Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Реалізація системи для навчання штучного інтелекту у комп’ютерній грі

Виконав: студент IV курсу, групи СНзс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Лабунський П.В. (прізвище та ініціали)

Керівник (підпис) Дмитроца Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль - 2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Лабунському Павлу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація системи для навчання штучного інтелекту у комп'ютерній грі

Керівник роботи Дмитроца Леся Павлівна, к.т.н., доц. каф. КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «02» 03 2021 року № 4/7-170

2. Термін подання студентом завершеної роботи 14.06.2021р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналітична частина. 1.1 ШНМ. 1.2. Математична модель нейрона. 2. Теоретична частина. 2.1. Методи навчання ШНМ. 2.2. Опис середовища моделювання і агентів. 3. Практична реалізація розробки та проведення експериментів. 3.1. Вимоги до системи. 3.2. Опис архітектури. 3.3 Алгоритм роботи програми. 3.4. Експерименти 4. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета, задачі дослідження. 4. Визначення поняття ШНМ 5. Математична модель нейрона, Методи навчання ШНМ. 6. Опис середовища моделювання і агентів. 7. Скріншоти роботи програми. 8. Вимоги до системи. 9. Опис архітектури. 10. Програмна архітектура системи. 11. Встановлення початкових параметрів. 12. Результати проведеного дослідження.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 30 квітня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	02.03 – 05.03	*Виконано*
2.	Підбір джерел про навчання штучного інтелекту	06.03 – 25.03	*Виконано*
3.	Опрацювання джерел про математичні моделі нейрона		
	та навчання ШНМ	26.03 – 12.04	*Виконано*
4.	Виконання дослідження щодо системи для навчання		
	штучного інтелекту у комп'ютерній грі	13.04 – 29.04	*Виконано*
5	Розроблення програмного коду	30.04 – 15.05	*Виконано*
6.	Оформлення розділу «Аналітична частина»	15.05 – 20.05	*Виконано*
7.	Оформлення розділу «Теоретична частина»	21.05 – 28.05	*Виконано*
8.	Оформлення розділу «Практична реалізація розробки та		
	проведення експериментів»	29.05 – 06.06	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	12.05 – 22.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	28.05 – 10.06	*Виконано*
11.	Нормоконтроль	11.06 – 14.06	*Виконано*
12.	Перевірка на плагіат	11.06 – 14.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	08.06 – 10.06	*Виконано*
14.	Захист кваліфікаційної роботи	14.06	

Студент _____ Лабунський П.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Реалізація системи для навчання штучного інтелекту у комп’ютерній грі // Лабунський Павло Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем та програмної інженерії, кафедра комп’ютерних наук, група СНзс–42 // Тернопіль, 2021 // С. – 53, рис. – 18, табл. – 1, слайдів – 12, бібліогр. – 17 .

Ключові слова: ГЕНЕТИЧНИЙ АЛГОРИТМ, НЕЙРОННА МЕРЕЖА, НАВЧАННЯ МЕРЕЖІ, АГЕНТ, UNITY3D

Кваліфікаційна робота присвячена розробці та реалізації системи для навчання штучного інтелекту у комп’ютерній грі. Докладно розглянуто поняття нейромережі, її структура, моделі нейрона і види функцій активації. Також описані можливі методи навчання нейронних мереж. Основна увага приділена генетичному алгоритму. Розкрито ідею середовища моделювання і механізму взаємодії агентів в ньому. Описано будову самої системи, алгоритм її роботи та програмну архітектуру. Система дозволяє створити середовище із заздалегідь заданими параметрами. Середовище містить в собі агентів, керованих нейромережею, навчання якої відбувається через генетичний алгоритм. Такий спосіб добре підійшов для завдання, кінцевий результат якого заздалегідь невідомий. Так само система надає користувачеві набір функцій для управління еволюцією агентів, збереження або завантаження нейронної мережі.

Система розроблялася на ігровому рушію Unity 3D, що дозволило скористатися просунутими методами візуалізації, а також полегшити впровадження середовища в ігрові проекти, що дозволяє отримати наочний результат роботи не тільки у вигляді цифр. Проведено ряд модельних експериментів, результати яких дозволяють стверджувати про еволюцію агентів, керованих нейромережею.

ANNOTATION

Implementation of artificial intelligence learning system in computer game // Labunskiy Pavlo // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2021 // P. - 53, Fig. - 18, Table - 1, Slide - 12, References - 17.

Keywords: GENETIC ALGORITHM, NEURAL NETWORK, NETWORK LEARNING, AGENT, UNITY3D

Thesis deals with the development and implementation of a system for teaching artificial intelligence in a computer game. The concept of a neural network, its structure, models of a neuron and types of activation functions are considered in detail. Possible methods of learning neural networks are also described. The main attention is paid to the genetic algorithm. The idea of the modeling environment and the mechanism of interaction of agents in it is revealed. The structure of the system itself, the algorithm of its operation and software architecture are described.

The system allows you to create an environment with predefined parameters. The environment includes agents controlled by a neural network, which is trained through a genetic algorithm. This method is well suited for the task, the end result of which is unknown in advance. Similarly, the system provides the user with a set of functions to control the evolution of agents, save or load the neural network.

The system was developed on the Unity 3D game engine, which allowed to use advanced visualization methods, as well as to facilitate the implementation of the environment in game projects, which allows you to get a clear result not only in the form of numbers. A number of model experiments have been performed, the results of which allow us to assert the evolution of neural network-controlled agents.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

ГА – генетичний алгоритм

Н - нейрон

ПЗ – програмне забезпечення

ШІ – штучний інтелект

ШНМ – щтучна нейронна мережа

ЗМІСТ

Вступ 7
1 Аналітична частина 7
1.1 ШНМ 9
1.2 Математична модель Н 11
1.2.1 Формальний Н 13
1.2.2 Одношаровий перцептрон 14
1.2.3 Багатошаровий перцептрон 15
1.2.4 Сигмовидний Н 16
2 Теоретична частина 20
2.1 Методи навчання ШНМ 20
2.1.1 Навчання з учителем 20
2.1.2 Метод Хеба 20
2.1.3 Правило корекції за помилкою 21
2.1.4 Навчання методом змагання 21
2.1.5 Метод зворотного поширення 22
2.1.6 ГА 25
2.2 Опис середовища моделювання і агентів 27
3 Практична реалізація розробки та проведення експериментів 34
3.1 Вимоги до системи 34
3.2 Опис архітектури 34
3.3 Алгоритм роботи програми 38
3.4 Експерименти 40
4 Безпека життєдіяльності, основи охорони праці 46
4.1 Навчання працюючих і інструктажі з охорони праці 46
4.2 Санітарно-гігієнічні вимоги до умов праці. 48
Висновки 51
Перелік використаних джерел 52
Додатки

ВСТУП

Теорія ШНМ почала розглядатися в 40-х роках 20 століття через досягнення науки в біології. Власне штучні Н містять набори компонентів, які дозволяють моделювати елементарні функції реальних біологічних Н. Організовуються ці елементи так, щоб відповідати анатомії мозку людини. ШНМ мають лише поверхневу подібність з природною нервовою системою, але не дивлячись на це такі мережі показують цікаві властивості, подібні до властивостей органічного мозку. Прикладом може служити поведінка ШНМ, оскільки в залежності від факторів навколишнього середовища така мережа змінює свою поведінку, адаптуючись під нього. Найбільш особливою і цікавою здатністю нейромереж є навчання і запам'ятовування інформації, імітація розумового процесу людини. ШНМ здатна навчитися шляхом отримання певних вхідних сигналів і в залежності від ситуації зреагувати на неї. Пройшовши навчання певну кількість разів, мережа перестає реагувати на слабкі зміни сигналів на вході. Це можна назвати здатністю мережі бачити образ, незважаючи на шум і спотворення. Такі мережі також володіють надійністю: в випадку некоректної роботи або відмови декількох елементів, мережа все одно здатна видати правильний результат, але менш точно. Незважаючи на це, варто зазначити, що ШНМ не є панацеєю. Вони дуже погано підходять для завдань, де потрібні точні і безпомилкові математичні розрахунок.

Немає сумнівів в актуальності даної проблеми, адже останнім часом спостерігається високий інтерес до ШІ і ШНМ. Зростання інтересу досить зрозуміле, адже ШНМ є нічим іншим, як моделлю природної нервової системи, виходячи з цього можна стверджувати, що вивчення і відтворення таких мереж надає можливість багато чого довідатися про функціонування самих природних систем.

У даній роботі проводилися дослідження впливу ШНМ на ігровий процес додатка.

Метою роботи є реалізація системи із заданими параметрами для навчання простого ШІ в комп’ютерній грі. Навчання ШІ через таку систему дозволить

урізноманітнити ігровий процес і уникнути штучного ускладнення рівня гри за рахунок одного лише підвищення характеристик і кількості ігрових ШІ.

Для досягнення мети виділено ряд завдань:

– побудова структури ШНМ;

– реалізація ГА;

– опис простого ШІ і середовища;

– проведення ряду експериментів для отримання даних про отриману систему.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 ШНМ

До сих пір немає єдиної встановленої думки щодо чіткого і однозначного тлумачення самого поняття, що таке власне ШНМ. Якщо звернутися до літератури, то можна навести наступні конкретні визначення цього терміну (табл. 1.1).

Таблиця 1.1 – Визначення поняття ШНМ

Визначення терміну	Література
Система, що складається з множини простих обчислювальних елементів, що працюють паралельно. Результат роботи мережі визначається структурою мережі, силою зв'язків, а також видом обчислень, виконуваних кожним елементом	3
Системи, здатні отримувати, зберігати і використовувати знання	5
Система, що складається з великої кількості простих обчислювальних елементів. Результат роботи кожного елемента залежить тільки від його внутрішнього стану. Все елементи працюють незалежно один від одного, тобто без синхронізації з рештою елементів	6
Паралельний розподілений процесор, здатний самостійно добувати дані з інформації, яка надходить. Робота такої мережі нагадує роботу мозку, так як знання отримуються за допомогою процесу навчання, а отримані знання зберігаються не в окремому елементі , а розподілені по всій мережі	7

Отже, якщо виходити із наведених в табл. 1.1 визначень, то можна сказати, що ШНМ - це система, до складу якої входить певна множина процесорів, кожен з яких володіє своєю локальною пам'яттю. Збережене в цій пам'яті носить назву

стану процесора. Процесори мають можливість обміну числовими даними один з одним. У свою чергу результат роботи процесора перебуває в залежності виключно від його стану і вхідних даних.

Загалом, ШНМ забезпечують хороші рішення проблем з такими особливостями:

– у цій проблемі використовуються «шумні» дані;

– чи може знадобитися швидка обробка;

– може не знадобитися найбільш досконале рішення проблеми.

Використовувати ШНМ можливо тільки після проведення процедури навчання. Під час навчання, стан кожного елемента коригуються на підставі даних, що надходять, таким чином мережу обчислює правильну відповідь. Іншими словами, мережа «навчається» на прикладах, подібно до того як дитина вчиться по картинках.

Немає простих правил для вирішення проблеми - все, що є, це набір типових рішень. Мережа може «навчитися» таким чином, щоб вона давала хороші відповіді на подібні нові випадки.

Існують дві основні проблеми з використанням таких мереж. По перше, не існує точного розуміння того, наскільки велика (скільки вузлів і з'єднань) мережа повинна бути для вирішення проблеми певної складності. Другим недоліком цих мереж може бути дуже тривалий час, який іноді необхідно для того, щоб навчити мережу відповідним відповідям - ці мережі навчаються контрольованим чином – вхідні дані багаторазово передаються в мережу, а з'єднання налаштовуються так, щоб спробувати досягти цільового виходу.

Варто пам'ятати, що в мережах саме людина вирішує, як мережа повинна реагувати і адаптуватися під час навчання. Це явно відрізняється від поведінки мозку, який «сам по собі» здатний встановлювати зв'язки між Н для виконання певних функцій - іншими словами, проявляє самоорганізацію.

Якщо розглядати актуальне використання ШНМ в ігровій індустрії, то можна виділити кілька хороших прикладів використання даної технології. Наприклад, дослідниками з Единбурзького університету була розроблена фазово – функціональна ШНМ, яка процедурно генерує анімацію моделі ігрового

персонажа для різних ландшафтів.

