінність кожної дії була відома, рішення задачі про багаторукого бандита було б тривіальним: завжди слід було б вибирати дію з найвищою цінністю. Але точні значення невідомі, тому потрібна серія експериментів для визначення оцінки цих значень [11]

2.2.4 Агент і середовище

Елемент, який навчається і приймає рішення тут називається агентом. Сукупність усіх об'єктів, що знаходяться поза агентом, і з якими цей агент взаємодіє, позначається терміном середовище. Така взаємодія відбувається постійно: агент вибирає дії, навколишнє середовище реагує на них, створюючи нові ситуації для агента). Навколишнє середовище також є джерелом винагород (особливих чисельних значень, які агент постійно намагається збільшити). Повний опис характеристик навколишнього середовища визначає завдання. [11]

Агент і середовище взаємодіють на кожному із дискретних кроків, фактично середовище жде відповіді агента. На кожному кроці агент отримує стан середовища $s_t \in S$ де S – множина можливих станів і аналізуючи стан він вибирає одну із можливих дій $a_t \in A(s_t)$, де $A(s_t)$ – множина можливих дій для

стану s_t. На наступному кроці агент отримує винагороду r_{t+1}, як результат вибраної дії. (Рис 2.2)

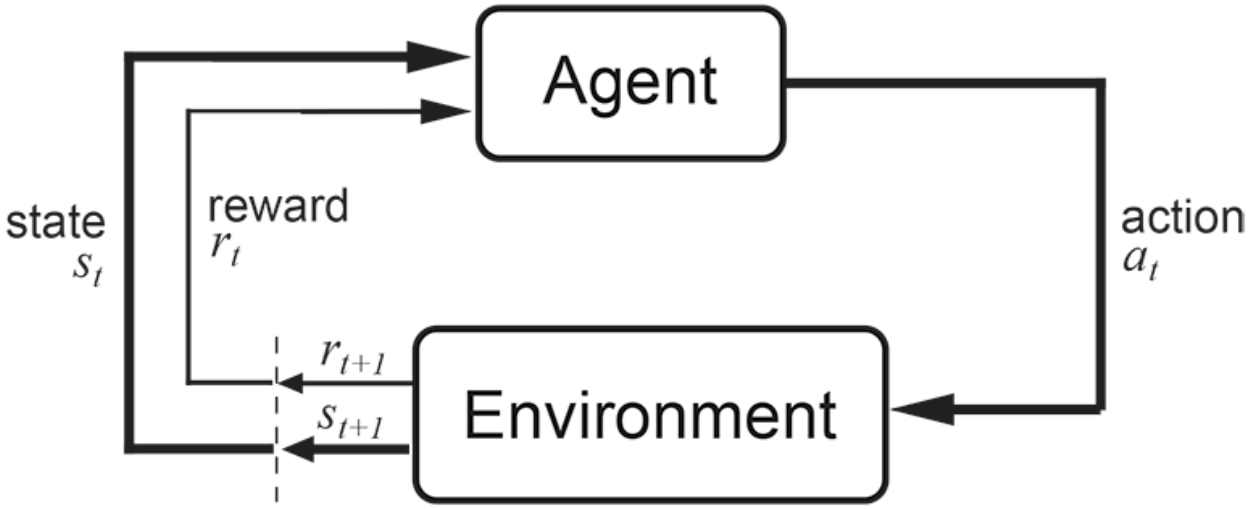


Рисунок 2.2 – Взаємодія агента і середовища [10]

Ця досить загальна і гнучка структура може бути використана для вирішення найрізноманітніших завдань різними способами. Наприклад, тимчасові кроки не обов'язково повинні відповідати фіксованим інтервалах реального часу, вони можуть відповідати довільним послідовним етапам прийняття рішень і виконання дій.

Строго кажучи, межа між агентом і навколишнім середовищем не завжди збігається з фізичною межею тіла робота або живого організму. Зазвичай ця межа знаходиться ближче до агента, ніж фізична межа його тіла. Наприклад, двигуни і сенсорні пристрої робота повинні розглядатися як об'єкти навколишнього середовища, а не як частини агента. Якщо розглядати з цієї точки зору м'язи, скелет і органи сприйняття також слід вважати частиною навколишнього середовища. Винагороди розраховуються зазвичай всередині живого тіла або штучної навчається системи, але також вважаються зовнішніми по відношенню до агента.

Існує правило: все, що не може довільно змінюватися агентом вважається навколишнім середовищем. Агент зазвичай досить багато знає про те, яким

чином розраховуються його винагороди і що є функціями від його дій і станів, в яких ці дії здійснюються. Однак процес розрахунку винагород завжди буде вважатися зовнішнім по відношенню до агента, оскільки цей процес визначає завдання агента, і з цієї причини агент не повинен мати можливість довільно змінювати його.

2.2.5 Цілі і винагороди

У навчанні з підкріпленням мета агента формалізується за допомогою спеціального сигналу винагороди, що надходить до агента з навколишнього середовища. На кожному часовому кроці t винагороду представляє гобой звичайне число $r_t \in R$. Мета агента максимізація сумарної винагороди. Це означає, що потрібно максимізувати НЕ винагороду на поточному кроці, а сукупна підсумкове винагороду за результатами досить тривалої послідовності кроків.

Використання сигналу винагороди для формалізації мети є найбільш характерною особливістю навчання з підкріпленням. Незважаючи на те що спочатку такий спосіб формалізації цілей може здатися обмеженим, він на практиці довів свою гнучкість.

Агент постійно вчиться максимізувати одержувану винагороду. Якщо ми хочемо, щоб він що-небудь робив для нас, ми повинні встановлювати винагороди таким чином, щоби при максимизуванні винагороди наші цілі здобувалися. Таким чином, необхідно, щоби встановлені нами винагороди правильно вказували чого ми хочемо досягти. Зокрема, сигнал винагороди не повинен наділяти агента апріорними знаннями про те, як досягти того, чого ми від нього хочемо. Наприклад, агент для грання в шахи повинен отримувати винагороди тільки за фактичні перемоги, але не за досягнення будь-яких проміжних цілей, таких як взяття фігур суперника або контроль центру дошки. Якби досягнення таких проміжних цілей винагороджувалося, агент міг би знайти спосіб їх задоволення, не досягаючи при цьому головну мету. Наприклад, він може знайти спосіб захоплювати фігури суперника навіть ціною

програшу гри. Сигнал винагороди є способом повідомити агенту, нього він повинен домагатися, але не як він повинен це зробити[11]

2.2.6 Марковська властивість

У навчанні з підкріпленням рішення, прийняті агентом, є функцію від сигналу з навколишнього середовища, який називається станом середовища. В ідеалі цей сигнал стану повинен коротко підсумувати минулі відчуття таким чином, щоб зберігалася вся необхідна інформація. Сигнал стану, що містить всю необхідну інформацію, називається марковським.

Розглянемо, яким чином середовище в момент $t+1$ може реагувати на дію, здійснену в момент t. В загальному варіанті реакція середовища залежить від будь-якої із попередніх подій. В цьому випадку динаміка середовища буде визначена тільки із заданням повного розподілу ймовірності(2.1)

$$\mathrm{P_r}\{s_{t+1} = s' , r_{t+1} = r \,|s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1} \ldots\ldots, r_1, s_0, a_0 \} \qquad (2.1)$$

Де s_i – стан середовища в момент і,

r_i – нагорода в момент і,

a_i – дія в момент і.

З іншого боку, якщо сигнал стану має марковську властивість, то реакція навколишнього середовища в момент часу s_{t+1} залежить лише від подання стану і дії в момент часу t. В такому випадку динаміку навколишнього середовища можна визначити. Знаючи тільки попередній стан (2.2)

$$\mathrm{P}_r\{s_{t+1} = s' , r_{t+1} = r \,|s_t, a_t\} \qquad (2.2)$$

Іншими словами, стан має марковскою властивістю і, отже, є марковським станом тоді і тільки тоді, коли(2.1) дорівнює (2.2). Якщо навколишнє середовище має марковскою властивістю, то її однокрокова динаміка (2.2) дозволяє передбачити наступний стан і наступну очікувану винагороду, яка є

результатом поточного стану і дії. Проводячи ітеративні обчислення для даного співвідношення є можливість передбачити всі майбутні стани і очікувані винагороди і всю передісторію станів, знаючи тільки поточний стан. Звідси випливає, що марковський стан забезпечує найкращий з усіх можливих базис для вибору дій. Таким чином, найкраща стратегія вибору дії, визначена як функція марковського стану і ефективна в тій же мірі, як і найкраща стратегія, сформована як функція всієї попередньої історії станів.

Навіть якщо сигнал стани не марковським, все ж доречно буде розглядати його в завданні навчання з підкріпленням як апроксимацію марковського стану. Зокрема, завжди бажано, щоб стан було хорошим базисом для передбачення подальших винагород і для вибору дій. Марковські стани є найкращим базисом для здійснення всіх цих цілей. Чим ближче характеристики даного стану будуть до характеристик марковських станів, тим краще буде проявляти себе система навчання з підкріпленням. Таким чином, на кожному часовому кроці стан вигідно розглядати як апроксимацію марковського стану, хоча необхідно пам'ятати, що воно може не повністю задовольняти марковській властивості.[11]

2.2.7 Марковський процес прийняття рішень

Завдання навчання з підкріпленням, яке задовольняє Марковскій властивості, називаються марковським процесом прийняття рішень

Марковський процес прийняття рішень це специфікація завдання послідовного прийняття рішень для повністю спостерігається середовища з марковською моделлю переходу і додатковими винагородами.

Цей процес характеризується наступними особливостями:

- Зовнішнє середовище розвивається на основі законів ймовірності, приймаючи скінченну множину дискретних станів. Але ці стани не враховують дані з попередніх статистик.

- В кожному стані існує множина можливих дій, які може виконати система
- При виконанні якоїсь дії знімається плата або видається винагорода.
- Спостережувані стани, дії, та плата/винагорода змінюються в дискретному часі.

2.2.8 Алгоритм Q-Learning

Найпростіша форма, однокроковий QL, визначається наступним чином:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, \alpha) - Q(s_t, a_t)] \qquad (2.3)$$

Де s_t – стан середовища в момент t, a_t – дія, яку виконав агент в під час стану s_t, r_t – нагорода після виконання агентом дії a_t

В цьому випадку шукана функція цінності дії Q, безпосередньо апроксимує Q*, оптимальну функцію цінності дії, незалежно від використаної стратегії. Це значно спрощує аналіз алгоритму і зберігає в силі запропоновані раніше докази збіжності.

Стратегія як і раніше визначає те, які пари стан дію відвідуються і коригуються. Проте для забезпечення збіжності необхідно лише, щоб всі пари продовжували коригуватися. Це є мінімальним вимогою в тому сенсі, що кожен метод, гарантовано знаходить оптимальну лінію поведінки, в загальному випадку повинен задовольняти цій умові. За такої умови і при такому варіанті звичайних умов стохастичною апроксимації для послідовності значень довжини кроку показано, що функція Q_t сходиться до Q* з імовірністю 1. Алгоритм QL в процедурній формі можна показати так:[11]

Ініціалізувати $Q(s, a)$ випадковими даними

Повторювати (для кожного епізоду)

Ініціалізувати s

Повторювати (для кожного кроку епізоду)

Знайти a по s, використовуючи стратегію отриману із Q

Виконати дію a, знайти r, s'

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, \alpha) - Q(s_t, a_t)]$

$s \leftarrow s'$

Поки s не стане кінцевим станом

2.3 Висновки

В даному розділі було розглянуте машинне навчання а також метод машинного навчання Q-Learning.

Було розглянуто 4 варіанта навчання нейронної мережі: навчання з учителем, навчання без учителя, часткове навчання і навчання із підкріпленням.

Також було детальніше розглянуто навчання із підкріпленням і Q-learning. Навчання з підкріпленням є навчанням на основі взаємодії того, якої поведінки потрібно дотримуватися для досягнення мети.

Агент навчання з підкріпленням і навколишнє його середовище взаємодіють протягом послідовності дискретних тимчасових кроків. Все, що знаходиться всередині агента, йому повністю відомо і підконтрольно: все, що знаходиться поза агента, не повністю підконтрольне, але може бути повністю відомо. Стратегія являє собою розподіл усіх правило, згідно з яким агент вибирає дію як функцію стану. Мета агента максимізувати сумарну винагороду, яке він отримає за деякий період часу.

Навколишнє середовище має марковську властивість, якщо її сигнал стану коротко резюмує минуле без погіршення здатності передбачати майбутнє.

3 СУЧАСНЕ ЗАСТОСУВАННЯ Q-LEARNING

3.1 Deep Q-Learning і Atari

3.1.1 Вступ

Компанія DeepMind створила систему штучного інтелекту, яка в більшу частину ігор Atari (Рисунок 3.1)грає краще за людину. Програма навчилася грати, не знаючи правил і не маючи доступу до коду, а просто спостерігаючи за картинкою на екрані.

Тренування DQN здійснив лондонський підрозділ Google DeepMind. Штучний інтелекту не повідомляли правила гри. Нейромережа сама аналізувала стан і шукала спосіб набрати максимальну кількість очок.

Ця розробка не така легковажна, як може здатися. Універсальна система яка може самостійно навчатися може знайти застосування в автономних автомобілях і інших проектах, де потрібно аналізувати стан навколишніх об'єктів і приймати рішення.

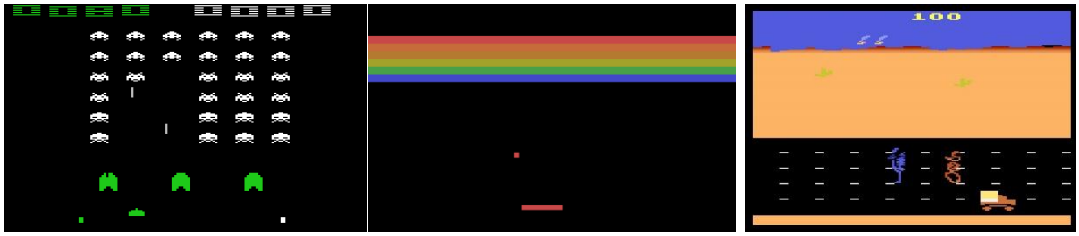


Рисунок 3.1 – Приклади ігор Atari. Зліва направо: Space Invaders, Breakout, Road Runner

3.1.2 Опис

Ігри Atari запускалися через емулятор. На кожному кроці агент вибирав одну із доступних дій, який визначався кнопками геймпада. Сам агент не отримував стану емулятора. Замість цього він отримував зображення з емулятора. В якості нагороди використовувався рахунок із гри.

Було вирішено що залежність станів від дій розглядати як кінцеву тобто певна дія впливає на кінцеву кількість станів. Це формалізувало задачу під марковський процес прийняття рішень, дозволяючи використовувати стандартні методики навчання із підкріпленням.

Q-Learning в цьому середовищі буде працювати але не ефективно із-за дуже великої кількості станів. Потрібно було зробити певне узагальнення: коли ми вивчаємо цінність певної пари дії/стан потрібно також покращувати знання про інші аналогічні дій/стани. Для цього було розроблено Deep Q Learning Network.

Алгоритм deep Q-learning використовує згорточну нейронну мережу (рис 3.2) як функцію апроксимації числа Q. Він, після деяких трансформацій, приймає картинку із емулятора на вхід. Алгоритм трансформує вхідні дані за допомогою пари шарів нелінійних функцій. На виході виходить 18-мірний вектор. Цей вектор виражає поточний стан і кожну із можливих дій. Дія вибирається серед найвищих Q.

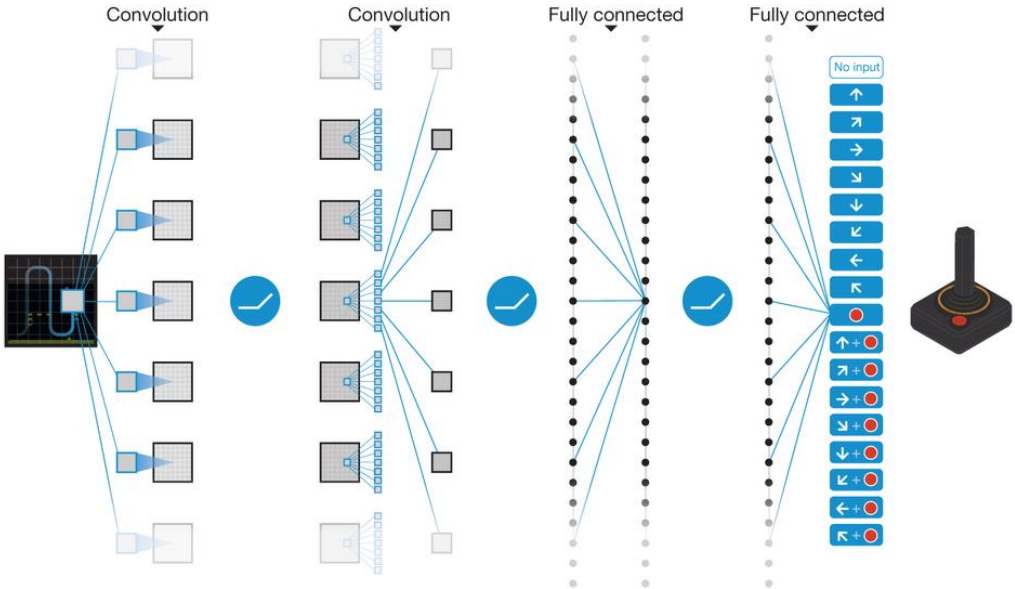


Рисунок 3.2 – Схематичні ілюстрація згорточної нейронної мережі[13]

3.1.3 Epsilon-greedy strategy

Спочатку, при тренуванні, цінність кожної дії невідома. І при першому проході рандомно вибрані дії можуть покращити свою цінність Якщо їх цінність буде більша за інші, неперевірені дії то, при жадібній стратегії, будуть вибиратися тільки вони, навіть якщо це не кращий вибір.

Рішенням цієї проблеми є використання стратегії epsilon-greedy під час тренування. В агента є незначний шанс ε вибрати випадкову дію замість вибору дії з кращим Q. Із-за цього кожна дія буде вибиратися не із-за Q чисел ініціалізації а із-за накоплення певного досвіду.

3.1.4 Frameskip

Atari 2600 використовувалася як зовнішній пристрій для телевізорів. Вона генерувала 60 кадрів в секунду. Але алгоритму DQN не було потрібно стільки багато кадрів. Навпаки обробка стількох кадрів сповільнювала тренування алгоритму. Тому експериментальним шляхом було вирішено використовувати тільки кожен 4 кадр для всіх ігор крім space invaders, де такий пропуск кадрів робив лазер невидимим. Тому там було вирішено використовувати кожен 3 кадр що є єдиною різницею між настройками для всіх ігор.[12]

3.1.5 Результати

В результаті DQN показала такі результати(таблиця 3.1):
В 43 із 49 вона перевершила результат спеціалізованих алгоритмів
В 22 із 49 вона перевершила найкращий результат людей. [12]

Таблиця 3.1 – Порівняння очок отриманих при методі DQN із людськими результатами і іншими методами навчання [14]

Game	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	939,20	103,20	6875,00	3069 (±1093)	42,70%
Amidar	103,40	183,60	1676,00	739,5 (±3024)	43,90%
Assault	628,00	537,00	1496,00	3359(±775)	246,20%
Asterix	987,30	1332,00	8503,00	6012 (±1744)	70,00%
Asteroids	907,30	89,00	13157,00	1629(±542)	7,30%
Atlantis	62687,00	852,90	29028,00	85641 (±17600)	449,90%
Bank Heist	190,80	67,40	734,40	429,7 (±650)	57,70%
Battle Zone	15820,00	42782,00	37800,00	26300 (±7725)	67,60%
Beam Rider	929,40	1743,00	5775,00	6846 (±1619)	119,80%
Bowling	43,90	36,40	154,80	42,4 (±88)	14,70%
Boxing	44,00	42956,00	42798,00	71,8 (±8,4)	1707,90%
Breakout	42771,00	42741,00	42978,00	401,2 (±26,9)	1327,20%
Centipede	8803,00	4647,00	11963,00	8309(±5237)	63,00%
Chopper Command	1582,00	42994,00	9882,00	6687 (±2916)	64,80%
Crazy Climber	23411,00	149,80	35411,00	114103 (±22797)	419,50%
Demon Attack	520,50	0,00	3401,00	9711 (±2406)	294,20%
Double Dunk	-13,10	-16,00	-15,50	-18,1 (±2,6)	17,10%
Enduro	129,10	159,40	309,60	301,8 (±24,6)	97,50%
Fishing Derby	-89,50	-85,10	42860,00	-0,8 (±19,0)	93,50%
Freeway	42754,00	42935,00	42915,00	30,3 (±0,7)	102,40%
Frostbite	216,90	180,90	4335,00	328,3 (±250,5)	6,20%
Gopher	1288,00	2368,00	2321,00	8520 (±3279)	400,40%
Gravitar	387,70	429,00	2672,00	306,7 (±223,9)	5,30%
H,E,R,O,	6459,00	7295,00	25763,00	19950 (±158)	76,50%
Ice Hockey	-9,50	-3,20	0,90	-1,6 (±2,5)	79,30%
James Bond	202,80	354,10	406,70	576,7 (±175,5)	145,00%
Kangaroo	1622,00	42955,00	3035,00	6740 (±2959)	224,20%
Krull	3372,00	3341,00	2395,00	3805 (±1033)	277,00%

Таблиця 2.1 (продовження)