Ще одним хорошим прикладом є ШНМ "Борис" для ШІ в комп'ютерній грі "Бліцкриг 3". Це перша ШНМ, створена для стратегії в режимі реального часу. "Борис" може реагувати на різні ігрові ситуації, а також здатний передбачати деякі дії гравця, не використовуючи при цьому допоміжних даних, тобто ШІ отримує дані симетричні до даних гравця і, таким чином, гра проходить чесно.

1.2 Математична модель Н

Власне, Н - це якась обчислювальна одиниця, котра одержує певну інформацію на вхід і проводить над нею ряд простих обчислень. Умовно Н можна розділити на три основні типи:

– вхідний;

– прихований;

– вихідний.

Якщо розглядати нейромережу, до складу якої входить безліч нейронів , то можна сказати, що вона складається з шарів. Відповідно, існує вхідний шар, котрий одержує інформацію, n прихованих шарів, котрі опрацьовують цю інформацію і вихідний шар , що виводить результати обчислень. Кожен нейрон володіє двома основними параметрами: вхідні дані (input data) і вихідні дані (output data). Вхідний нейрон в даному випадку: input = output. В інших нейронах в поле input передається сумарна інформація нейронів з попереднього шару , після чого отримана інформація нормалізується , за допомогою функції активації $f(x)$, і передається в поле output [4].

Синапс - зв'язок між парою Н. Синапси мають параметр, умовно званий вагою. Завдяки даному параметру, при передачі від одного Н до іншого інформація на вході змінюється. У того Н, котрий володіє великою вагою, інформація буде домінувати в наступному Н. Властиво сукупність ваг можна назвати матрицею ваг ШНМ. Така матриця є своєрідним мозком системи. Варто згадати, що тільки дякуючи вагам, мережа має можливість обробляти інформацію і видавати результат.

Власне функція активації є способом нормалізації вхідних даних. Виникають ситуації, коли вхідна величина перевищує потрібний діапазон, але коли ця величина пройде через функцію нормалізації на вихід, тоді отримується величина, котра задовольняє заданому діапазону.

Існує три основних функції нормалізації:

– лінійна;
– сигмовидна (логістична);
– гіперболічний тангенс.

Дані функції відрізняються діапазоном значень.

Розглянемо їх докладніше.

Лінійна функція. Застосовується при тестуванні мережі чи передачі інформації без змін (рис.1.1).

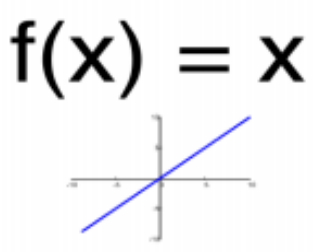

Рисунок 1.1 – Лінійна функція

Сигмоід. Володіє діапазоном значень [0,1] (рис.1.2).

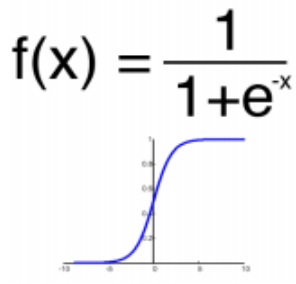

Рисунок 1.2 – Сигмовидна функція

Гіперболічний тангенс. Діапазон значень для цієї функції визначений, як [-1,1] (рис. 1.3). Функцію слід застосовувати лише в тому випадку, якщо мається

на увазі наявність негативних значень. Якщо ж в мережі використовуються виключно позитивні числа, то використання даної функції недоцільно, так як в такому випадку результати мережі сильно погіршуються.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

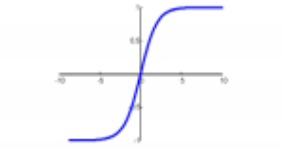

Рисунок 1.3 – Гіперболічний тангенс

Існує безліч моделей Н, головна відмінність яких полягає в обчислювальної складності і схожості з живим Н.

1.2.1 Формальний Н

Найпростішою є класична модель або «формальний Н» (рис. 1.4). У такій моделі Н володіє декількома вхідними каналами і одним вихідним каналом. Н отримує дані завдання на вхідні канали, а результат обчислень формується на вихідному каналі. Сам Н займається обчисленням зваженої суми сигналів w_1,..., w_k з вхідних каналів, і наступним перетворенням отриманої суми через функцію активації *F(S)*.

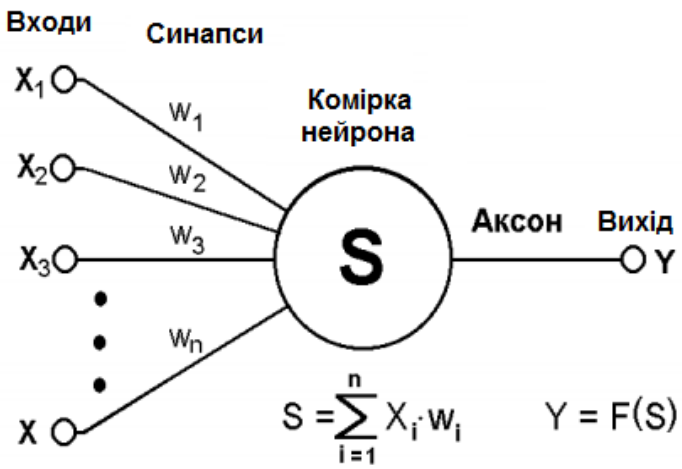

Рисунок 1.4 – Формальний Н

Необхідно навести наступні позначення:

x_i - значення сигналу на вхід,

w_i - ваговий коефіцієнт Н,

F - функція активації,

Y - значення Н на виході.

Множина, що складається з порогового рівня і всіх ваг, називається параметрами Н. Аналогічно, параметрами мережі називають множину параметрів всіх складових її Н. Вихід Н можна визначити за формулою 1.1 [8]:

$$Y = F(\sum (x_i * w_i) \tag{1.1}$$

Варто сказати і про недоліки формального Н. Мається на увазі, що Н здатний обчислювати вихід миттєво, внаслідок чого немає можливості моделювати системи з будь-яким внутрішнім станом. На відміну від біологічних, формальні Н не мають можливості синхронної обробки інформації. Так само мінусом є відсутність будь - яких усталених алгоритмів, які використовуються для вибору функції активації. Відсутня можливість регулювання роботи всієї мережі взагальному. Надмірна формалізація таких понять як «поріг» і «вагові коефіцієнти». У природних Н зміна порога відбувається динамічно, в залежності від стану всієї мережі і безпосередньо від активності самого Н, а зміна вагових коефіцієнтів залежить від того, як проходять сигнали [8].

1.2.2 Одношаровий перцептрон

Один Н може виконати дуже прості обчислення, але основний функціонал ШНМ забезпечується не окремо взятими Н, а безпосередньо зв'язками між ними. Одношаровий перцептрон (рис.1.5) є простою ШНМ, що має в своєму складі групу Н, котрі утворюють шар. Дані на вході кодуються вектором значень $X = (x_1,..., x_n)$, кожному елементу x і відповідає певний вхід кожного Н в шарі.

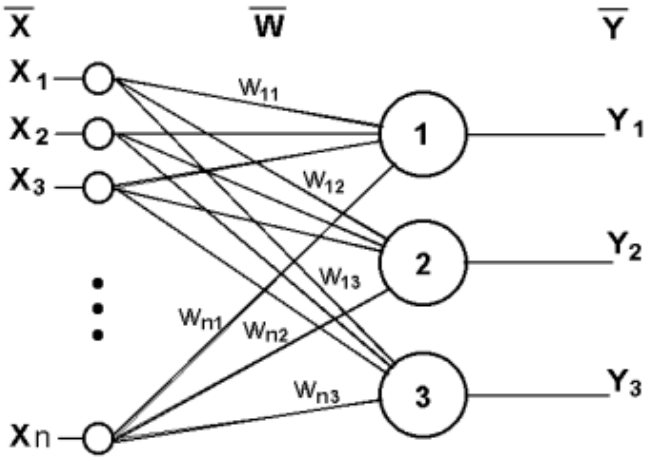

Рисунок 1.5 – Одношаровий перцептрон

Н обчислюють вихідні значення незалежно один від одного. Кількість синапсів у кожного з Н має бути рівною і збігатися з розмірністю вхідного сигналу, а розмірність виходу дорівнюватиме кількості Н.

1.2.3 Багатошаровий перцептрон

У багатошаровому перцептроні (рис.1.6), умови задачі сформульовані у вигляді множини векторів $\{x_1,..., x_S\}$, відповідно, мережа займається обчисленням значення деякої функції $Y = F(X)$, яка називається векторною. Саме рішення поставленого завдання набуває вид векторів $\{y_1,..., y_S\}$, і для всіх сум $S\, y_s = F(x_s)$.

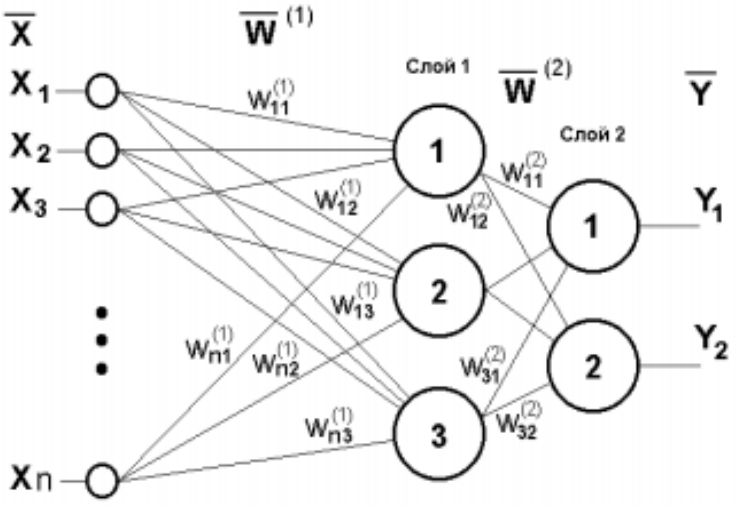

Рисунок 1.6 – Багатошаровий перцептрон

Перцептрон тільки формує відображення $F: X > Y$ для всіх $x \in X$. На людину покладається завдання формалізації, тобто саме людина вибирає сенс, яким будуть наділятися компоненти як вхідного, так і вихідного векторів. Варто відзначити, що на даний момент не існує якоїсь конкретної інструкції формалізації для ШНМ.

При побудові багатошарового перцептрона, необхідно дотримуватися такого алгоритму.

1. Встановити певний сенс, який буде вкладено в компоненти даного вектора X.

2. Встановити формалізовану умову виконуваного завдання для вектора входу, іншими словами, необхідно наділити вхідний вектор всією необхідною інформацією для отримання відповіді.

3. Встановити всі необхідні компоненти для вектора виходу Y, які містять повну відповідь для завдання, яке було поставлене.

4. Визначитися з вибором відповідного виду функції активації Н.

5. Визначити кількість шарів і Н у шарі.

6. Визначити для входів та виходів їх діапазон зміни.

7. Призначити початкові значення пороговим рівнями і вагам. Варто відзначити, що початкові значення задаються випадково, але процес навчання ШНМ може сповільнитися, якщо ці значення будуть занадто великими або дуже маленькими.

8. Провести навчання ШНМ.

1.2.4 Сигмовидний Н

Навчання ШНМ вносить невелику зміну ваги на певних синапсах. Ця зміна призводить до невеликої відповідної зміни на виході з мережі. Саме ця властивість робить навчання можливим. Повторюючи ряд таких змін можна навчити мережу видавати правильний результат [2].

Проблема полягає в тому, що це, власне, не трапляється тоді, коли в мережі наявні перцептрони. Насправді, при незначній зміні ваги, а також зміщення якогось одного перцептрону в мережі деколи може трапитися таке, що

вихід цього перцептрону повністю перевернеться, наприклад, з 0 до 1. Цей факт може привести до того, що поведінка іншої частини мережі буде повністю змінюватися надзвичайно складно. А це, в свою чергу, значно ускладнює уявлення про те, як поступово модифікувати ваги з метою наближення мережі до жаданої поведінки.

Ця проблема долається шляхом введення нового типу штучного Н, званого сигмовидним Н. Сигмовидні Н схожі на перцептрони, але незначні зміни їх ваги і зміщення провокують тільки незначну зміну їх виходу (рис. 1.7). Це критичний факт, котрий дає змогу мережі сигмовидних Н вчитися [1].

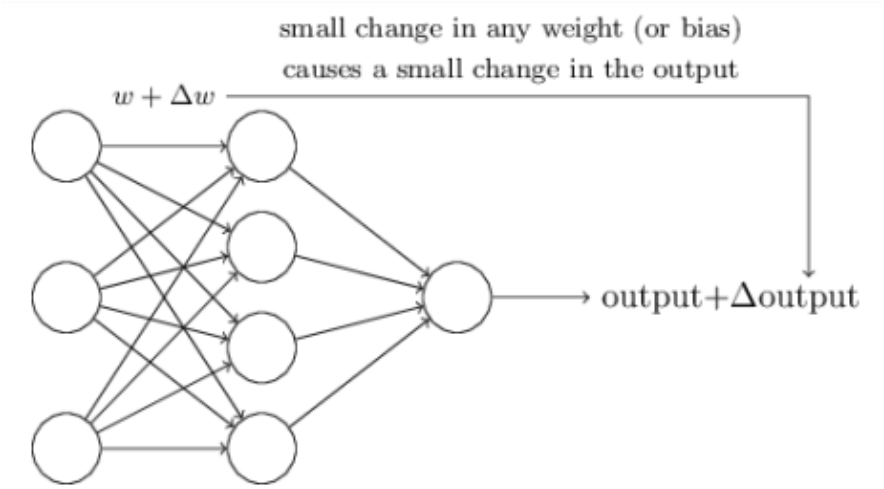

Рисунок 1.7 – Зміна ваги в сигмовидному Н

Такий Н, подібно до перцептронів, володіє входами x_1, x_2,.... Проте ці входи замість того, щоб приймати тільки 0 або 1, додатково можуть набувати будь-яких значень у проміжку від 0 до 1. Для прикладу 0,566 є допустимим входом для сигмовидного Н. Аналогічно як перцептрон, сигмовидний Н володіє вагами для кожного входу w_1, w_2,..., і порогом (зміщення) b. Проте на виході не 0 або 1. В нього замість цього є величина $\sigma\ (w * x + b)$, де σ називається сигмовидною функцією (формула 1.2).

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}. \tag{1.2}$$

Щоб показати все це більш явно, вихід сигмовидного Н з входами x_1, x_2,..., вагами w_1, w_2,... і b дорівнює (формула 1.3):

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}. \tag{1.3}$$

Варто зауважити, що точна форма σ не є надзвичайно важливою. Дійсно має велике значення форма функції при створенні графіка (рис. 1.8).

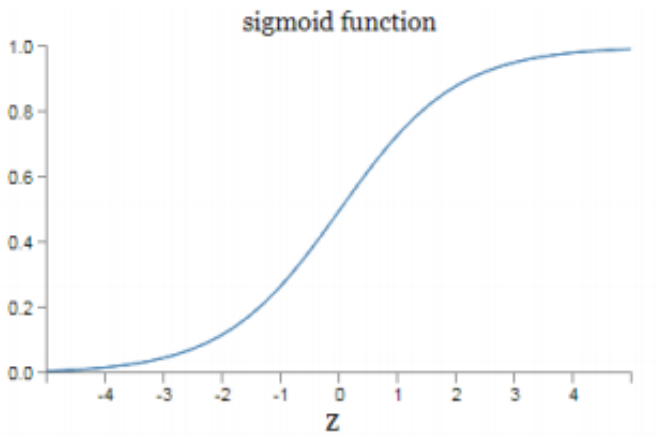

Рисунок 1.8 – Графік сигмовидної функції

Ця форма згладжена версія крокової функції (рис.1.9).

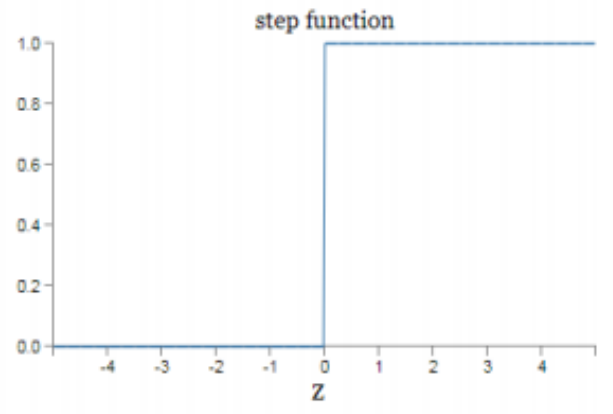

Рисунок 1.9 – Графік крокової функції

Якщо σ - функція кроку, тоді сигмовидний Н буде перцептроном, оскільки вихід буде дорівнює 1 або 0 в залежності від того, чи *w*x + b* був позитивним або негативним. Вище по тексту згадувалося, що за допомогою σ - фактичної функції можна отримати згладжений перцептрон. Насправді, саме гладкість функції є вирішальним фактом, а не її деталізована форма. Плавність σ означає, що незначні зміни Δw_j в вагах і Δb в порогах будуть виробляти незначні *Δoutput* зміни на виході Н. Фактично, обчислення говорить нам про те, що вихідна потужність output добре наближена до

$$\Delta \text{output} \approx \sum_j \frac{\partial \, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \, \text{output}}{\partial b} \Delta b, \tag{1.4}$$

де підсумовування ведеться за всіма вагами, w_j, *і* $\partial output / \partial w_j$ *і* $\partial output / \partial b$ позначають частинні похідні *output* щодо w_j *і* b, відповідно.

Δoutput є лінійною функцією змін Δw_j *і* Δb в вагах і порогах. Ця лінійність дозволяє легко вибрати незначні зміни в вагах і порогах для досягнення будь-якої бажаної незначної зміни на виході. Отже, сигмовидні Н мають таку ж якісну поведінку, як перцептрони, але вони спрощують зміну ваги і порога для зміни виходу Н.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Методи навчання ШНМ

Прийнято, що всі методи можна поділити на два класи:

– детерміновані;

– стохастичні.

Перші з них можуть ітеративно коригувати параметри ШНМ, грунтуючись на їх поточних значеннях, величинах входів, а також на фактичних і бажаних виходах. Другі методи випадково змінюють параметри мережі випадковим чином. При цьому зберігаються тільки ті зміни, котрі призвели до покращень. Варто зазначити, що для стохастичного методу навчання існує ймовірність потрапляння в «пастку» локального мінімуму.

2.1.1 Навчання з учителем

В такому алгоритмі навчання мережа володіє достовірними відповідями, які є виходами мережі, для кожного вхідного прикладу. Іншими словами, для мережі наперед вказується множина пар векторів *{(xi, di)}*, де xi ∈ X вектор, який визначає умову задачі, а di ∈ Y є наперед відомим розв’язком завдання для вектора xi. Під час проведення навчання мережа починає змінює власні параметри так, щоб в результаті видавати необхідне відображення X → Y.

2.1.2 Метод Хеба

Постулат навчання Хеба заснований на його гіпотезі про навчання біологічних Н. Хеб вважав, що посилення ваги з'єднання між двома Н відбувається за умови, що обидва Н порушено. Іншими словами, якщо Н, пов'язані між собою, активізуються власне одночасно і регулярно, то їх сила зв'язку буде збільшуватися. Варто відмітити, що особливістю цього правила полягає в тому, що сама зміна ваги перебуває в безпосередній залежності від активності Н, які з'єднані цим зв'язком.

Алгоритм:

– всім вагам присвоюються малі значення випадковим чином при ініціалізації;

– на вхід подається вхідний сигнал і після обчислюється вихід;

– вагові коефіцієнти змінюються на підставі вихідних значень Н;

– потім повторюється крок 2 з новими значеннями із вхідної множини, повторення відбувається, поки вихідні значення мережі не будуть стабілізовані з певною точністю, яка задається заздалегідь.

2.1.3 Правило корекції за помилкою

Модель, розроблена Розенблатом, застосовує алгоритм навчання з учителем, навчальний набір містить множину вхідних векторів, і кожен має вихідний вектор.

Кожному вхідному задається певний вихід. Після чого відбувається коригування параметрів мережі за умови, що реальний вихід мережі не збігся з бажаним. Для обрахунку корекції застосовується різниця між бажаним і реальним виходами мережі.

2.1.4 Навчання методом змагання

У методі Хеба безліч вихідних Н мають можливість збуджуватися одночасно, а в методі навчання змаганням вихідні Н змагаються один з одним за право активізації. Таким чином, використовується лише один Н з усієї множини вихідних Н, так само цей Н має найбільший вихід. Таке навчання мережі дозволяє зробити класифікацію вхідних даних: мережа групує схожі приклади в один клас і представляє їх одним зразковим елементом. Варто зазначити, що кожен Н з множини вихідних Н несе відповідальність лише за один клас. Цілком зрозуміло, що така мережа здатна працювати тільки з кількістю класів рівною кількості вихідних Н.

При такому навчанні відбувається модифікація ваги Н - переможця. Таким чином, зразковий елемент стає ближче до вхідного прикладу.

2.1.5 Метод зворотного поширення

Метою зворотного поширення є обчислення частинних похідних *∂C / ∂w і ∂C / ∂b* функції вартості *C* по відношенню до будь-якої ваги w або зміщення (порогу b в мережі. Функція *C* показана у формулі 2.1:

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2 \qquad (2.1)$$

де: сума по окремим прикладів навчання - *x; y = y (x)* - відповідний бажаний результат; *L* позначає кількість шарів в мережі; $a^L = a^L(x)$ - вектор активування, виведений з мережі при введенні *x*.

Для обчислення частинних похідних $\partial C/\partial w_{jk}^L$ *і* $\partial C/\partial b_j^L$. Але для їх обчислення, спочатку вводиться проміжна величина δj^L, яка називається помилкою в j Н в шарі *L*.

Щоб зрозуміти, як визначається помилка, потрібно уявити, що в ШНМ є демон (рис. 2.1) [1].

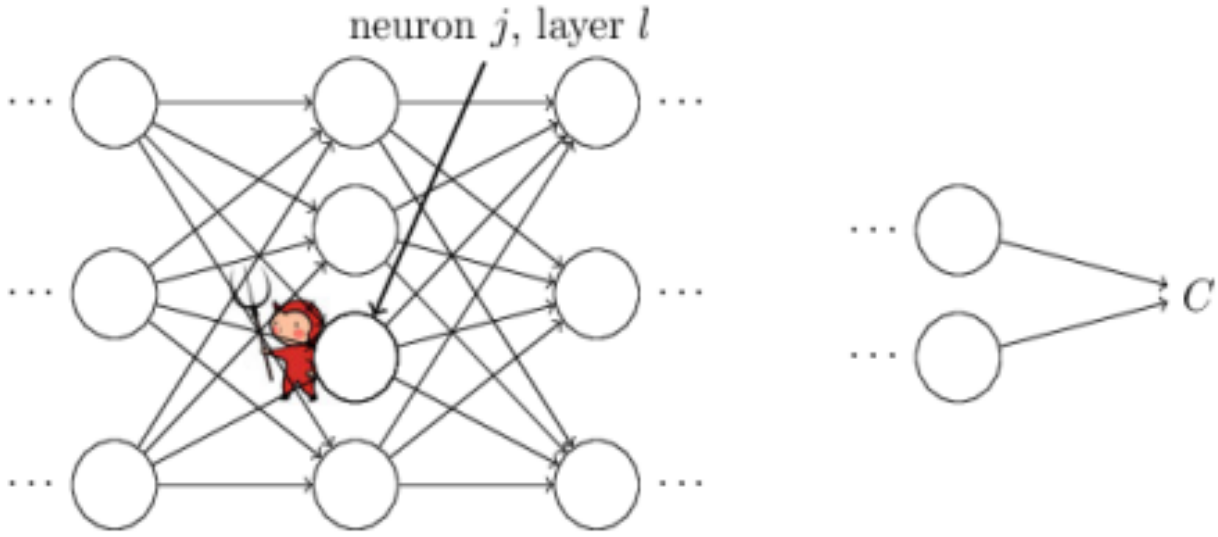

Рисунок 2.1 – ШНМ з демоном на *j* Н в шарі *L*

Демон сидить в *j* Н в шарі *L*. Коли на Н надходять вхідні дані, демон

безладно працює з Н. Він додає невелику зміну Δzj^L до зваженого входу Н, так що замість виводу $\sigma(zj^L)$ Н виводить $\sigma(zj^L + \Delta zj^L)$. Ця зміна поширюється через більш пізні шари в мережі, в результаті чого загальна вартість змінюється на величину $(\partial C / \partial z\, j^L) * \Delta z\, j^L$.