1	2	3	4	5	6
Kung-Fu Master	19544,00	29151,00	22736,00	23270 (±5955)	102,40%
Montezuma's Revenge	42926,00	259,00	4367,00	0(±0)	0,00%
Ms, Pacman	1692,00	1227,00	15693,00	2311(±525)	13,00%
Name This Game	2500,00	2247,00	4076,00	7257 (±547)	278,30%
Pong	-19,00	-17,40	42803,00	18,9 (±1,3)	132,00%
Private Eye	684,30	86,00	69571,00	1788 (±5473)	2,50%
Q*Bert	613,50	960,30	13455,00	10596 (±3294)	78,50%
River Raid	1904,00	2650,00	13513,00	8316 (±1049)	57,30%
Road Runner	67,70	89,10	7845,00	18257 (±4268)	232,90%
Robotank	42944,00	42837,00	42989,00	51,6 (±4,7)	509,00%
Seaquest	664,80	675,50	20182,00	5286(±1310)	25,90%
Space Invaders	250,10	267,90	1652,00	1976 (±893)	121,50%
Star Gunner	1070,00	42834,00	10250,00	57997 (±3152)	598,10%
Tennis	-0,10	0,00	-8,90	-2,5 (±1,9)	143,20%
Time Pilot	3741,00	43002,00	5925,00	5947 (±1600)	100,90%
Tutankham	114,30	98,20	167,60	186,7 (±41,9)	112,20%
Up and Down	3533,00	2449,00	9082,00	8456 (±3162)	92,70%
Venture	66,00	0,60	1188,00	38Q0 (±238,6)	32,00%
Video Pinball	16871,00	19761,00	17298,00	42684 (±16287)	2539,40%
Wizard of Wor	1981,00	36,90	4757,00	3393 (±2019)	67,50%
Zaxxon	3365,00	42846,00	9173,00	4977 (±1235)	54,10%

3.2 Fuzzy Q-Learning і хмарні контролери

3.2.1 Вступ

Ціль хмарних контролерів реагувати на вимоги програми автоматично масштабуючи комп’ютерні ресурси під час виконання для вдоволення потреб продуктивності і мінімізації витрат ресурсів. Існуючі контролери часто видаються до масшабуючої стратегії, яка визначена набором правил

масштабування. Однак часто значна хмарної інфраструктури виглядає як чорні ящики, створюючи труднощі для побудови оптимальних правил.

Тому було запропоновано замінити правила на нейронну мережу, використовуючи Fuzzy Q-Learning, версії QL яка базується на нечіткій логіці. Названо її було FQL4KE[15]

3.2.2 Причини вибору алгоритму

Існує декілька характеристик, які дають виклик існуючим автомасштабуючим технікам і приладам:

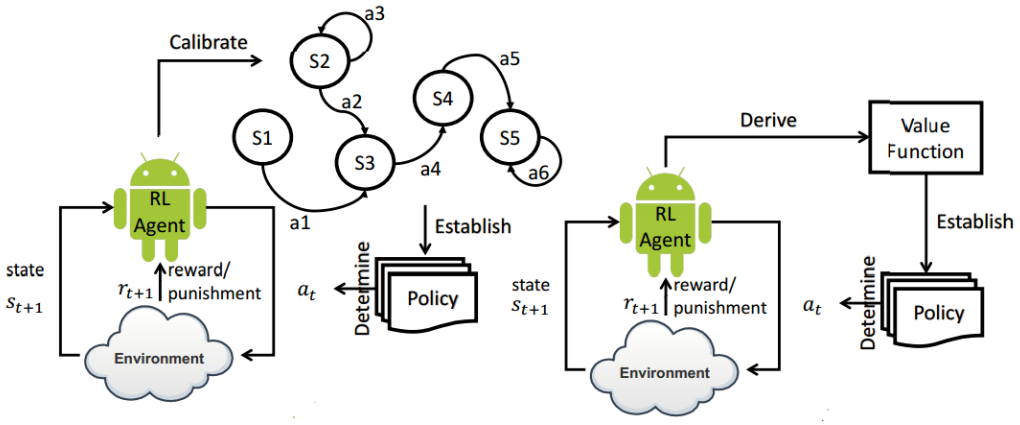
- Середовище не епізодичне, тобто наступні вибори впливають на наступні дії
- Хмарна інфраструктура дуже складна для моделювання.
- Робочі навантаження іррегулярні і динамічні.

Ці характеристики середовища заставляють вирішувати проблему послідовних рішень, де кожна дія поточного стану впливає на майбутні. Звичайний спосіб вирішення цієї проблеми полягає в створенні плану, правила або стратегії дій, які працюватимуть, доки не виникне специфічна ситуація.

Враховуючи характеристики середовища було вирішено використовувати нейронну мережу основану на навчанні із підкріпленням.

Агент вибирає дії, використовуючи правило, яке підвищить майбутню нагороду. RL основана на моделі (рис. 3.3 а) могла би допомогти із вибором рішення, але із-за складнощів моделювання хмарної інфраструктури в цієї RL велика складність використання.

Рішенням цієї проблеми є вибір без модельної версії RL(рис. 3.3 б). Вибір в користь QL був ще із-за того, що в нього оптимальна політика із отриманням негайної і відкладеної нагороди.



а) б)

Рисунок 3.3 – Варіанти RL нейронної мережі: а) основана на моделі; б) без модельна[15]

QL був вибраний ще із-за того, що із-за непередбачуваності навантаження на хмарні програму, що призводило до фактичної неможливості получити тренувальні дата сети. QL же в свою чергу не потребує дата сетів для навчання.

FQL був вибраний із-за проблеми неповного знання. Із-за значної комплексності і складності хмарного середовища не можливо було отримати повну інформацію, щоби правильно коригувати дії. FQL же дозволяє використовувати неповні знання. Нечіткі правила продовжують покращуватися під час збору даних в реальному часі.

3.2.3 FQL4KE

Поки програма працює на хмарній платформі , що спричиняє потрібність ресурсів FQL4KE (рис. 3.4) моніторить програму і керує розподілом ресурсів. Фактично FQL4KE веде постійний моніторинг, перевіряє задоволеність системних цілей і відповідно розподіляє ресурси у випадку відхилення від цілей.

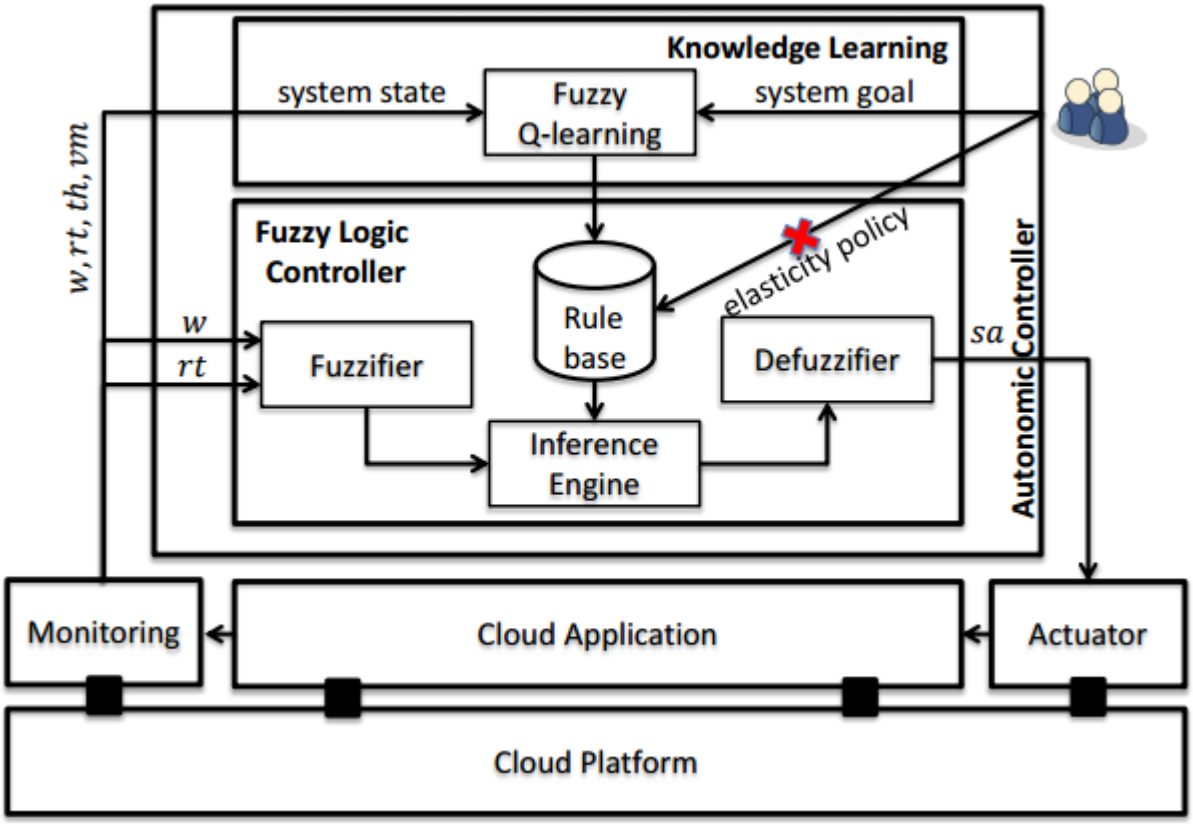


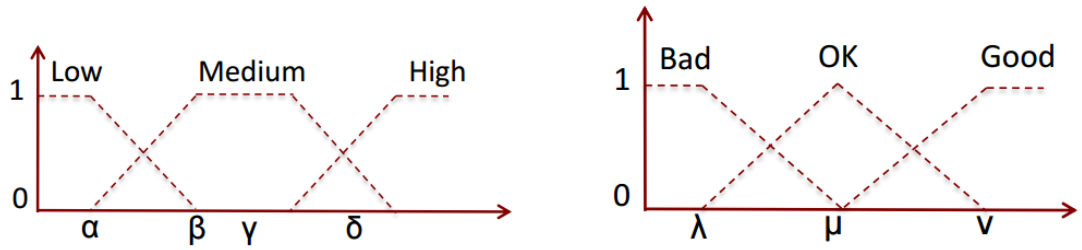
Рисунок 3.4 – Архітектура FQL4KE [15]

3.2.4 Логічний нечіткий контролер

Контролер співставляє набір вхідних даних із набором виходів управління за допомогою нечітких правил. Головна сфера використання нечітких контролерів це типи проблем, які не можуть бути представлені в явних математичних моделях із-за високої нелінійності системи. Потенціал нечітких контролерів лежить у можливості апроксимувати нелінійність, виражаючи знання в варіанті схожому на людське сприйняття і логіку.

У нечіткому контролері визначаються нечіткі сети і функції приналежності вхідних сигналів. Кожен нечіткий сет асоціюється із лінгвістичними термінами, наприклад “low”, “high”. Також визначається база правил, яка установлює поведінку контролера використовуючи вхідні дані.

Вхідними даними є завантаження (рис. 3.5 а) і Час відповіді (рис 3.5 б)



а) б)

Рисунок 3.5 – Вхідні дані: а) Завантаження; б) Час відповіді[15]

Контролер складається із фуззіфікатора, дефузіфікатора механізму логічного виводу і бази правил, з якою взаємодіє FQL(рис. 3.4).

Фуззіфікатор перетворює вхідні дані в нечіткі сети. Дефуззіфікатор, в свою чергу, перетворює відповідь, яка представлена у вигляді нечіткого сету, у дані, які сприймаються хмарною платформою.