Потім цей демон стає хорошим демоном і намагається допомогти знизити вартість, тобто він намагається знайти Δzj^L, що робить вартість меншою. Якщо $\partial C / \partial zj^L$ має велике значення (позитивне чи негативне), тоді демон може трохи знизити вартість, вибравши Δzj^L, щоб мати протилежний знак $\partial C / \partial zj^L$. Навпаки, якщо $\partial C / \partial zj^L$ близька до нуля, то демон не може взагалі значно поліпшити вартість, оскільки йому може здатися, що Н вже досить близький до оптимального.

І тому існує евристичний сенс, в якому $\partial C / \partial zj^L$ є мірою помилки в Н (формула 2.2).

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}. \tag{2.2}$$

Метод базується на 4-х фундаментальних рівняннях. Всі разом вони дають нам спосіб обчислення як помилки δ^L, так і градієнта функції вартості.

1. Рівняння (2.3) для помилки в вихідному шарі δ^L:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L). \tag{2.3}$$

Перший член праворуч, $\partial C / \partial zj^L$, просто вимірює, як швидко змінюється вартість в залежності від активації j Н. Другий доданок праворуч $\sigma'(\partial z\, j^L)$ вимірює, наскільки швидко функція активації σ змінюється в zj^L.

Рівняння (2.3) є покомпонентним виразом для δ^L. Але для мережі потрібно використовувати рівняння (2.4) в матричній формі:

$$\delta^L = \nabla_a C \odot \sigma'(z^L). \quad (2.4)$$

Тут ∇aC визначається як вектор, компоненти якого є частинними похідними $\partial C / \partial aj^{L.}$

2. Рівняння (2.5) для помилки δ^L в елементах в наступному шарі, $\delta^{L} + 1$:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad (2.5)$$

де $(w^{l+1})^T$ - транспонування вагової матриці w^{l+1} для (l + 1) - ого шару. Потім ми беремо добуток Адамара $\odot\sigma'(z^l)$. Це переміщує помилку назад через функцію активації в шарі l, даючи помилку δ^L в зваженому вході в шар l.

Об'єднавши рівняння (2.3) і (2.5) чи (2.4), можна обчислити помилку δ^L для будь-якого рівня в мережі. Спочатку необхідно використати формулу (2.3) для обчислення δ^L, потім застосовуємо рівняння (2.5) для обчислення δ^{L-1}, потім рівняння (2.5) знову обчислює δ^{L-2} і т. д.

3. Рівняння (2.6) для швидкості зміни вартості відносно будь-якого зсуву в мережі:

$$\frac{\partial C}{\partial b} = \delta, \quad (2.6)$$

4. Рівняння (2.7) для швидкості зміни вартості по відношенню до будь-якої ваги в мережі:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j. \quad (2.7)$$

Формула (2.7) дозволяє обчислити частинні похідні $\partial C/\partial w\ jk^L$ в термінах

величин δ^L *і* $a^{L-1,}$ які вже відомо, як обчислити. Рівняння (2.7) можна переписати в меншому індексованому значенні – отримається формула (2.8):

$$\frac{\partial C}{\partial w} = a_{\text{in}}\delta_{\text{out}} \qquad (2.8)$$

Хорошим наслідком рівняння (2.8) є те, що коли активація a_{in} мала, $a_{in} \approx 0$, градієнтний член $\partial C/\partial w$ також буде малий. У цьому випадку вага вчиться повільно. Це означає, що вона не сильно змінюється при градієнтному спуску. Іншими словами, одним із наслідків рівняння (2.7) або (2.8) є те, що ваги, отримані з Н, котрі низько активізовуються, вчаться повільно.

Алгоритм виконання:

– введення x: встановлюється відповідна активація a^1 для вхідного шару;

– для кожного $l = 2,3,..., z^l = w^l a^{l-1} + b^l$ і $a^l = \sigma(z^l)$;

– помилка виведення $\delta^{l:}$ обчислити вектор $\delta^l = \nabla aC \odot \sigma'(z^l)$;

– помилка зворотного поширення: для кожного l = L-1, L-2,..., обчислюється $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$;

– вихід: градієнт функції вартості задається $\partial C/\partial w_{jk}^l = a_k^{l-1} \delta_j^l$ і $\partial C / \partial b_{jk}^l$

2.1.6. ГА

Девід Гольдберг в 1989 році запропонував таке визначення: "ГА є алгоритмами пошуку на основі механіки природного відбору і природної генетики". Цей метод поєднує в собі Дарвінівську теорію еволюції на основі виживання найбільш пристосованих середовищ "штучних істот" зі структурованим, але рандомізованих обміном інформацією [4].

Що стосується природної еволюції, то процес можна розкласти на три компактні частини [13].

1. Ініціалізація складається з налаштування різних параметрів (наприклад, чисельність агентів) і визначення власного зразка кодування. Також важливою частиною є формування фітнес - функції. Потім створюється

початкове покоління. Ініціалізація відбувається тільки на початку і всього один раз в ході роботи алгоритму

Параметри. Спочатку необхідно задати визначену кількість параметрів - чисельність населення, ймовірність мутації, кількість поколінь і т. д.

– чисельність населення - це кількість окремих особин в популяції. В кожній ітерації популяція має однакову кількість особин;

– кількість поколінь - число нових наборів особин, які повинні бути відтворені з поточного;

– імовірність мутації - це ймовірність того, що буде задіяна операція мутації в етапі відтворення.

Фітнес – функція – формула для ранжування особин в межах одного покоління. Чим вище функціональне значення фітнес -функції, тим вище поточний індивід ранжується. Функція формується таким чином: F (індивід) = кількість правильно розташованих чисел в закодованому індивіді / довжина закодованого фізичного індивіда. Чим правильніше розташовані цілі числа в рядку, тим ближче значення придатності рядка до 1. Значення придатності = 1.

2. Вибір - це підпроцес, коли певні особи в поточному поколінні оцінюються і ранжуються. Послідовно кожному з них присвоюється ймовірність вибору. Вибір проводиться один раз для покоління. Обчислення фітнес - функції.

Рейтинг населення з фітнесу. Наступним кроком є сортування окремих особин з фітнесу. Завдання фітнес - функції полягає в знаходженні кращої особини, в поточному поколінні, і надання їй більш високого шансу бути відтвореною в наступному поколінні.

Призначення ймовірностей окремим особам. Підсумкове завдання етапу відбору тісно пов'язана з ранжуванням населення. Існує кілька методів відбору окремих особин. Одним з них є метод рулетки.

Метод рулетки позначає підхід, при якому кожній особині присвоюється ймовірність вибору виходячи з значення фітнес – функції. Чим вище значення придатності, тим вище ймовірність того, що особина буде обрана.

3. Відтворення є частиною, відповідальною за вибір пари особин відповідно до можливостей відбору перш за все. Потім вибираються особини шляхом застосування генетичних операцій схрещування і мутації.

4. Результат - нові особини стають учасниками наступного покоління. Відтворення триває до тих пір, поки наступне покоління не досягне такого ж розміру, як поточне.

Популяцією називається набір векторів $P = \{pi\} = \{p1,..., pN\}$, тут N розмірність популяції. Елементи pi є особинами популяції. Кожен з векторів pi має всі параметри, необхідні для опису кожної особини. Функція $E(p)$ обчислює помилку. Треба знайти мінімум E.

ГА мають три основних операції [12]:

– розмноження. Якщо $E(p0)$ мале, то особина $p0$ вважається вдалою і буде мати при розмноженні пріоритет. У випадку, коли $E(p0)$ велике, то особина $p0$ вважається невдалою. Чим нижче $E(p)$, тим більша ймовірність виживання особини p.

– схрещування. Це процес, який намагається штучно відтворити процес схрещування ДНК двох батьків для створення ДНК дитини. Цей процес можна розділити на два етапи: вибирається пара батьків; вибираються місця змішування.

– мутація. Яка-небудь особина може піддаватися мутації, це означає, що значення може бути зміщене на малу величину $p0 = p0 + \Delta p$, де Δp є малим за модулем вектором, котрий характеризує власне величину мутації.

При відсутності мутації найсильніший вид особин починає швидко домінувати, і еволюція особин сповільнюється, застряє на одному місці.

2.2 Опис середовища моделювання і агентів

Нехай існує середовище (рис.2.2), а в цьому середовищі також можуть існувати агенти різних видів, наприклад, агенти, що виконують роль збирачів і агенти, що виконують роль зібраного об'єкта.

Перший вид агентів умовно назвемо "збирачі", а другий "їжа". Збирачі

отримують інформацію про середовище навколо себе за допомогою сенсорів. Коли збирач досить близько підбирається до їжі, то вона переходить в стан "з'їденої" і видаляється з середовища, в цей момент збирач отримує очко за зібрану їжу, а у випадковому місці середовища генерується нова частинка їжі, якщо ж агент стикається з іншим агентом того ж типу, то кількість очок зменшується. Таким чином, головним і єдиним завданням агентів - збирачів є збір частинок їжі з уникненням зіткнення з іншими агентами - збирачами. Ефективність агента розглядається з кількості очок, які він набрав в ході ітерації [9].

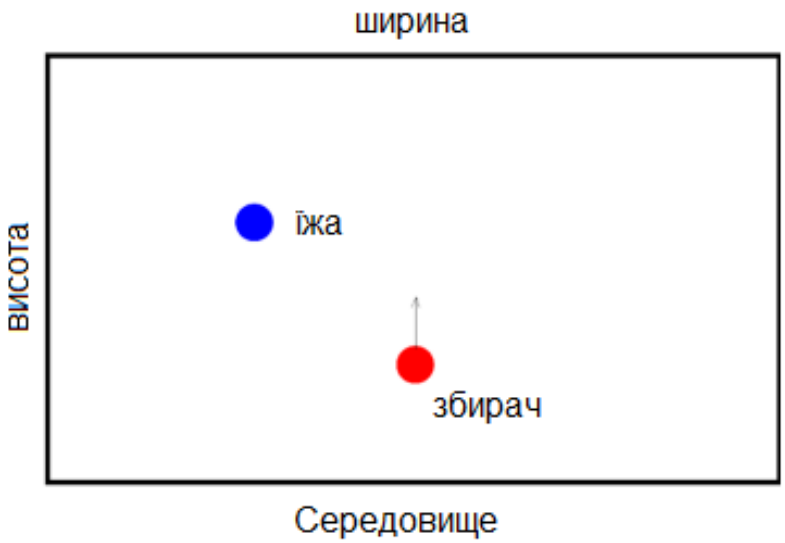

Рисунок 2.2 – Середовище

Для автоматичного пошуку оптимальної поведінки агентів - збирачів в конкурентному середовищі, алгоритм поведінки агентів буде сконструйований у вигляді нейронного середовища.

При моделюванні конкурентоспроможного середовища буде використаний ГА, і, відповідно, в середовищі буде проводитися ряд турнірів. Спочатку кожному агенту - збирачеві присвоюється своя ШНМ з випадковою зміною. Таку ШНМ можна умовно назвати "мозком агента" (рис. 2.3).

Протягом ітерації агент набирає і втрачає очки. При запуску еволюційного процесу, ітерація закінчується і проводиться порівняння отриманих агентами

очок. У підсумку, виграють нейронні мережі, чиї агенти зібрали найбільшу кількість їжі. На основі тих, хто виграв нейронних мереж, буде формуватися нова популяція, потім турнір проводиться знову.

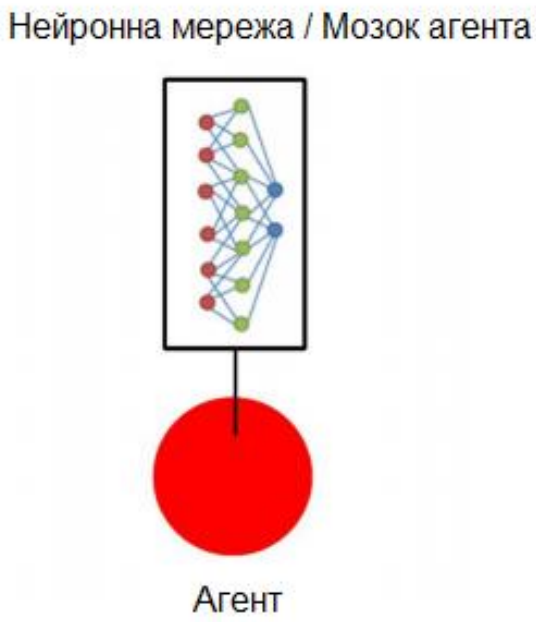

Рисунок 2.3 – "Мозок" агента

Використання ГА в моделюванні такого конкурентного середовища обумовлено відсутністю будь - яких теоретичних передумов в питанні того, яку саме топологію і кількість Н краще використовувати в рішенні даного завдання. Так само ГА є досить цікавою для вивчення топологією, інтерпретує еволюційну теорію. Для підтвердження вірного вибору реалізації можна проводити практичні експерименти з реалізацією інших алгоритмів, але, з теоретичної точки зору, вибір даної реалізації підкріплений тим, що ГА не вимагає від нас знань кінцевого результату для навчання мережі.

При моделюванні, в середовищі задаються такі параметри як ширина, довжина (для двовимірної моделі середовища), кількість агентів і кількість їжі. Також необхідно відмітити, що навколишнє середовище замкнуте, відповідно агенти не можуть виїхати за межі середовища.

Їжа має лише параметри координат, якщо розглядати двовимірну реалізацію середовища, то це параметри X і Y. Агент - збирач має своє

розташування в поточний момент часу (X, Y), свій вектор напрямку, радіус видимості і кут огляду.

Як згадувалося вище, агенти володіють сенсорами, які отримують від середовища інформацію про наявність поблизу їжі, про відстань до найближчої їжі, про наявність інших збирачів поблизу. Варто також відзначити, що агент здатний "бачити" тільки в певному радіусі, відповідно це є обмеженням здатності агента отримувати інформації про середовище. Дане обмеження є налаштованим і при бажанні можна змінити можливості видимості агента (рис. 2.4).

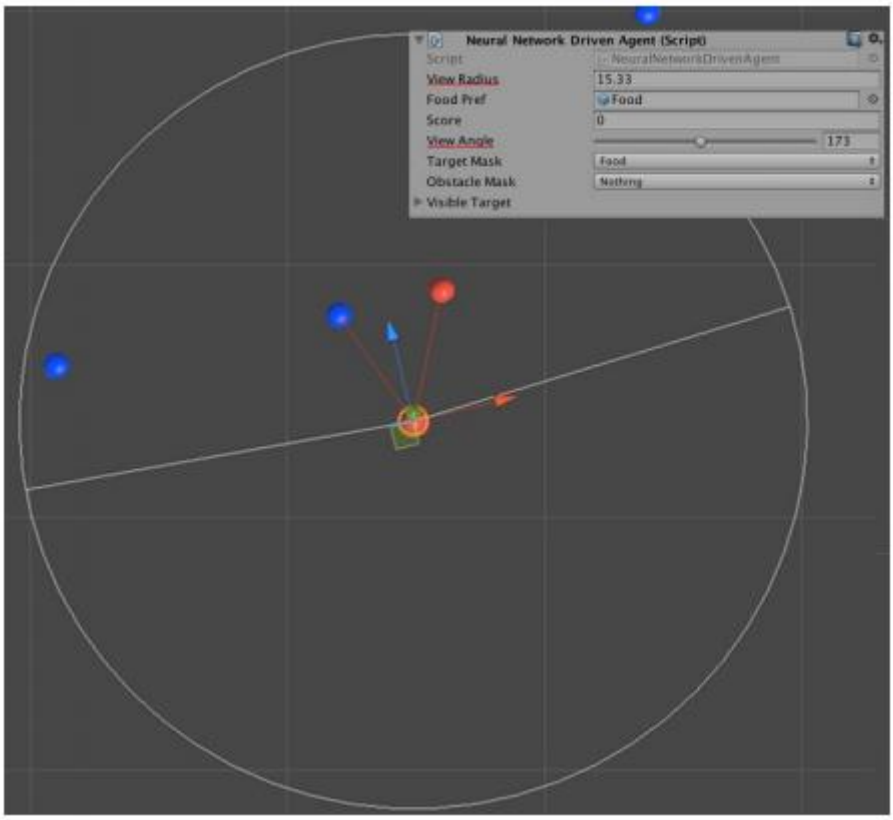

Рисунок 2.4 – Параметри налаштування агента

Робота ШНМ відбувається наступним чином: мережі отримують на вхід інформацію з сенсорів (рис. 2.5), після чого зчитується кут, на який агент повинен повернути, і обчислюється швидкість, з якою агент повинен рухатися. Таким чином, змінюючи свій напрямок і положення в просторі, агенти - збирачі взаємодіють із середовищем.

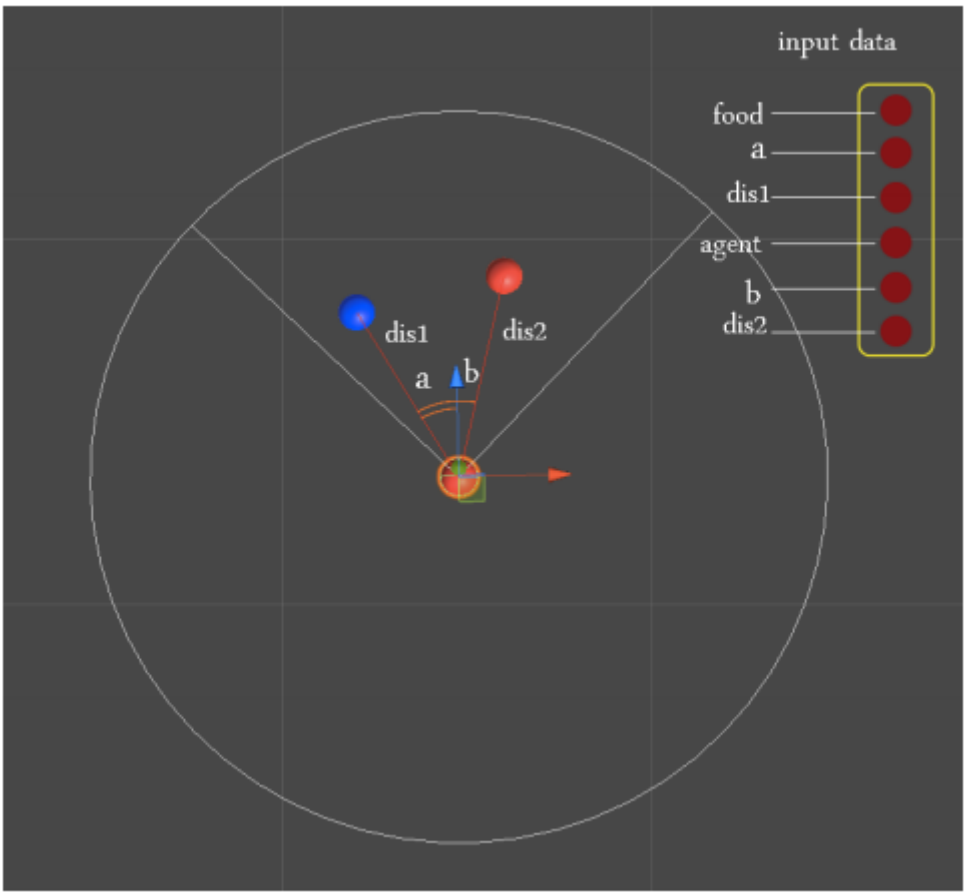

Рисунок 2.5 – Схема надходження інформації з сенсорів на вхід ШНМ

Описавши середовище моделювання і еволюцію агентів в цьому середовищі, можна висунути деякі припущення або гіпотези.

Виходячи зі складності завдання, що стоїть перед агентами, можна припустити, що ШНМ, котра володіє оптимальною конфігурацією, вийде досить швидко, незважаючи навіть на використання мутацій в ГА, які підвищують різноманітність одержуваних мереж.

Також можна припустити, що при невеликому ускладненні поставленого завдання, отримана ШНМ з підсумкової оптимальною конфігурацією буде мало пристосована до несподіваних змін середовища і можливо покаже гарні результати. Прикладом таких змін може бути додавання агентам, які грають роль їжі, здатність пересування.

Варто пам'ятати, що агенти функціонують в обмеженому, недетермінованому, та такому середовищі, яке частково спостерігається. Якщо сказати по іншому, середовище має свої межі, їжа ініціалізується в випадкових

позиціях, і агенти не мають повної інформації про всі одиниці їжі та інших агентів в середовищі. Виходячи з цієї інформації можна припустити, що в процесі еволюції будуть вигравати саме ті ШНМ, які найкраще будуть боротися з обмеженнями середовища. Головними обмеженням агентів - збирачів є швидкість і радіус огляду. Важко уявити, як мережа буде боротися з обмеженням огляду агентів, але висока ймовірність того, що саме такі мережі, здатні будь-яким способом домогтися збільшення огляду, будуть переможцями в проведених турнірах.

Після отримання оптимальної конфігурації ШНМ, також можна припустити, що поведінка популяції агентів - збирачів буде нести стадний характер (рис. 2.6). Це може бути обумовлено тим, що на даному етапі кожен агент буде володіти схожою, по конфігурації, з іншими агентами ШНМ. Отже, розбиваючись на кілька груп і переслідуючи найближчу частку їжі, агенти почнуть полювати натовпом за одним і тим же об'єктом, поки той не буде зібраний, після чого вся ця група переключиться на найближчу частку їжі і так далі.

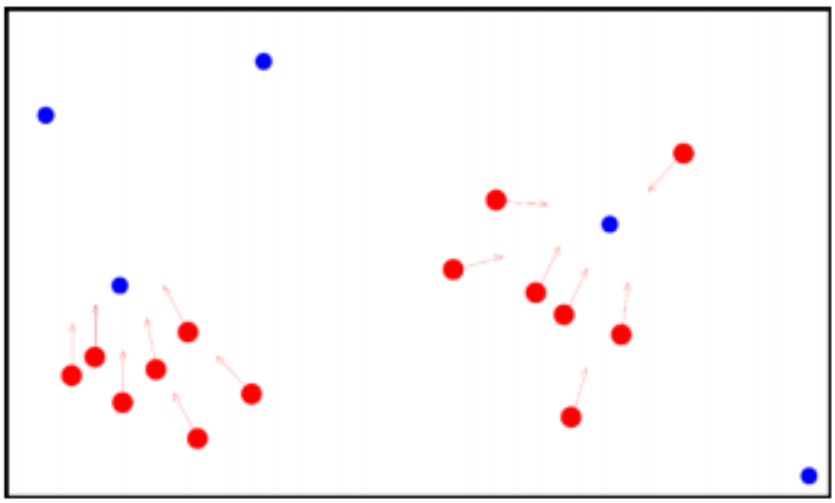

Рисунок 2.6 – Стадна поведінка агентів

Оцінку ефективності ГА можна розглядати з кількості проведених турнірів [11]. Якщо ГА оптимізував конфігурацію ШНМ за найменше число турнірів, тобто подальше проведення турнірів вносить зовсім непомітні коригування в

поведінку агентів, то такий алгоритм можна вважати ефективним. Також, алгоритм можна вважати ефективним, якщо він навчає ШНМ боротися з обмеженнями мережі, як говорилося вище.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБКИ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Вимоги до системи

Середовище, яке моделюється, повинно володіти наступними можливостями:

– налаштовувати агентів;

– налаштування середовища;

– еволюції агентів;

– збереження отриманої конфігурації;

– завантаження збереженої конфігурації.

Для реалізації системи був обраний ігровий рушій Unity3D. Дане середовище розробки має гнучкий базовий функціонал, який надає можливості щодо розв'язання завдань [10].

Переваги середовища Unity3D:

– надає готові методи візуалізації ;

–дозволяє уникнути в деяких моментах написання громіздких математичних обчислень;

– надає зручні можливості для організації налаштувань системи.

До недоліків можна віднести складність роботи з багатопотоковими процесами.

3.2 Опис архітектури

Для системи була розроблена архітектура (рис. 3.1). Її умовно можна поділити на такі 4 частини:

– environment (середовище);

– neural network (нейронна мережа);

– genetic algorithm (ГА);

– system controller (система контролю).