3.2.5 Результати

Для тестування було використано було використано 6 згенерованих паттернів навантаження:

- Великий пік навантаження(рис. 3.5 а).
- Подвійна фаза (рис. 3.5 б).
- Великі варіації (рис. 3.5 в)
- Швидкі зміни (рис. 3.5 г)
- Повільна зміна (рис 3.5 д)
- Три фази з крутою зміною навантаження (рис. 3.5. е)

У FQL4KE використовувалася стратегія RobusT2Scale.Порівняння проходило із стандартним австомасштабуванням Azure. Було проведено 10 000 епох навчання.

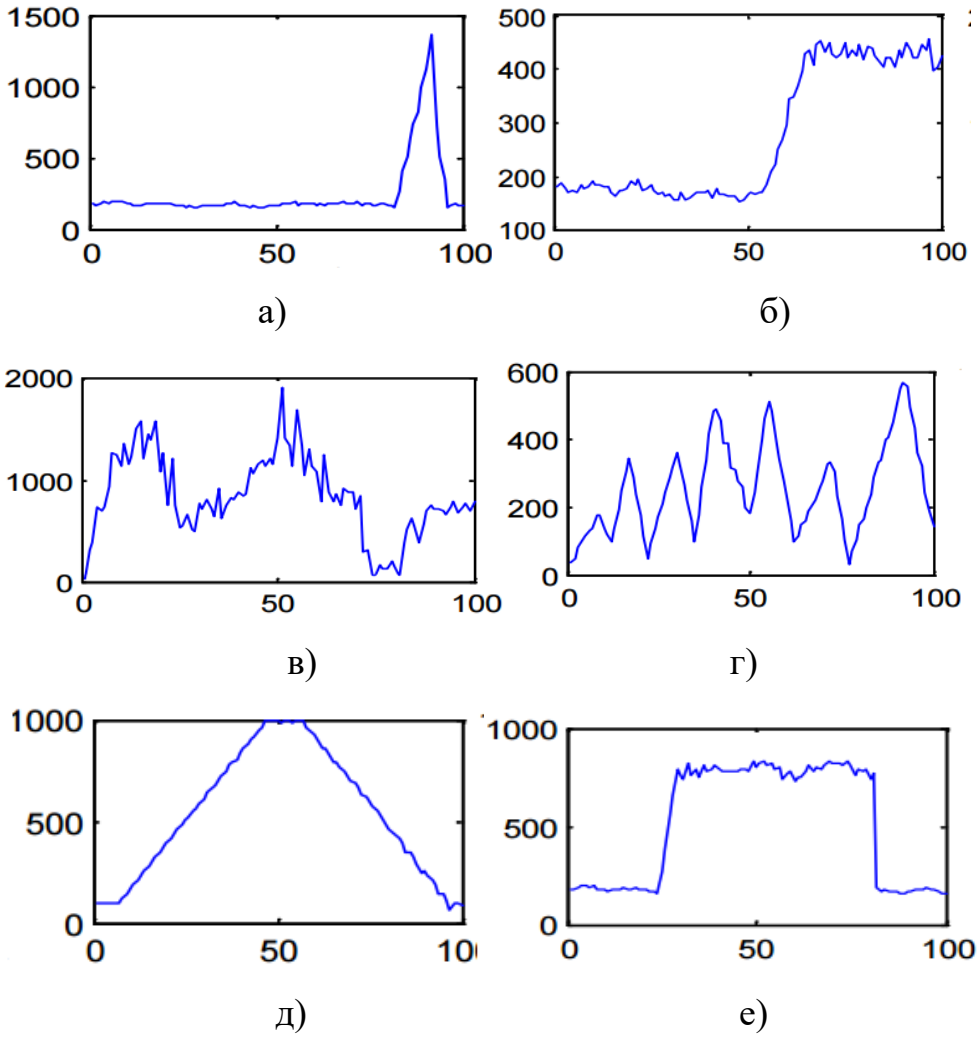


Рисунок 3.6 – Штучні паттерни навантажень: а) Великий пік; б) Подвійна фаза; в) Великі варіації; г) Швидкі зміни; д) Повільна зміна; е) Три фази з крутими змінами.[15]

Було розглянуто такі варіанти стратегій:

- Послідовне зниження коефіцієнта дослідження (S1)
- Початковий високий коефіцієнт розвідки(S2)
- Високий постійний коефіцієнт розвідки (S3)
- Стратегія із використанням RobusT2Scale(S4)

Ці стратегії було порівняно із стандартним австомасштабуванням Azure.

Таблиця 3.2 – Порівняння різних стратегій FQL4KE і стандартного автомасштабування Azure[15]

Стратегія	Критерії	Великий пік навантаження	Подвійна фаза	Великі варіації	Швидкі зміни	Повільна зміна	Три фази з крутою зміною
S1	$rt_{95\%}$	1212 мс.	548 мс.	99 мс.	1319 мс.	512 мс.	561 мс.
	node change	390	360	420	432	355	375
	convergence	32	34	40	65	24	27
S2	$rt_{95\%}$	1298 мс.	609 мс.	1191 мс.	1350 мс.	533 мс.	603 мс.
	node change	412	376	429	486	370	393
	convergence	38	36	87	98	45	28
S3	$rt_{95\%}$	1262 мс.	701 мс.	1203 мс.	1287 мс.	507 мс.	569 мс.
	node change	420	387	432	512	372	412
	convergence	30	29	68	86	40	23
S4	$rt_{95\%}$	1339 мс.	729 мс.	1233 мс.	1341 мс.	567 мс.	512 мс.
	node change	410	377	420	479	366	390
	convergence	N/A	N/A	N/A	N/A	N/A	N/A
Azure	$rt_{95\%}$	1409 мс.	712 мс.	1341 мс.	1431 мс.	1101 мс.	1412 мс.
	node change	330	299	367	398	287	231
	convergence	N/A	N/A	N/A	N/A	N/A	N/A

3.2 Висновки

В цьому розділі було розглянуто сучасні приклади застосування машинного навчання Q-Learning, а саме:

- Deep Q-Learning у застосуванні до ігор Atari 2600
- Fuzzy Q-Learning у застосуванні до Хмарних контролерів.

Deep Q-Learning у використанні до ігор Atari 2600 показав значні результати: у 22 із 49 ігор DQN обігнала людські результати. Результати проти спеціалізованих алгоритмів були навіть кращі – результати були більші в 43 із 49 ігор.

Враховуючи, що на DQN поступала тільки картинка із екрану і рахунок і нейронна мережа не мала ніяких змін в порівнянні з нейронними мережами для інших ігор, можна припустити що DQN можна використовувати в проектах, де потрібен аналіз навколишнього середовища, наприклад управління автомобілем.

В прикладі із хмарними обчисленнями і Fuzzy Q-Learning FQL4KE обійшла стандартний алгоритм автомасштабування в Azure в більшості випадків. Стандартний алгоритм в Azure обійшов тільки стратегію із використанням RobusT2Scale в подвійній фазі (рис. 3.5 б) і в швидких змінах навантаження (рис. 3.5 г).

4 ПРИКЛАДИ ЗАСТОСУВАННЯ Q-LEARNING

4.1 Flappy Bird бот

4.1.1 Опис Flappy Bird

Flappy Bird являє собою 2D гру (рис. 4.1). Мета гри полягає в управлінні польотом пташки, яка безперервно пересувається між рядами зелених труб. При зіткненні з ними відбувається завершення гри. Гравець може виконувати тільки одну дію, при виконанні якої пташка робить ривок на певну величину. При відсутності ривків птах падає через сили тяжіння. Під час прольоту крізь пару труб до рахунку добавляється один бал

Завершення гри відбувається при падінні пташки на землю або дотику до труби.

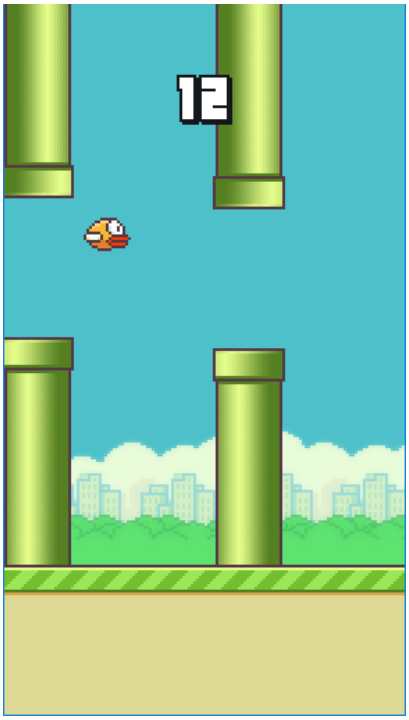


Рисунок 4.1 – Flappy Bird

4.1.2 Взаємодія агенту і середовища

Агент приймає на вхід положення труб і положення пташки(рис. 4.2).

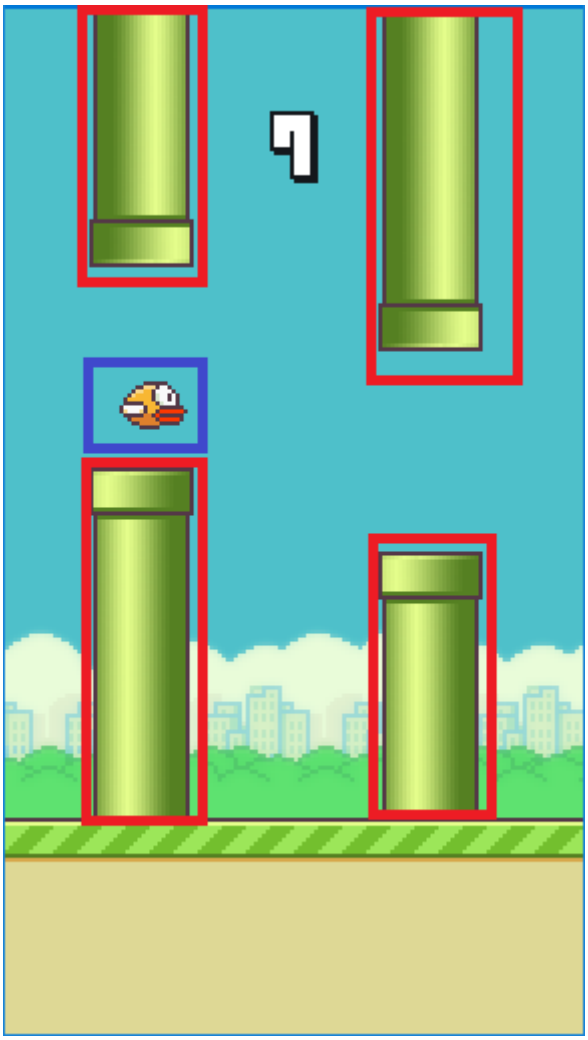


Рисунок 4.2 – Виділення вхідних даних середовища.

Аналізуючи ці дані, агент вибирав одну із двох дій:

- Зробити ривок
- Нічого не робити

Нагорода визначалася таким чином, за кожен крок гри, при умові живої пташки, до нагороди добавлялася 1, при прольоті крізь трубу добавлялося 10 а при смерті пташки від нагороди віднімалася 1000. Метою агента було покращити винагороду.

4.1.3 Тренування

Була використана ϵ -greedy стратегія навчання. В цьому випадку в началі тренування існувала ϵ – шанс, що буде вибрана не дія з найкращим Q, а будь-яка випадкова інша.

Причиною вибору такого алгоритму було те, що при використанні жадібної стратегії в більшості випадків агент не проходив навіть першу пару труб. Відбувалося це із-за такої причини:

В самому початку гри пташка знаходиться одна на екрані, парні труби відсутні (рис 4.3). На цьому етапі агент навчається, що для того, щоби покращити нагороду потрібно давати пташці команду на ривок. І після того, як парні труби появляються, агент уже навчається ігнорувати вхідні дані про положення труб. Із-за цього він не розуміє, що потрібно пролітати між ними. Епсілон же в свою чергу призводить до випадкового вибору, що, в свою чергу, дозволяє дослідити інші варіанти стан/подія, навіть, якщо вони не мають максимальної Q.

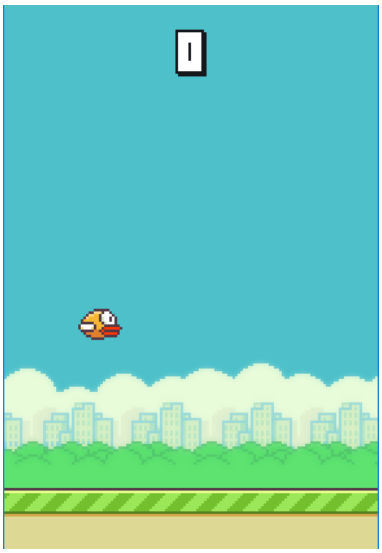


Рисунок 4.3 – Початок гри

4.1.4 Результати

Було проведено 10 000 епох навчання QL агента. Після закінчення тренування середній результат за 100 ігор був 207 очок (рис 4.4)

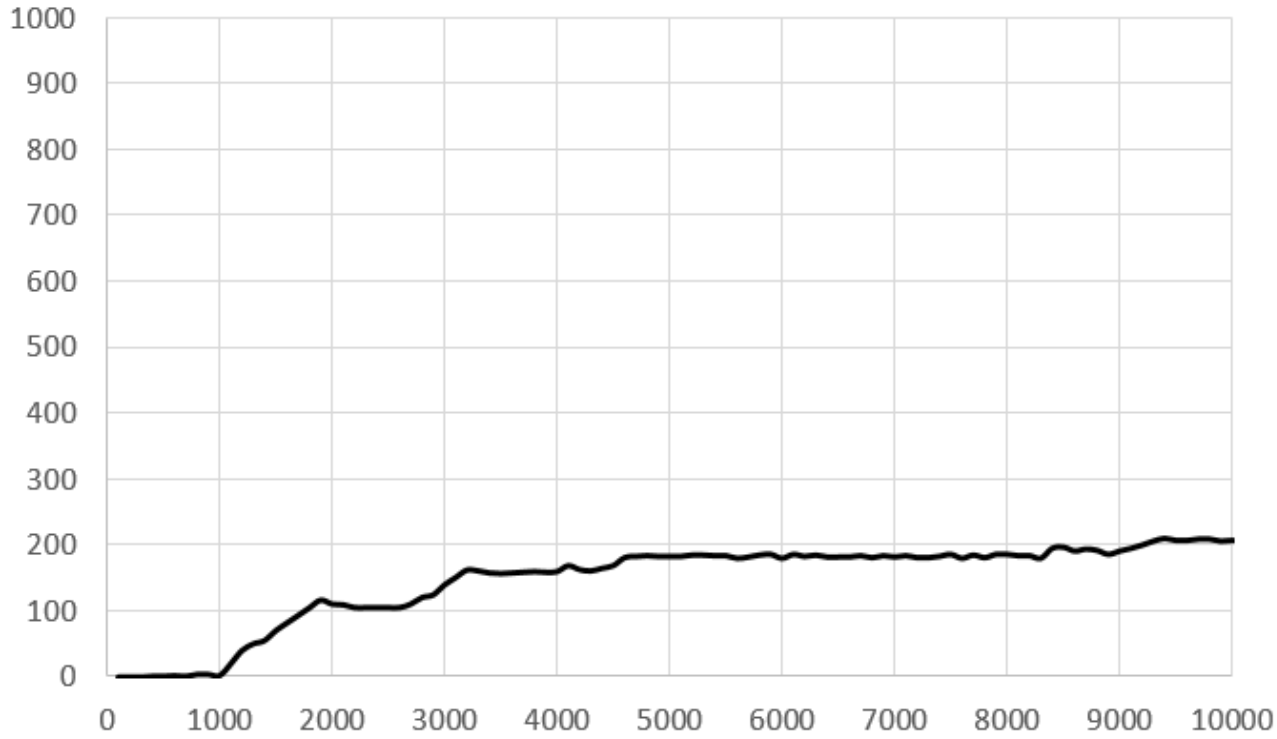


Рисунок 4.4 – Графік рахунку в Flappy Bird під час навчання

Враховуючи дані, отримані із отриманих винагород під час навчання(рис. 4.5) можна судити, що доходити до початку появи парних труб агент навчився доходити десь на відрізку між 100-200 епохою, між 300-400 він навчився проходити першу пару, але лише починаючи із 1000 і вище епох агент починає розбиратися як проходити далі. Десь біля 1900 епохи агент робить помилку у виводах, що призводить до падіння нагороди, але згодом це було виправлено.

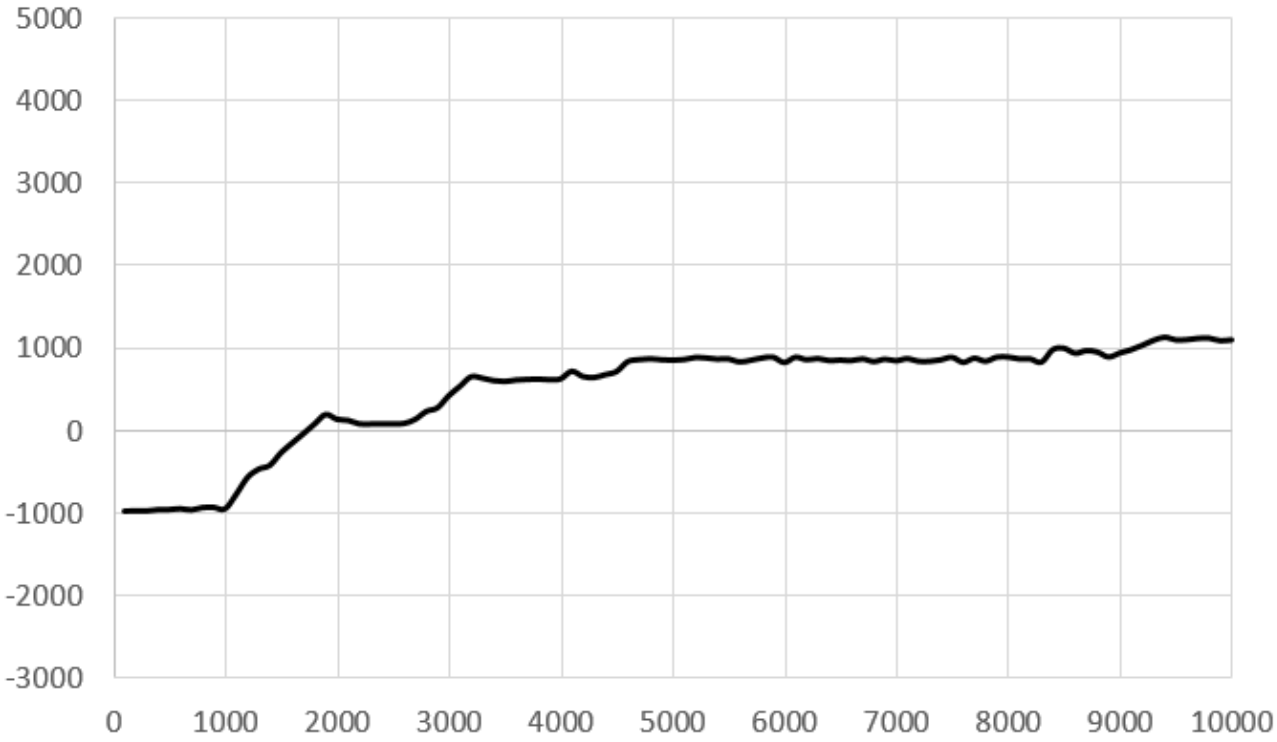


Рисунок 4.5 – значенні винагороди під час навчання

4.2 Bitcoin

4.2.1 Вступ

Bitcoin - це мережа глобального консенсусу, яка створює нову платіжну систему і повністю цифрові гроші. Це перша децентралізована p2p платіжна мережа, яка обслуговується її ж користувачами, без центральних органів управління або посередників.

Bitcoin - це перша реалізація концепції криптовалюти, яку вперше описав в 1998 Вей Дай. В e-mail розсилці, він запропонував ідею нової форми грошей, яка для контролю емісії та транзакцій використовує криптографію замість центрального органу управління.[16]

4.2.2 Взаємодія агенту і середовища

Для формування станів використовуються дані про ціну bitcoin за 2013-2017 рік(рис. 4.6).

Рисунок 4.6 – Курс bitcoin до USD[17]

Агент на вхід приймає дані курсу bitcoin за попередні 20 днів. Також агенту відомі скільки в нього в розпорядженні bitcoin і USD. Аналізуючи ці дані він має вибрати 1 із трьох дій:

- Купити bitcoin
- Продати bitcoin
- Нічого не робити

Нагорода вираховується за як різниця між поточною і попередньою вартістю його портфоліо. Початкове портфоліо має 100 USD і ні одного bitcoin.

Завданням агенту полягає у максимізації вартості його портфоліо у довготривалій перспективі. Так, як в нас завдання максимізувати ціну нашого портфоліо в довготривалій перспективі, ми беремо фактор дисконтування 0.9.

4.2.3 Trading Monkey

Результати роботи агенту буде порівняно із Trade Monkey. Суть Trade Monkey полягає у тому, що на кожному кроці вона з однаковим шансом може вибрати 1 із 3 можливих дій.

Було проведено 3500 ітерацій і була вирахувана кількість bitcoin (рис 4.7), кількість валюти (рис. 4.8) і загальна ціна портфоліо (рис 4.9)

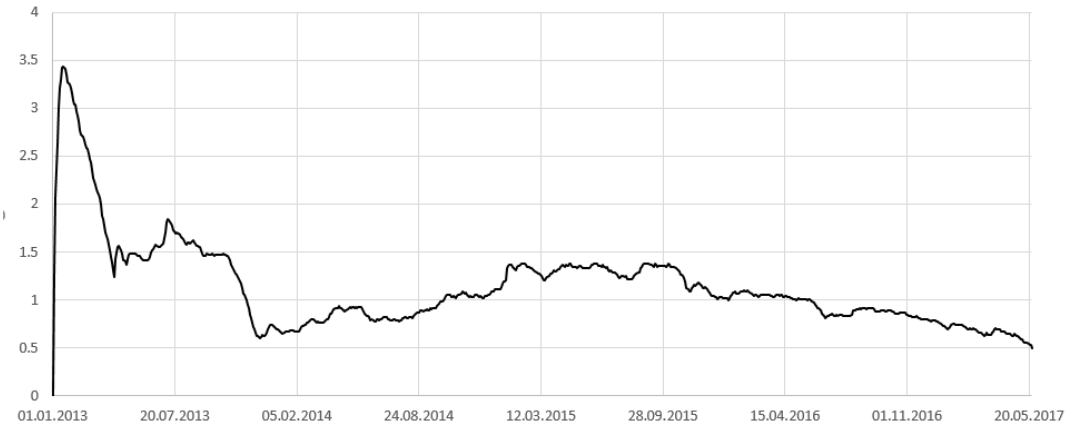


Рисунок 4.7 – кількість bitcoin у портфоліо Trade Monkey на кожному кроці.