Середовище відповідає за зберігання та опис всіх агентів, нейронна мережа

описує структуру "мозку" кожного агента, ГА зберігає інформацію про популяції середовища і реалізує процес еволюції агентів, система контролю контролює всю систему і ініціалізує саме середовище.

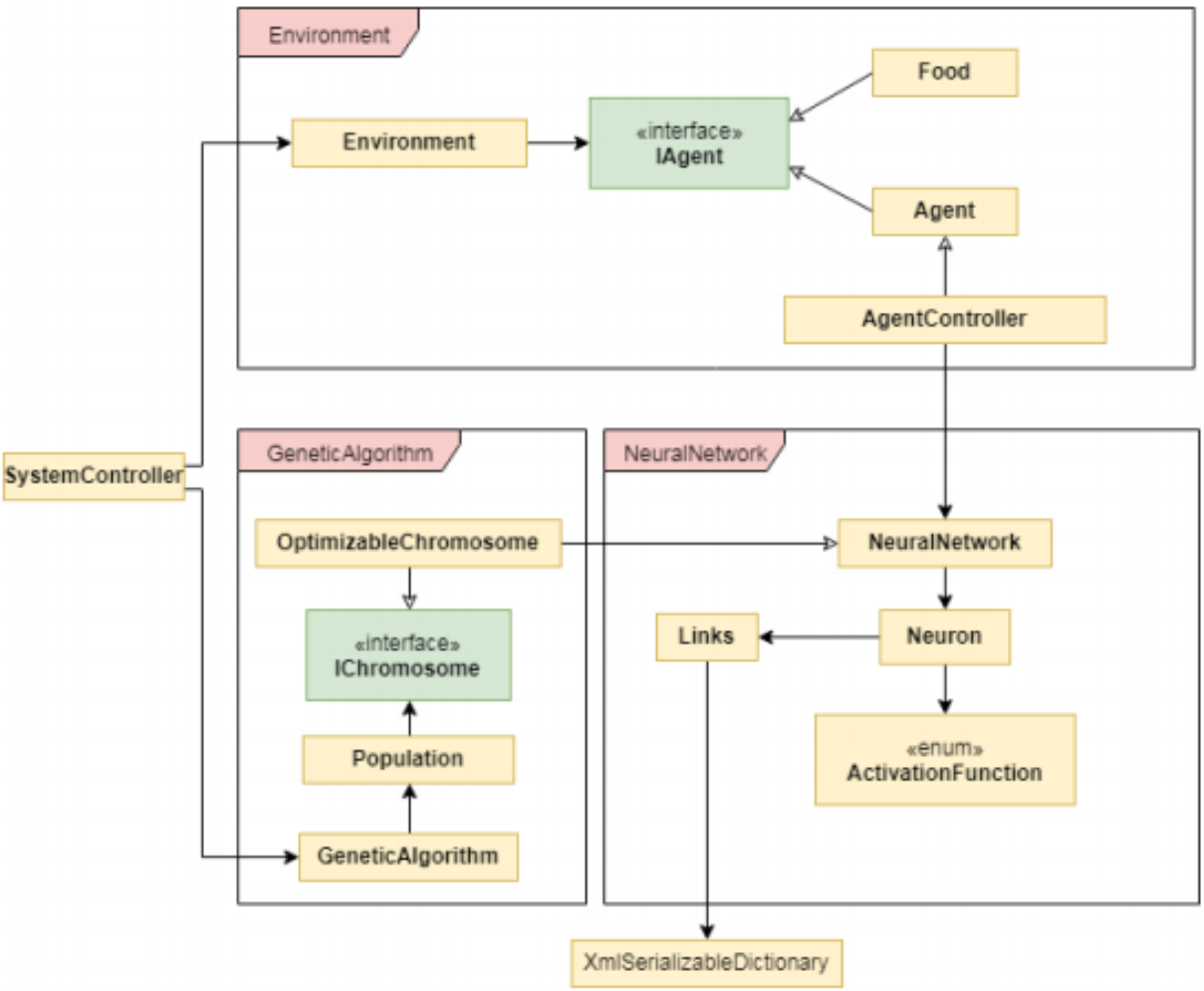

Рисунок 3.1 – Програмна архітектура системи

Розглянемо докладніше складові архітектури.

Environment.

IAgen - інтерфейс, що дозволяє реалізувати різні типи агентів, існуючих в середовищі (в подальшому дозволить створювати безліч типів агентів з різними цілями і завданнями).

Environment - клас, який описує середовище в якому знаходяться агенти. Зберігає список посилань на агентів і інформацію про довжину та ширину середовища.

Основні методи:

– public void TimeStep () - запускається кожен фрейм, запускає у агентів розрахунок даних нейронної мережі.

– private void AvoidMovingOutsideOfBounds (IAgent agent) - не дозволяє агентам виходити за рамки середовища.

– public List <GameObject> Filter (string tag) - дозволяє фільтрувати агентів по типу.

Food і Agent - класи, успадковують від інтерфейсу IAgent і описують два різних типи об'єктів.

Основний метод:

– public void Move () - відповідає за переміщення агентів.

AgentController - клас, який відповідає за розрахунок вхідних даних нейронної мережі і управління агентом на підставі цих даних.

Основні методи:

– public override void Interact (Environment env) - збирає інформацію з сенсорів агентів і посилає її на вхідні Н, запускається процес активації всіх Н і підрахунку підсумкової швидкості і кута повороту. Потім агент приводиться в рух на основі отриманих даних.

– private void ActivateNeuralNetwork (List <double> nnInputs) - відповідає за виклик активації Н.

– protected List <double> CreateNnInputs (Environment environment) відповідає за формування вхідних даних для нейронної мережі.

– public static OptimizableChromosome RandomNeuralNetworkBrain () - створення нейронних мереж з випадковою зміною.

NeuralNetwork.

Neuron - клас, який описує структуру Н і зберігає в собі функцію активації Н і її параметри.

Основний метод:

– public void Activate () - для розрахунку сигналу з використання функції активації.

Links - клас, який зберігає словник зі зв'язками між Н і значення ваг між

ними.

ActivationFunction - перерахування, що містить основні функції активації: сигмовидну, лінійну і сигнум.

Основні методи:

– public static ActivationFunction GetRandomFunction (this ActivationFunction fun) - створення випадкової функції з випадковим набором параметрів.

– public static double Calculate (this ActivationFunction f, double value, List <double> parameters) - розраховує дані відповідно з функцією.

NeuralNetwork - клас, що описує структуру нейронної мережі, це своєрідний "мозок" агента, який відповідає за його поведінку і реагування на середовище, зберігає список агентів і зв'язків між ними.

Основний метод:

– public void Activate () - активує Н і передає сигнали від Н до Н.

GeneticAlgorithm.

OptimizableChromosome - клас, який відповідає за оптимізацію нейронної мережі.

Основні методи:

– public List <OptimizableChromosome> Crossover (OptimizableChromosome anotherChromosome) - для схрещування ваг між Н або самих Н в нейронних мережах.

– public OptimizableChromosome Mutate () - для зміни ваг між Н або параметрів у функціях.

IChromosome - інтерфейс для реалізації хромосом - оптимальних нейронних мереж.

Population - клас, який зберігає популяцію середовища.

Основний метод:

– public void SortPopulationByFitness (Comparer <C> chromosomesComparator GeneticAlgorithm) - сортування хромосом.

GeneticAlgorithm - клас, який реалізує ГА.

Основні методи:

– public void Evolve () - відповідає за еволюцію агентів шляхом схрещувань і мутацій.

– private class ChromosomesComparator: Comparer <C> - клас, який потрібен для порівняння хромосом однією з одною.

XmlSerializableDictionary - клас, який реалізує словник для серіалізації.

SystemController - клас, що моделює і контролює систему.

Основні методи:

– private void InitializeEnvironment (int environmentWidth, int environmentHeight, int agentsCount, int foodCount) - створення середовища.

– public void InitializeAgents (List <OptimizableChromosome> brain, int agentsCount) - ініціалізації агентів.

– public void InitializeFood (int foodCount) - ініціалізації "їжі".

– private static void InitializeGeneticAlgorithm (int populationSize, int parentalChromosomesSurviveCount, OptimizableChromosome baseNeuralNetwork) - ініціалізації ГА.

– public void ApplayBestBrain () - доволяє застосувати найкращу хромосому до всіх агентів.

– public void EvolveAgents () - запускає еволюцію агентів.

– public void SaveBrain () - зберігає кращу хромосому в xml файл.

– public void LoadBrain () - дозволяє завантажити хромосому з xml файлу.

3.3 Алгоритм роботи програми

Для початку роботи системи задаються такі параметри як кут видимості, радіус огляду агентів, кількість їжі і агентів - збирачів, розміри середовища (рис. 3.2). Є також приховані параметри, які відповідають за обмеження кута повороту, швидкості агентів, кількості ітерацій при одній мутації [12].

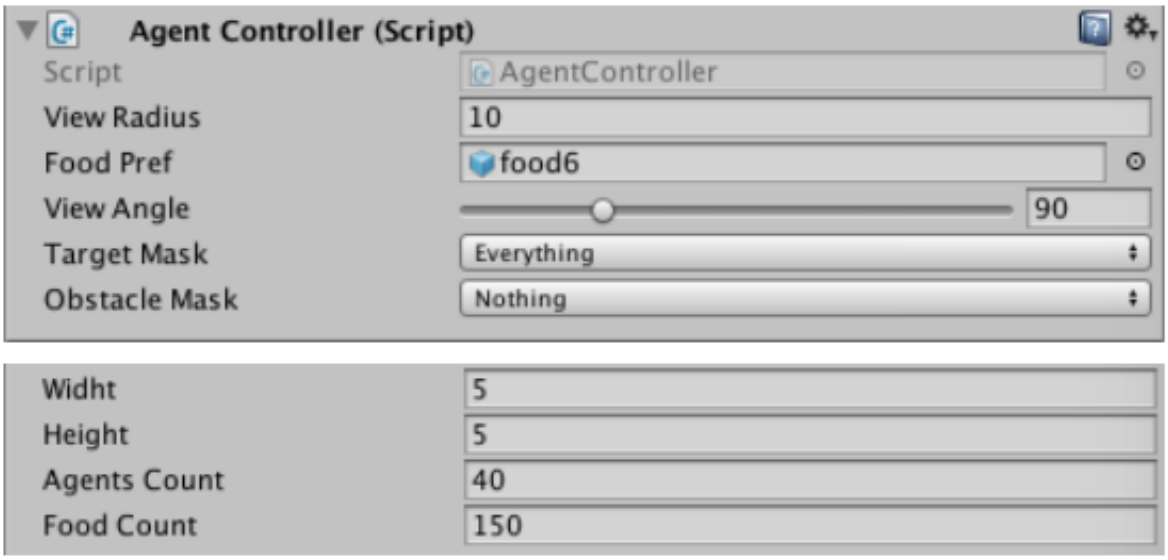

Рисунок 3.2 – Початкові параметри

Вподальшому ініціалізується ГА, який генерує популяцію хромосом, кожна з яких має свою нейронну мережу.

Потім ініціалізується середовище з заданими раніше параметрами, у випадковому порядку виставляються агенти - збирачі і їжа.

Подальша робота програми залежить від користувача. Після ініціалізації середовища наступні кроки можна виконати з меню програми (рис. 3.3).

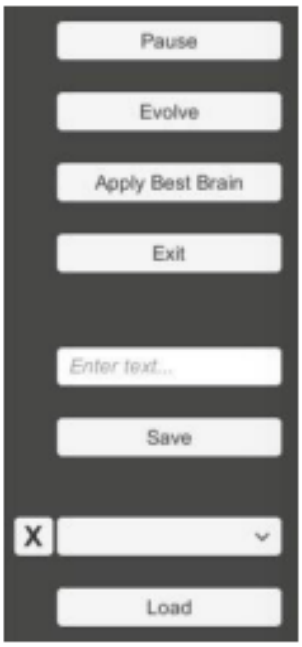

Рисунок 3.3 – Меню додатка

Pause запускає і зупиняє роботу агентів.