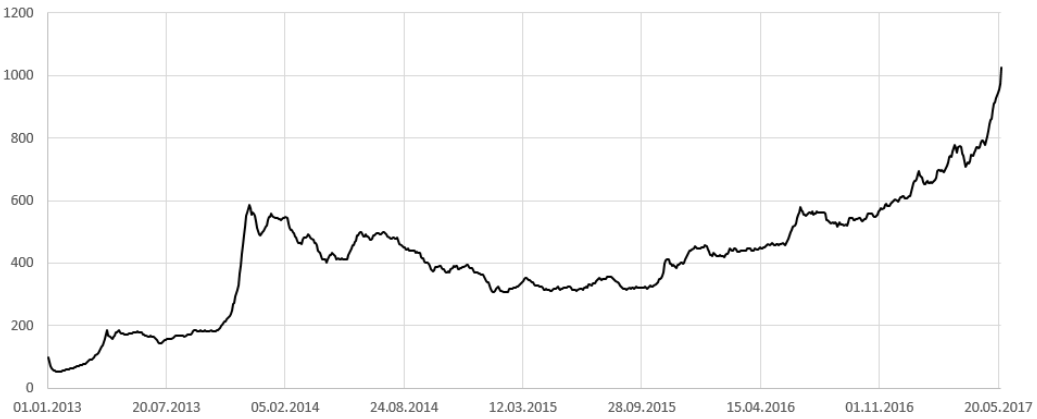


Рисунок 4.8 – кількість валюти у портфоліо Trade Monkey на кожному кроці

Рисунок 4.9 – Загальна ціна портфоліо Trade Monkey у валюті

Із результатів (рис. 4.9) видно, що середня ціна портфоліо сягає 2200-2300

4.2.4 Результати

Після перших 100 епох, QL агент ще не встиг розібратися як правильно опрацьовувати дані. Результати роботи агента(рис 4.10) лиш на трохи перевищили результати Trade Monkey

Рисунок 4.10 – Ціна портфоліо QL агента на кожному кроці після 100 епох

Якщо поглянути на кількість валюти у портфоліо агента (рис. 4.11) то можна побачити, що він ще вибирає дії досить випадково

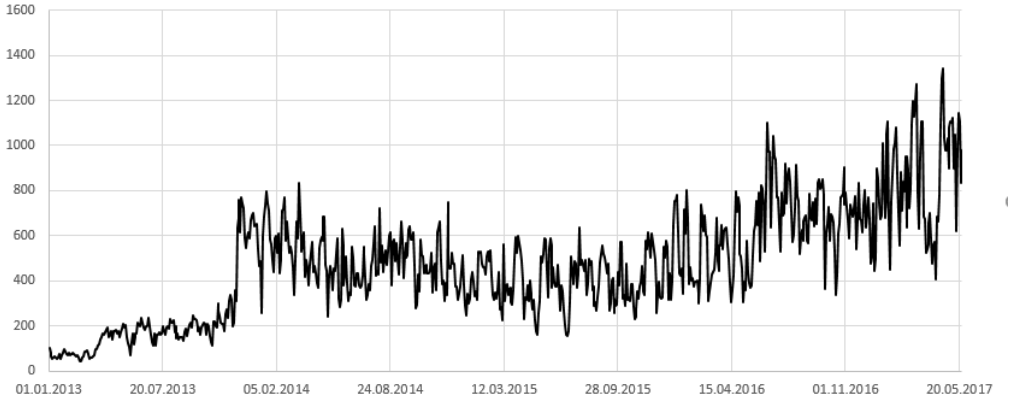


Рисунок 4.11 – Кількість валюти у портфоліо QL агента після 100 епох

Після 3 500 ітерацій агент зміг довести своє портфоліо до 9100-9300 USD(рис 4.12).

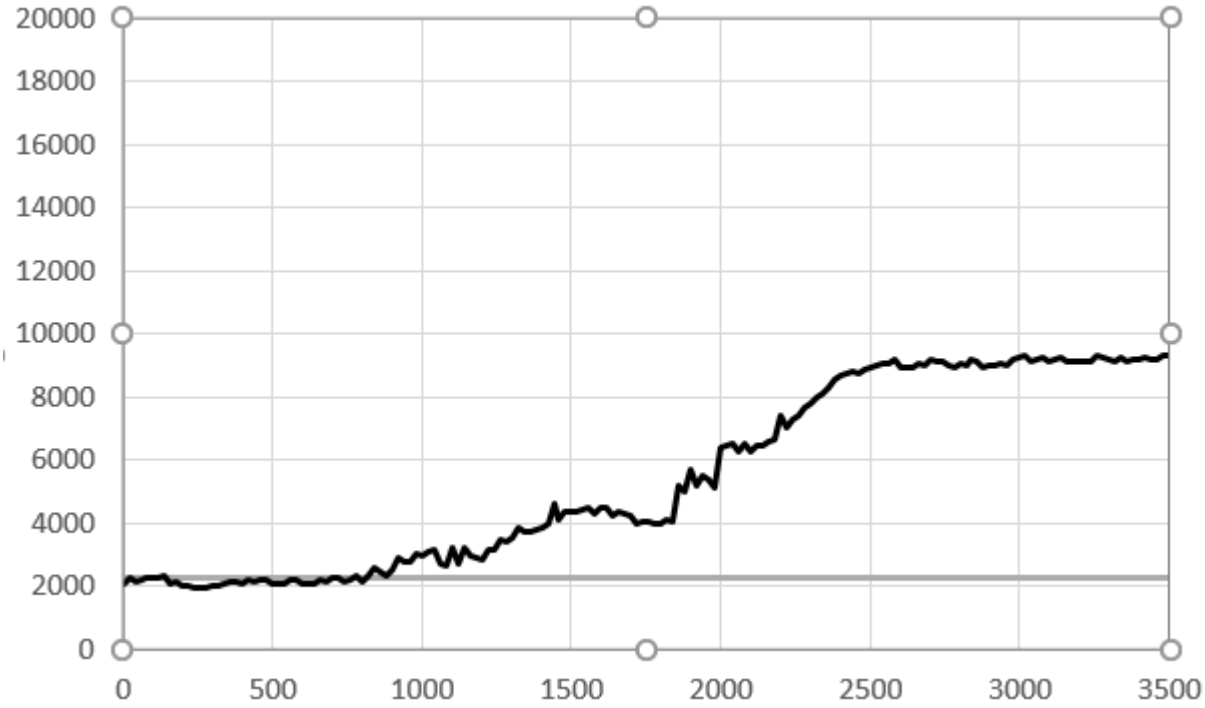


Рисунок 4.12 – зміна ціни портфоліо впродовж епох

4.4 Висновки

В цьому розділі було розглянуто приклади застосування методики машинного навчання Q-Learning. Була створена 1 тестова реалізацію алгоритму QL, яку потім було настроєно під дві різні задачі: для гри Flappy Bird і для купівлі/продажу bitcoin.

QL агент для Flappy Bird справився зі своїм завданням але покращення результатів у нього почалося тільки після 1000 епохи.

У торгівлі bitcoin результати в самому початку були лише трохи кращі ніж у Monkey Trading але доходячи до 700 епохи вони стали покращуватися, згодом пройшов ривок і після 2500 епохи ціна портфоліо стала триматися приблизно одного рівня.

5 ФУНКЦІОНАЛЬНО ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

5.1 Вступ

В даному розділі проводиться аналіз варіантів реалізації модулю з метою вибору оптимальної, з економічної точки зору. А саме проводиться функціонально-вартісний аналіз (ФВА).

Функціонально-вартісний аналіз — це метод комплексного техніко-економічного дослідження об’єкта з метою розвитку його корисних функцій при оптимальному співвідношенні між їхньою значимістю для споживача і витратами на їхнє здійснення. Є одним з основних методів оцінки вартості науково-дослідної роботи, оскільки ФВА враховує як технічну оцінку продукту, що розробляється, так і економічну частину розробки. Крім того, даний метод дозволяє вибрати оптимальний варіант розв’язання задачі, як з погляду розробника, так і з точки зору покупця. Також він дозволяє оптимізувати витрати й час виконання робіт.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.
- для кожної функції визначаються повні річні витрати й кількість робочих часів.

- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

5.2 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах. Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;
- забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;
- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

5.3 Обґрунтування функцій програмного продукту

Головна функція F0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – вибір мови програмування;

F2 – Вибір фреймворка машинного навчання;

F3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування Python

б) мова програмування Java;

в) мова програмування C#, .NET 4.5.

Функція F2:

а) фреймворк TensorFlow

б) фреймворк Bulrap.

в) фреймворк Accord.NET Framework.

Функція F3:

а) інтерфейс користувача, створений за технологією PyQt;

б) інтерфейс користувача, створений за технологією JavaFx;

в) інтерфейс користувача, створений за технологією .NET Framework.

5.4 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5.1).