Evolve - запускає процес еволюції агентів, що призводить до отримання нової популяції, заснованої на кращій хромосомі. Спочатку проводиться сортування хромосом за кількістю набраних очок в ході однієї ітерації. Хромосома, яка набрала найбільшу кількість очок, стає кращою і використовується для схрещування з іншими хромосомами. Можливо кілька варіантів схрещування, які вибираються випадковим чином:

– випадковий набір ваг міняється місцями з набором іншої хромосоми;

– ваги однієї хромосоми змінюються з вагами іншої хромосоми у випадковому порядку;

– випадковий набір H хромосоми міняється місцями з набором іншої хромосоми;

– H однієї хромосоми міняються місцями з H іншої хромосоми.

Після схрещування відбувається процес мутації. Мутація так само вибирається випадковим чином з декількох можливих варіантів:

– відбувається зміна значень випадкового, упорядкованого набору ваг;

– зміна параметрів функції випадкових H;

– перемішування випадкового набору ваг.

Apply Best Brain - дозволяє застосувати найкращу хромосому до всіх інших агентів.

Save - дає можливість зберегти кращу хромосому у вигляді xml файлу з ім'ям, введеним в поле вище.

Load - дає можливість завантажити xml файл, який містить хромосому, зі списку.

Deleted [X] – дає змогу видаляти файли в списку.

3.4 Експерименти

Було проведено ряд експериментів для середовища з різним набором параметрів і конфігурацій мережі для отримання інформації про параметри та результативності роботи системи. Нижче наведені найбільш значущі для

виведення експерименти.

Експеримент 1. Проводився з середніми розмірами середовища і великою кількістю агентів та їжі. Так само перевірялося вплив випадкової конфігурації нейронних мереж.

Параметри системи:

- розмір середовища: 50x50;
- кількість агентів - збирачів: 50;
- кількість " їжі ": 200;
- радіус видимості: 5;
- кут огляду: 180;
- кут максимального повороту: 10;
- максимальна швидкість: 4;
- бал за збір їжі: 1;
- штрафний бал за зіткнення: -0.5.

При ініціалізації агентів, нейронні мережі отримували наступну конфігурацію: перші 6 вхідних Н наділялися лінійною функцією, інші 9 отримували випадкову функцію і трьох можливих. Процес еволюції включав в себе всі методи схрещування і всі методи мутацій. В ході схрещування зберігалася краща хромосома і на її основі створювалися нові шляхом схрещування з будь-хромосомою з популяції, вибір відбувався випадково. Таким чином на виході можна було отримати нейронну мережу, котра повністю відрізняється від вихідної.

Таймер, написаний для експерименту, працював наступним чином: кожні 30 секунд відбувалася еволюція агентів і кожну 5 ітерацію всім агентам присвоювалась найкраща хромосома на даний момент. Дані про кращу і гіршу хромосоми записувалися в окремий файл на кожній ітерації.

За підсумками першого експерименту було помітно, що велика кількість агентів заважають один одному набирати очки, навіть через велику кількість ітерацій, для агентів в більшому пріоритеті залишається пошук їжі, а спроби ухиляння важко здійсненні через велике число агентів, що призводить до отримання штрафних балів. Варто відзначити, що часто виникають ситуації,

коли агенти збиваються в купи в місцях, де їжі майже не залишилося, оскільки вона була згенерована приблизно в одній області середовища, в таких випадках спостерігається різке зменшення кількості набраних агентами очок. Такий же спад спостерігається і при присвоєнні кожному агенту однієї і тієї ж хромосоми. Спостерігається різка зміна поведінки агентів, яка призводить до того, що вони починають збиватися в купи або ходять змійкою один за одним.

Експеримент 2. Невеликий розмір середовища, менша щільність агентів.

Параметри системи:

- розмір середовища: 30 х30;
- кількість агентів - збирачів: 30;
- кількість " їжі ": 150;
- радіус видимості: 5;
- кут огляду: 180;
- кут максимального повороту: 10;
- максимальна швидкість: 4;
- бал за збір їжі: 1;
- штрафний бал за зіткнення: -0.5.

Принцип роботи методу еволюції і таймера залишився таким же, як і в першому експерименті. У другому експерименті перевірялося вплив щільності наповнення середовища агентами. Результати вийшли трохи краще, ніж у першому експерименті, але основні риси результатів спостереження залишилися такими ж, надалі було вирішено зменшити щільність агентів в середовищі.

Експеримент 3. Розміри середовища трохи збільшені, зменшена щільність агентів в середовищі.

Параметри системи:

- розмір середовища: 40х40;
- кількість агентів - збирачів: 20;
- кількість " їжі ": 100;
- радіус видимості: 5;
- кут огляду: 180;
- кут максимального повороту: 10;

– максимальна швидкість: 10;
– бал за збір їжі: 1;
– штрафний бал за зіткнення: -0.5.

На відміну від другого експерименту був збільшений розмір середовища і зменшено кількість агентів в середовищі, що знизило щільність агентів в ігровому просторі. Так само була змінена робота таймера, умова про застосування однієї хромосоми до всіх агентам, кожні 5 ітерацій, було видалено. Це дозволило уникнути частих скупчень агентів в одному місці в ході еволюції агентів. Так само була збільшена максимальна швидкість агентів, що не дало відчутних плюсів, можливо навіть погіршило роботу агентів, але в цьому питанні не можна робити конкретних висновків. У підсумку, кількість різких спадів очок, котрі набираються, зменшилася, але вони як і раніше мають місце бути.

Експеримент 4. В цьому експерименті перевірялася сувора конфігурація нейронної мережі, яка зберігається до кінця роботи програми. Змінено метод еволюції агентів.

Параметри системи:

– розмір середовища: 40 x40;
– кількість агентів - збирачів: 20;
– кількість " їжі ": 100;
– радіус видимості: 10;
– кут огляду: 90;
– кут максимального повороту: 45;
– максимальна швидкість: 5;
– бал за збір їжі: 1;
– штрафний бал за зіткнення: -0.5.

Збільшено максимальний радіус повороту агентів, трохи збільшена максимальна швидкість в порівнянні з початковою, кут огляду знижений до 90, радіус видимості збільшений до 10. Будова таймера така ж, як і в третьому експерименті, але час однієї ітерації збільшено до хвилини. Було вирішено змінити структуру нейронних мереж і процес еволюції. Нейронні мережі як і

раніше складаються з 15 Н: 6 вхідних, 7 у внутрішньому шарі і 2 на виході. Початкова структура тепер виглядає так: вхідні і вихідні Н містять лінійну функцію для розрахунку даних, а проміжні сигмовидну, від сігнум-функції вирішено було відмовитися, на користь більш класичного підходу з використанням тільки сигмовидної функції. Так само було вирішено зберегти загальну структуру нейронної мережі в ході еволюційного процесу, тому мутації, при яких функції між різними Н хромосом мінялися місцями, були видалені. Таким чином, на виході отримуємо нейронну мережу такої ж конфігурації, як і початкова, змінюються тільки значення параметрів функцій і ваг між Н мережі. При виборі змішування хромосом по черзі використовується кожна хромосома з популяції, а не випадково обрана.

Такі заходи спрощують мережу і виникає менше випадків різкого зниження очок. Агенти збиваються в купі тільки в разі потрапляння в порожню зону середовища, це обумовлено випадковою ситуацією, яка має місце бути, так само агенти стали діяти вільно.

Експеримент 5. Перевірка роботи системи при зміні пріоритету агентів.

Параметри системи:

- розмір середовища: 40 х40;
- кількість агентів - збирачів: 20;
- кількість " їжі ": 100;
- радіус видимості: 10;
- кут огляду: 90;
- кут максимального повороту: 45;
- максимальна швидкість: 5;
- бал за збір їжі: 0.5;
- штрафний бал за зіткнення: -1.

Середовище налаштоване так само, як і в четвертому експерименті. Є лише одна відмінність: штрафний бал тепер вищий, ніж бал за збір їжі. Експеримент проводився для того, щоб дізнатися чи почнуть агенти віддавати пріоритет ухиленню від інших агентів або так само зосередяться на зборі їжі.

За підсумком експерименту можна сказати, що агенти стали уникати

зіткнення трохи краще, але збір їжі залишився в пріоритеті. Можливо це пов'язано з тим, що як і раніше виникають ситуації скупчення агентів, в яких важко ухилитися від безлічі зіткнень, в той час як існують одиниці агентів, що знаходяться в більш сприятливих зонах середовища і займаються збиранням їжі, що і виводить їх на вершину еволюції.

Загалом хочеться відзначити цікаві властивості поведінки агентів в ході еволюційного процесу. Деякі агенти почали дивитися по сторонах під час свого руху і виявлялися більш ефективними, ніж ті агенти, які дивилися тільки в напрямку свого руху. Так само з'являлися особини, які, в разі відсутності їжі по близькості, починали оглядатися навколо, а не тільки по сторонах, обмеженими кутом максимального повороту. У початкових етапах можна було помітити, як деякі агенти вирішили рухатися не за їжею, а за іншими агентами. Якщо ж дозволити агентам рухатися не тільки вперед, але і назад, то це призводить до появи особин, які починали задкувати, побачивши інших агентів або зовсім намагалися збирати їжу виключно переміщаючись спиною вперед. Така можливість була неоптимальною і як наслідок не була включена в підсумкові випробування.

За підсумком експериментів варто відзначити, що висока щільність агентів в середовищі уповільнює їх еволюцію, але при цьому при занадто зниженій щільності частіше виникають порожні зони в середовищі, що призводить до скупчення агентів і різкого падіння показників зібраних очок. Тому слід шукати баланс в таких параметрах як розмір середовища, кількість їжі і агентів-збирачів.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Навчання працюючих і інструктажі з охорони праці

Однією із складових ефективної роботи з профілактики виробничого травматизму є належна підготовка, навчання та підвищення кваліфікації працівників з питань охорони праці. Загальний порядок проведення навчання з питань охорони праці встановлений Законом України «Про охорону праці» (ст. 18. «Навчання з питань охорони праці»).

Виконання вимог Закону України «Про охорону праці» в частині проведення навчання та перевірки знань з питань охорони праці здійснюється відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держкомітету України з нагляду за охороною праці 26 січня 2005 р. № 15 (далі — Типове положення).

Нагляд за дотриманням вимог Типового положення здійснюють органи державного нагляду за охороною праці, а координацію та методичний супровід — Головний навчально-методичний центр та навчальні підрозділи експертно-технічних центрів Держгірпромнагляду.

Вивчення предмета «Охорона праці» при підготовці, перепідготовці та підвищенні кваліфікації працівників, які залучаються до виконання робіт з підвищеною небезпекою, на підприємстві регламентується п. 2.3 Типового положення. На підприємствах згідно з п. 1.1 Додатку 3 Типового положення навчання та перевірку знань з питань охорони праці повинні проходити керівники, заступники керівників, головні спеціалісти, керівники основних виробничих та технічних служб, безпосередньо пов’язані з організацією безпечного ведення робіт. Крім цього, згідно з п. 5 Додатку 3, навчання та перевірку знань з питань охорони праці мають проходити керівники, спеціалісти служб охорони праці, члени комісій з перевірки знань з питань охорони праці, особи, відповідальні за технічний стан і безпечну експлуатацію об’єктів підвищеної небезпеки підприємств.

Типове положення встановлює порядок та місце проведення навчання та перевірки знань з питань охорони праці посадових осіб (п. 5.2 та п. 5.3). Посадові особи, перелік яких наведено в п. 5.2, проходять навчання у Головному навчально-методичному центрі Держнаглядохоронпраці. Перевірка знань цієї категорії посадових осіб проводиться комісією, створеною наказом Держгірпромнагляду.

Організацію навчання та перевірки знань з питань охорони праці працівників на підприємстві здійснюють працівники служби кадрів або інші спеціалісти, яким роботодавець доручив організацію цієї роботи. Навчання та перевірка знань з питань охорони праці працівників (виконавців і посадових осіб), які не залучаються до виконання робіт підвищеної небезпеки, проводиться не рідше ніж один раз на три роки. Посадові особи та працівники, які виконують роботи підвищеної небезпеки, проходять спеціальне навчання та перевірку знань відповідних нормативно-правових актів з охорони праці не рідше одного разу на рік.