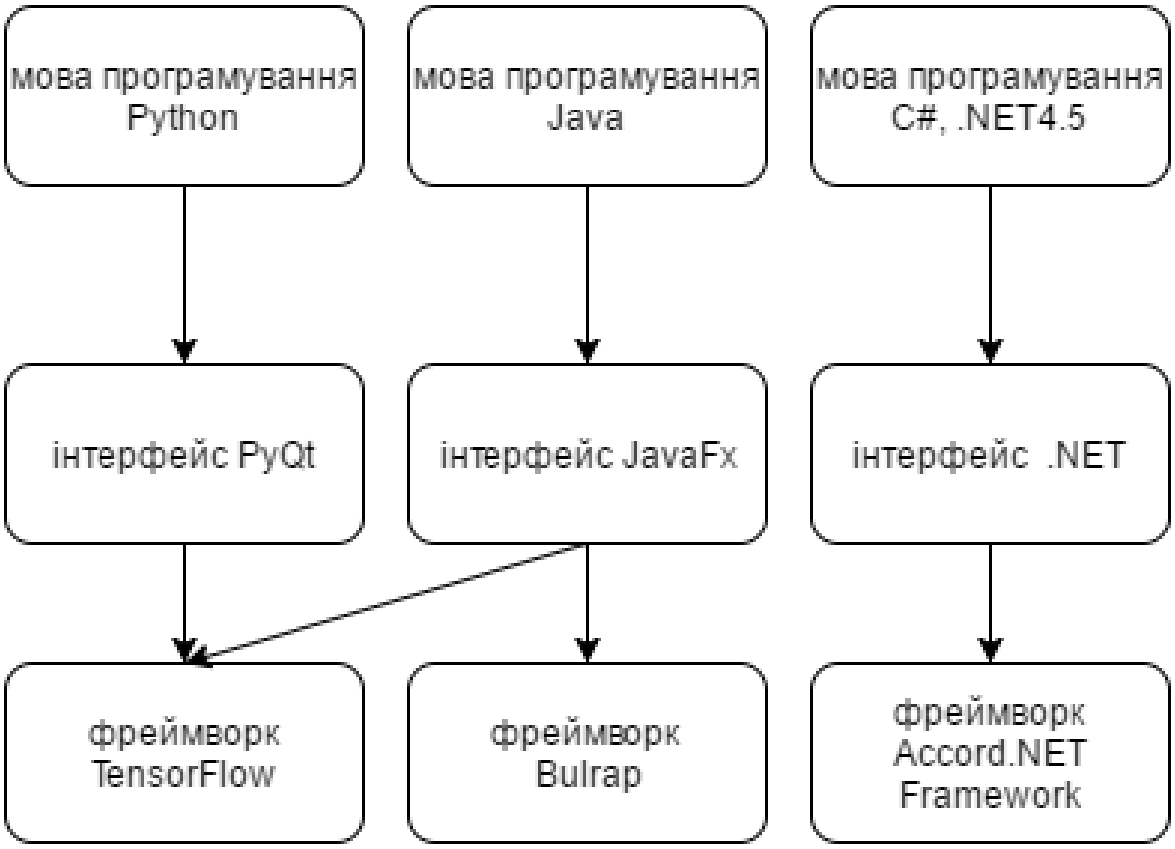


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Код швидко виконується, кросплатформений	Для запуску програми потрібно поставити чимало бібліотек
	Б	Простий запуск на різних платформах, швидкодія	Велика кількість коду
	В	Займає менше часу при написанні коду	Не кр осплатформений
F2	А	велика швидкодія, детальна документація, точність	Складна настройка
	Б	Зручна настройка, Простота запуску	Заплутаний у використанні,
	В	Простий у створенні.	Відсутність кросплатформеності, мало документації
F3	*А*	Легкий у створенні, велика швидкодія	Необхідна додаткова Інсталяція
	Б	Стабільний у використанні	Мала швидкодія
	В	Швидкий, легкий у створенні	Не кросплатформенний

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки для задачі потрібна швидкодія і кросплатформеність вибираємо варіанти а) і б)

Функція F2:

Оскільки нам потрібна велика швидкодія а також точність беремо варіант а).

Функція F3:

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду. Варіант в) відкинутий із-за вибору мов програмування.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1б – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.5 Обґрунтування системи параметрів ПП

5.5.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.
Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – Об’єм Використаної пам’яті
- *X2* – Загруженість процесора
- *X3* – Швидкість відклику інтерфейсу
- *X4* – Час обробки даних

X1: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.
X2: Відображає Загруженість процесора під час виконання програми.
X3: Відображає швидкість відклику інтерфейса
X4: Показує Час обробки даних обраним фреймворком.

5.5.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2

Таблиця 5.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Об’єм пам’яті для збереження даних	X1	Мб	128	64	16
Загруженість процесора	X2	%	100	40	5
Швидкість відклику інтерфейсу	X3	мс	1000	300	20
Час обробки даних	X4	с	2000	1000	400

За даними таблиці 5.2 будуються графічні характеристики параметрів – (рис. 5.2 – рис. 5.5).

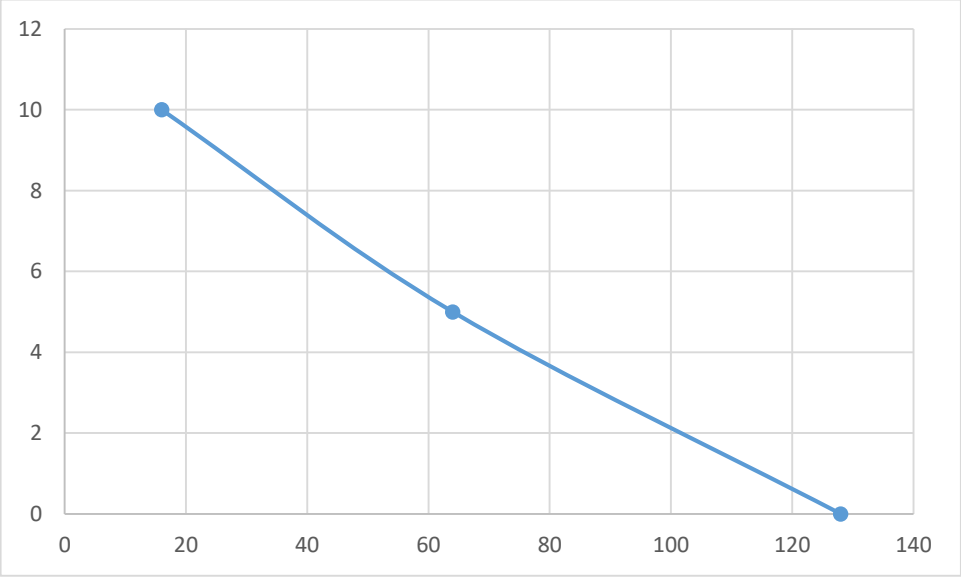


Рисунок 5.2 – X1, об’єм пам’яті збережених даних

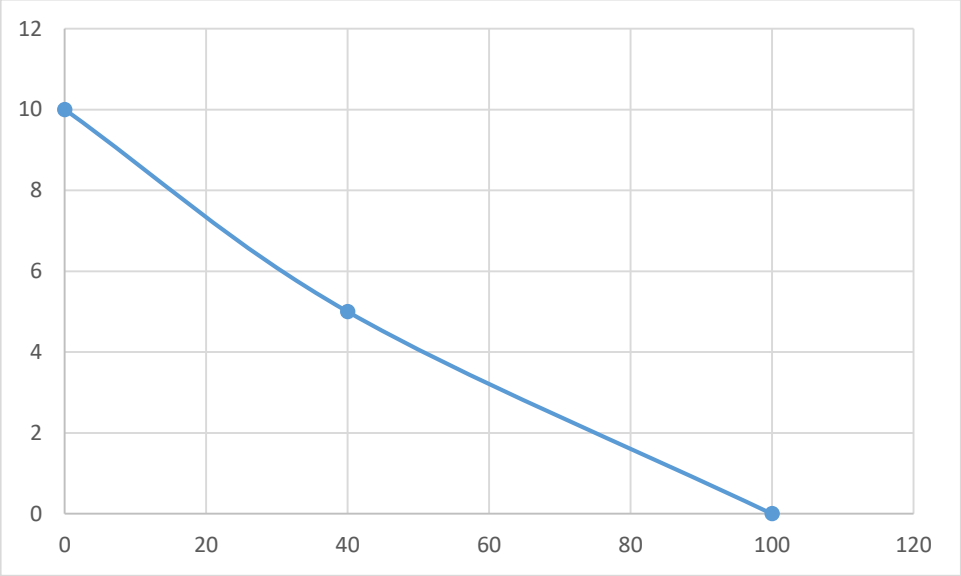


Рисунок 5.3 – X2, Загруженість процессора

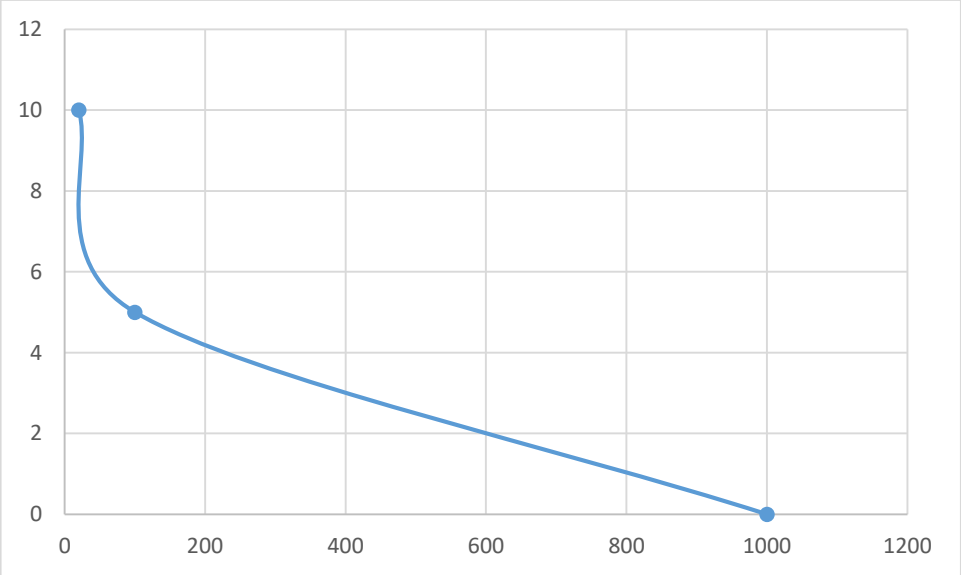


Рисунок 5.4 – X3, Швидкість відклику інтерфейсу

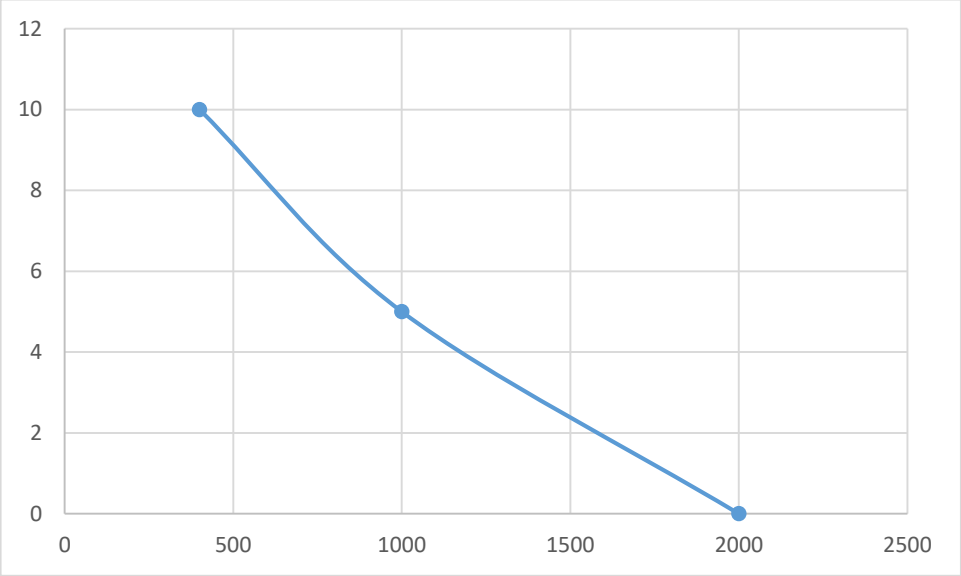


Рисунок 5.5 – X4, Час обробки даних

5.5.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Об’єм пам’яті для збереження даних	Мб	1	2	1	2	1	1	2	10	-8,75	76,5625
X2	Загруженість процесора	%	2	1	2	5	2	3	1	16	-2,75	7,5625
X3	Швидкість відклику інтерфейсу	мс	4	3	5	4	5	4	3	28	9,25	85,5625
X4	Час обробки даних	с	3	4	4	2	2	2	4	21	2,25	5,0625
	Разом		10	10	12	13	10	10	10	75	0	174,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 75,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 18.75.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 174,75$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 174.75}{7^2(4^3 - 4)} = 0{,}71 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	>	<	<	<	<	>	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	<	<	>	<	<	<	<	0,5
X2 і X4	<	<	<	>	<	>	<	<	0,5
X3 і X4	>	<	<	>	>	>	<	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{\text{ij}} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}$,де $b_i = \sum_{i=1}^{N} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$,де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	0,5	2,5	0,156	9,25	0,156	34,125	0,157
X2	1,5	1,0	0,5	0,5	3,5	0,218	12,25	0,207	44,875	0,207
X3	1,5	1,5	1,0	1,5	5,5	0,343	21,25	0,360	77,875	0,360
X4	1,5	1,5	0,5	1,0	4,5	0,281	16,25	0,275	59,125	0,273
Всього:					16	1	98	1	1	59

5.6 Аналіз рівня якості варіантів реалізації функції

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Загруженість процесора) та *X1*(Об’єм пам’яті для збереження даних) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*Швидкість відклику інтерфейсу) обрано не найгіршим (не максимальним).

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри, що беруть участь у реалізації функцій	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1, X2)	А	X1	50	6,7	0,157	1,0519
		X2	30	6,9	0,207	1,4283
	Б	X1	90	3,2	0,157	0,5024
		X2	50	4,8	0,207	0,9936
F2(X3)	А	X3	200	6,4	0,360	2,304
	Б		190	6,5	0,360	2,34
F3(X4)	А	X4	1200	4,6	0,360	1,656

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$$\text{КК1} = 1{,}0519 + 1{,}4283 + 2{,}304 + 1{,}656 = 6{,}4402$$

$$\text{КК2} = 0{,}5024 + 0{,}9936 + 2{,}34 + 1{,}656 = 5{,}492$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.7 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка архітектури класів та їх логіки;

2. Реалізація графічного інтерфейсу

3. Тестування програмного функціоналу;

Варіант I також має завдання

4. Вивчення фреймворка TensorFlow

Варіант II також має завдання

5. Вивчення фреймворка Bulrap

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 2, трудомісткість дорівнює: T_P =36 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.51. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 36 \cdot 1.51 \cdot 0.8 = 43.49$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =19 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

$T_2 = 19 \cdot 0.8 \cdot 0.9 = 13.68$ людино-днів.

Для третього завдання (використовується алгоритм третьої групи складності, степінь новизни А), тобто $Т_Р$ =27 людино-днів, $К_П$ =1.26 ,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_3 = 27 \cdot 0.8 \cdot 1.26 = 27.22$ людино-днів.

Для четвертого завдання (використовується алгоритм третьої групи складності, степінь новизни А), тобто $Т_Р$ =27 людино-днів, $К_П$ =1.26,$К_{СК}$ = 1,$К_{СТ}$ =1:

$Т_5 = 27 \cdot 1.51 = 34.02$ людино-днів.

Для п’ятого завдання (використовується алгоритм першої групи складності, степінь новизни Б), тобто $Т_Р$ =64 людино-днів, $К_П$ =1,021,$К_{СК}$ = 1, $К_{СТ}$ =1:

$Т_4 = 64 \cdot 1.021 = 65.344$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I = (43.49 + 13.68 + 27.22 + 34.02) \cdot 8 = 947.28$ людино-годин;

$Т_{II} = (43.49 + 13.68 + 27.22 + 65.344) \cdot 8 = 1197.87$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 9000 грн., один аналітик з окладом 6000 грн . Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.},$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_Ч = \frac{9000 + 9000 + 6000}{3 \cdot 21 \cdot 8} = 47.6 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д,$$

де $С_Ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 47.6 \cdot 947.28 \cdot 1.2 = 54108.63$ грн.

II. $С_{ЗП} = 47.6 \cdot 1197.87 \cdot 1.2 = 68422.33$ грн.

Відрахування на всі види соціального страхування становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 54108.63 * 0.22 = 11903.9$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 68422.33 * 0.22 = 15052.91$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 9000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 9000 \cdot 0,2 = 21600$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 21600 \cdot (1 + 0.2) = 25920$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 25920 * 0,22 = 5702.4$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 25000 = 7187.5$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 25000 \cdot 0.05 = 1437.5$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,75 \cdot 1.93819 = 2480.49$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 25000 \cdot 0,67 = 16750$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 25920 + 5702.4 + 7187.5 + 1437.5 + 2480.49 + 16750 = 59477.89$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 59477.89/1706.4 = 34.86$ грн/год.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 34.86 * 947.28 = 33022.18$ грн.;

II. $С_М = 34.86 * 1197.87 = 41757.74$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 54108.63 * 0,67 = 36252.78$ грн.;

II. $С_Н = 68422.33 * 0,67 = 45842.96$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 54108.63 + 11903.9 + 33022.18 + 36252.78 = 135287.49$ грн.;

II. $С_{ПП}$ = 68422.33 + 15052.91 + 41757.74 + 45842.96 = 171075.94 грн.;

5.8 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1}$ = 6,4402 / 135287.49 = $0,47 \cdot 10^{-4}$;

$К_{ТЕР2}$ =5,492 / 171075.94 = $0.32 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0.47 \cdot 10^{-4}$.

5.9 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного

комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,47 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Python;

– фреймворк TensorFlow

– інтерфейс користувача, створений за технологією PyQt.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

Дипломна робота присвячена дослідженню застосування методики машинного навчання Q-Learning.

В першому розділі було описано цілі дипломної роботи і сформовані задачі, які мають бути виконані в ній. Також там була обґрунтована актуальність роботи

Другий розділ було присвячено дослідження предметної області. Було загально розглянута область машинного навчання і детальніше було розглянуто Q-learning. Було розглянуто принцип роботи агента, що потрібно вважати середовищем і як задаються цілі і винагороди. Також було розглянуто марковську властивість і марковський процес прийняття рішень.
У третьому розділі проводився аналіз прикладів сучасного застосування Q-learning а саме застосування QL у хмарних контролерах і при грі в ігри Atari 2600.

В агенті, який використовувався в іграх Atari 2600 використовувалася модифікація алгоритму QL під назвою Deep Q-Learning (Deep Reinforcement Learning). В ній, замість Q функції використовувалася convolutional neural network. В результаті у 43 із 49 був перевершений результат спеціалізованих алгоритмів а в 22 із 49 було перевершено результат людей.

У хмарних контролерах використовували модифікацію QL на основі нечіткої логіки. Це дозволило покращити опрацювання даних, так як із-за особливостей хмарних технологій і їх комплексності, дані частіше всього приходять неповними. В результаті алгоритм показав кращі результати ніж автомасштабування використане в Azure.

В цьому розділі було розглянуто приклади застосування методики машинного навчання Q-Learning. Була створена 1 тестова реалізацію алгоритму QL, яку потім було настроєно під дві різні задачі. Першою задачею являлась гра в Flappy Bird. Агент в цьому випадку показав непогані, починаючи з 1000 епохи. Другою задачею була покупка/продаж bitcoin з метою збільшити ціну

свого портфоліо. У торгівлі bitcoin результати в самому початку були лише трохи кращі ніж у Monkey Trading але доходячи до 700 епохи вони стали покращуватися, згодом пройшов ривок і після 2500 епохи ціна портфоліо стала триматися приблизно одного рівня.

У четвертому розділі було проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Даний варіант виконання програмного комплексу дає зручний користувацький інтерфейс, найкращі показники надійності програмного продукту, функціонал, що задовольняє заданим вимогам та непогану швидкодію.

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини. В першій з них проведено дослідження ПП з технічної точки зору, а другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого.

ПЕРЕЛІК ПОСИЛАНЬ

1. Philosophical Transactions − the world's first science journal. – Режим доступу: http://rstl.royalsocietypublishing.org/. – Дата доступу: 05.05.2017
2. A. M. TURING I.—COMPUTING MACHINERY AND INTELLIGENCE/ A.M. Тюрінг // Mind. – 1950. – № 236. – С. 433-460.
3. Stochastic neural analog reinforcement calculator. – Режим доступу: http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/. – Дата доступу 05.05.2017
4. How Many Computers to Identify a Cat? – Режим доступу: http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html. – Дата доступу: 05.05.2017
5. In a huge breakthrough, google's ai beats a top player at the game of go. – Режим доступу: https://www.wired.com/2016/01/in-a-huge-breakthrough-googles-ai-beats-a-top-player-at-the-game-of-go/. – Дата доступу: 05.05.2017
6. Онлайн журнал engadget (Google DeepMind AI wins final Go match for 4-1 series win). – Режим доступу: https://www.engadget.com/2016/03/14/the-final-lee-sedol-vs-alphago-match-is-about-to-start/. – Дата доступу: 05.05.2017
7. Comparison of the DQN agent with the best reinforcement learning methods. – Режим доступу: http://www.nature.com/nature/journal/v518/n7540/fig_tab/nature14236_F3.html. – Дата доступу: 05.05.2017
8. Офіційний сайт TensorFlow. – Режим доступу: https://www.tensorflow.org/. – Дата доступу: 05.05.2017
9. Моделирование процессов обучения в нейронных сетях. – Режим доступу: http://old.exponenta.ru/soft/others/mvs/stud3/3.asp. – Дата доступу: 05.05.2017
10. CS234: Reinforcement Learning – Режим доступу: http://web.stanford.edu/class/cs234/index.html. – Дата доступу: 05.05.2017

11. Саттон Р.С Обучение с подкреплением / Саттон Р.С, Э. Г. Барто // БИНОМ, Лаборатория знаний, 2014 – С. 42-96.
12. Mnih, V. Playing Atari with deep reinforcement learning. Technical Report / Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller M. – DeepMind Technologies, 2013 – С. 7
13. Schematic illustration of the convolutional neural network. – Режим доступу: http://www.nature.com/nature/journal/v518/n7540/fig_tab/nature14236_F1.html. – Дата доступу: 05.05.2017
14. Comparison of games scores obtained by DQN. – Режим доступу: http://www.nature.com/nature/journal/v518/n7540/fig_tab/nature14236_ST2.html. – Дата доступу: 05.05.2017
15. Pooyan J. Self-Learning Cloud Controllers: Fuzzy Q-Learning for Knowledge Evolution / Pooyan J., Amir S., Claus P., Andreas M., Giovani E. – International Conference on Cloud and Autonomic Computing, 2015 – С. 8
16. Офіційний сайт Bitcoin. Режим доступу: https://www.bitcoin.com/. – Дата доступу: 05.05.2017
17. Офіційний сайт Blockchain. Режим доступу: https://blockchain.info/charts/market-price?timespan=all. – Дата доступу: 05.05.2017