Посадові особи малих підприємств (п. 5.4), які не мають можливості створити власні комісії з перевірки знань з питань охорони праці та провести навчання з питань охорони праці, проходять навчання та перевірку знань в навчальних закладах, які мають відповідний дозвіл.

Спеціальне навчання з питань охорони праці може проводитись безпосередньо на підприємстві або навчальним закладом, який має відповідний дозвіл. При проведенні такого навчання на підприємстві навчальні плани та програми розробляються з урахуванням конкретних видів робіт, виробничих умов і функціональних обов’язків працівників і затверджуються наказом керівника підприємства.

Періодичність інструктажів, навчання та перевірки знань з питань охорони праці залежить від видів виконуваних робіт та встановлюється Типовим положенням. Перевірка знань з питань охорони праці після проведення спеціального навчання проводиться комісією підприємства.

Якщо на підприємстві неможливо створити комісію з перевірки знань з питань охорони праці (п. 4.4 Типового положення), перевірка знань проводиться

комісією спорідненого підприємства або Теруправління Держгірпромнагляду.

Всі працівники та посадові особи підприємства, включаючи посадових осіб, відповідальних за виконання робіт підвищеної небезпеки (крім зазначених в п. 5.2 та п. 5.3 Типового положення), проходять навчання та перевірку знань з питань охорони праці на підприємстві. Типове положення не зобов'язує, але й не забороняє проводити навчання всіх виконавців та посадових осіб (особливо тих, що виконують роботи підвищеної небезпеки) в навчальних закладах. У нашій країні є багато підприємств, де таке навчання проводиться, і це має позитивні наслідки. Ті витрати, які при цьому несуть підприємства, перекриваються створенням більш безпечних умов праці і в результаті збереженням життя та здоров'я працівників.

Також в навчальних закладах проходять навчання та перевірку знань із загальних питань охорони праці всі посадові особи та фахівці, які проводять інструктажі або навчання підлеглих працівників з питань охорони праці, виконують роботи з проектування об'єктів, а також інші працівники, незалежно від того, передбачено таке навчання Типовим положенням чи ні.

4.2 Санітарно-гігієнічні вимоги до умов праці

На робочих місцях працівників, які відповідальні за експлуатацію сервісу управління механізмом авторських прав на мультимедійні файли, необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення для роботи з ЕОМ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24. Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН

3.3.6.042-99 в офісних приміщеннях, обладнаних ЕОМ, температура повітря повинна становити 22-25°С, відносна вологість повітря – 40-60 %, швидкість руху повітря – не більше 0,1 м/с [16].

Приміщення, призначені для роботи з ЕОМ, повинні мати природне освітлення. У виробничих приміщеннях, обладнаних ЕОМ, необхідно створити належне освітлення. При експлуатації сервісу управління механізмом авторських прав на мультимедійні файли, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18 [17]. Природне світло повинно бути бічним, зорієнтованим, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно-епідеміологічної служби. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути всеосяжним і рівномірним. У випадку, коли переважають роботи з документами, допускається комбіноване освітлення (додатково до загального освітлення встановлюється світильники місцевого освітлення). Світильники розміщуються збоку від робочих місць (переважно ліворуч), або локально над робочим місцем (при розташуванні відеотерміналів ЕОМ за периметром приміщення). Як джерело світла при штучному освітленні застосовуються, як правило, люмінесцентні лампи. У світильниках місцевого освітлення допускається застосування ламп розжарювання. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні

розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерною технікою не повинен перевищувати середньодобові концентрації [16].

Організація робочого місця фахівця із експлуатації сервісу повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Рівень шуму не повинний перевищувати: на місцях, де працюють програмісти та оператори ЕОМ, 55 дБА, у лабораторіях, де складаються алгоритми та ведеться робота з документацією – 60 дБА, у машинному залі – 65 дБА, у приміщеннях, де розміщені гучні агрегати обчислювальних машин – 75 дБА.

ВИСНОВКИ

Була розроблена система, що дозволяє створити середовище із заздалегідь заданими параметрами. Середовище містить в собі агентів, керованих нейронною мережею, навчання якої відбувається через ГА. ГА добре підійшов для завдання, кінцевий результат якого заздалегідь невідомий. Так само система надає користувачеві набір функцій для управління еволюцією агентів, збереження або завантаження нейронної мережі. Система розроблялася на ігровому рушію Unity 3D, що дозволило скористатися просунутими методами візуалізації, а також полегшити впровадження середовища в ігрові проекти, що дозволяє отримати наочний результат роботи не тільки у вигляді цифр.

В ході роботи було проведено певну кількість експериментів. Кожен експеримент переслідував певну мету і мав власні налаштування параметрів і конфігурації нейронної мережі, ГА або самих агентів. Це дозволило зробити ряд висновків по роботі самої системи і, на основі цих висновків, оцінити результати дослідження.

За підсумками роботи не можна сказати, що були отримані виключно позитивні результати. Часто відбуваються такі ситуації, коли агенти не роблять спроб уникнути зіткнення один з одним. Але, в цілому, агенти прагнуть виконати поставлене перед ними завдання і навіть роблять цікаві спроби подолання обмежень середовища.

В подальшому потрібно приділити більше часу і потужності на проведення однієї ітерації при проведенні експерименту, так само необхідно знайти більш чіткий баланс щільності агентів і, можливо, переглянути структуру нейронної мережі або методів навчання ГА.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Michael A. Nielsen, "Neural Networks and Deep Learning". How the backpropagation algorithm works [Електронний ресурс] – Режим доступа: http://neuralnetworksanddeeplearning.com/chap2.html - (дата звертання: 01.05.2021).
2. Филип Вассерман, Neural Computing Theory and Practice, 1989, pp. 23-20.
3. Smith L., An Introduction to Neural Networks, 2001, pp. 5-18.
4. Генетичні алгоритми. Ключові поняття та методи реалізації. URL: http://www.znannya.org/?view=ga_general (дата звертання: 21.05.2021).
5. Zurada J.M., Introduction To Artificial Neural Systems, 1992, p. 14.
6. Nigrin A., Neural Networks for Pattern Recognition, 1993, p. 10.
7. Haykin S., Neural Networks: A Comprehensive Foundation, 1994, p. 12.
8. Данила Васенков. Методы обучения искусственных нейронных сетей. [Електронний ресурс] - Режим доступа: http://www.ipo.spb.ru/journal/content/733/ - (дата звертання: 21.05.2021).
9. Юрий Лаходюк, Эволюция агентов управляемых нейронной сетью, [Електронний ресурс] - Режим доступа: https://habr.com/post/168067/ - (дата звертання: 20.05.2021).
10. Сравнительный анализ популярных движков разработки игр/ [Електронний ресурс] – Режим доступу: https://repetitora.com/sravnitelnyj-analiz-populyarnyh-dvizhkov-razrabotki-igr - (дата звернення 18.05.2021).
11. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М: Горячая линия, 2006. 452 с.
12. Олійник А.О., Субботін С.О., Олійник О.О. Еволюційні обчислення та програмування : Навчальний посібник. Запоріжжя : ЗНТУ, 2010. 324 с.
13. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М: Физматлит, 2006. 320 с.

14. Dopico J. Encyclopedia of artificial intelligence; Eds.: J.R. Dopico, J.D. de la Calle, A.P. Sierra. New York: Information Science Reference. 2009. Vol. 1–3. 1677 p.

15. Nilsson N., Introduction to Machine Learning, 1996, p. 13.

16. Козлов С.С. Методичні вказівки до виконання розділу “Охорона праці та безпека в надзвичайних ситуаціях” в дипломних проектах для підготовки студентів факультету електроніки за освітньо-кваліфікаційним рівнем “Спеціаліст” та ”Магістр”. "Вимоги безпеки під час експлуатації обчислювальної техніки" / К.:НТУУ ”КПІ”, 2015, - 30 с.

17. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 – (дата звертання: 01.06.2021).

ДОДАТКИ

ДОДАТОК А
Фрагмент програмного коду

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using System.Xml.Serialization;
using UnityEngine;
using UnityEngine.UI;

public class SystemController:
MonoBehaviour {
  [SerializeField]
  public GameObject _agentPref;
  public GameObject _foodPref;
  public GameObject _groundPref;
  public GameObject _camera;
  public Text _fileForSave;
  public Text _info;
  public Text _infoIteration;
  public Dropdown _fileForLoad;
  // [Range (5, 14)]
  public int widht;
  // [Range (5, 14)]
  public int _height;
  public int _agentsCount;
  public int _foodCount;
  static public Environment environment;
  private static GeneticAlgorithm <OptimizableChromosome, double>
  ga;
  public bool _play = false;
  int _populationNumber = 0;
  public int _iterCount = 1;
  public int _gaPopulationSize = 5;
  int _parentalChromosomesSurviveCount = 3;
  // public float timer = 0.0f;

  void Start ()
  {

    // пошук і додавання файлів , збережених
    раніше string path =
    Application.persistentDataPath; string
    [] files2 = Directory.GetFiles (@path,
    "* .xml");
List <string> filesDir1 = (from a in Directory.GetFiles (@path, "*
.xml") select
 Path.GetFileName (a)). ToList ();
    for (int i = 0; i <files2.Length; i ++)
    {
      _fileForLoad.options.Add (new Dropdown.OptionData (filesDir1
```

```
      [i]));
    }
    widht = widht * 10;
    _height = _height * 10;
    _gaPopulationSize = _agentsCount;
    InitializeGeneticAlgorithm (_gaPopulationSize,
    _parentalChromosomesSurviveCount, null);
    InitializeEnvironment (widht, _height, _agentsCount,
    _foodCount); StartCoroutine ( "Timer");
  }

  void Update ()
  {
     MainEnvironmentLoop ();
  }
  IEnumerator Timer ()
  {
    for (;;)
    {
      Activate ();
      yield return new WaitForSeconds (60f);
    }
  }

  public void Activate () {
    OnPause ();
    ga.SortPopulation ();
    int iter = ga.GetIteration ();
    string text = _infoIteration.text + "(" + ga.GetBest (). score
    + "," + ga.GetWorst (). score + ") \ n"; EvolveAgents ();
File.AppendAllText ( "/ Users / User / Documents / DML /
Unity_project / EvoNNAgent / Assets / Resources / test3.txt",
text);
    // if (iter% 5 == 0)
    // {
    // ApplayBestBrain ();
    //}
    OnPause ();
  }
  private void MainEnvironmentLoop ()
  {
    if (_play == true)
    {
      environment.TimeStep ();
    }
  }
  public void OnPause ()
  {
    if (_play == false)
    {
      _play = true;
      _info.text = "Play";
    }
    else
```

```
{
  _play = false;
  _info.text = "Pause";
}
}


private void InitializeEnvironment (int environmentWidth, int
environmentHeight, int agentsCount, int foodCount)
{
GameObject ground = Instantiate (_groundPref, new Vector3
(widht / 2, -0.5f, _height / 2), Quaternion.identity);
ground.transform.localScale = new Vector3 (widht / 10 + 0.1f,
1, _height / 10 + 0.1f);
_camera.transform.position = new Vector3 (widht / 2, _height *
8, _height / 2);
//_groundPref.transform.position = new Vector3 (widht / 2, -
0.5f, _height / 2);
environment = new Environment (environmentWidth,
environmentHeight);
List <OptimizableChromosome> brain = ga.GetPopulation ().
GetAllChromosomes ();
InitializeAgents (brain, agentsCount);
InitializeFood (foodCount);
}
public void InitializeAgents (List <OptimizableChromosome>
brain, int agentsCount) {
int environmentWidth = environment.GetWidth (); int
environmentHeight = environment.GetHeight (); for (int i = 0;
i <agentsCount; i ++) {
int x = Random.Range (0, environmentWidth);
int z = Random.Range (0, environmentHeight);
double direction = Random.value * 2 * Mathf.PI;
GameObject agentObj = Instantiate (_agentPref, new Vector3
((float) x, 0f, (float) z), Quaternion.AngleAxis (Random.Range
(0, 360), Vector3.up));
agentObj.GetComponent <AgentController> ().
SetParametersNeuralNetworkDrivenAgent (x, z, direction);
agentObj.GetComponent <AgentController> (). SetBrain (brain
[i]);
agentObj.GetComponent <AgentController> (). SetSpeed (0);
agentObj.name = "Agent" + i;
environment.AddAgent (agentObj);
}
}
…
```