МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

"Інформаційна система-тренажер для дисципліни "Теорія ігор" з використанням генетичних алгоритмів машинного навчання"

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Берест О.Б.**

Студент гр. ІН–72 — **Малежик В.А.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую _____

Зав. кафедрою Довбиш А.С.

“_____” _____2021 р.

Завдання

до випускної роботи

Студента четвертого курсу, групи ІН-72 спеціальності “Інформаційно-комунікаційні технології” денної форми навчання Малежика Віктора Андрійовича.

Тема: “**Інформаційна система-тренажер для дисципліни “Теорія ігор” з використанням генетичних алгоритмів машинного програмування**”

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів побудови інформаційної системи-тренажера; 2) постановка завдання й формування завдань дослідження; 3) огляд і опис засобів для розробки; 4) розробка інформаційної системи тренажера для “Теорії ігор” з використанням генетичних алгоритмів машинного навчання; 5) аналіз результатів.

Дата видачі завдання “_____”_____2021 г.

Керівник випускної роботи _____ Берест О.Б.

Завдання прийняв до виконання _____ Малежик В.А.

РЕФЕРАТ

Записка: 40 стор., 7 рис., 2 додатків, 10 джерел.

Об'єкт дослідження — Інформаційні процеси при використанні генетичних алгоритмів машинного навчання.

Мета роботи — розробити інформаційну систему-тренажер для “Теорії ігор” з використанням генетичних алгоритмів машинного навчання. Система-тренажер повинна виконувати маніпуляції для розвитку та проходження гри.

Результати — виконано вибір методів вирішення поставленої задачі; на основі генетичних алгоритмів кожне покоління розвивається і з кожною новою спробою заходить далі в системі-тренажері для “Теорії гри”.

СИСТЕМА-ТРЕНАЖЕР, ТЕОРІЯ ІГОР, МАШИННЕ НАВЧАННЯ, ГЕНЕТИЧНІ АЛГОРИТМИ, НЕЙРОННА МЕРЕЖА.

ЗМІСТ

ВСТУП

Загальний термін «Машинне навчання» означає безліч математичних, статистичних та вичислюваних методів які приймають участь в розробці алгоритмів, мають змогу вирішувати задачі не безпосереднім способом, а ґрунтуючись на пошук закономірностей у різноманітних вхідних даних. Результат обчисляється не за чіткою формулою, а з використанням встановленої раніше залежності результатів від певного набору ознак та їх значень. Впливає що машинне навчання може застосуватися для діагностики, прогнозування, розподілу та прийняття рішень у різних прикладних сферах: від медицини до банківської діяльності.

Існує безліч методів машинного навчання. перерахуємо найпопулярніші, залишивши їх детальну класифікацію спеціалізованим ресурсів.

В Машинному навчанні виділяють 2 класичні види:

Якщо функціональна залежність результатів входу і побудови алгоритму повинна бути знайдена, то застосовується вид з вчителем. Через визначення середньої помилки відповідей алгоритму за всіма об'єктами визначається функціонал якості. Завдання типу класифікації, регресії, ранжування і прогнозування відносяться до навчання з вчителем.

У випадку якщо відповіді не задаються та є потрібність в пошуку залежності між об'єктами то застосовується вид без вчителя. До цього складу входять завдання типу пошуку асоціативних правил, заповнення пропущених значень, фільтрації викидів,скорочення розмірності та кластеризації і побудови довірчої області. Некласичні, але найпопулярніші методи включають розподілене навчання, зокрема, генетичні алгоритми та штучні нейронні мережі. Пара працює як вхідний об’єкт “ситуація, рішення”, і її відповідь - значення функції якості, що представляє точність рішення

(реакція навколишнього середовища). Ці методи успішно застосовуються для формування інвестиційних стратегій, автоматичного управління технологічними процесами, самонавчання роботів та інших подібних завдань.

Машинне навчання на даний момент є дуже перспективною галуззю в сфері інформаційних технологій. Можна чітко сказати що машинне навчання є кроком до створення справжнього штучного інтелекту.

В роботі буде взято за основу генетичні алгоритми як метод машинного навчання.

Генетичний алгоритм - це еволюційний алгоритм, який використовується для систематичного відбору, поєднання та вирішення складових проблем та моделей, необхідних для біологічної еволюції.

Генетичний алгоритм характеризується спрямованістю на використання операторів "кросоверу", які виконуються в поєднанні з рішеннями-кандидатами, які відіграють однакову роль для гібридів дикої природи.

Спочатку ми дізнаємося, як створити гру. А потім за допомогою генетичних алгоритмів машинного програмування навчити комп'ютер виконувати вимоги завдання які поставлення для гравці в грі.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Короткий огляд системи-тренажера “Flappy Bird”

Flappy Bird - це аркадна гра, в якій гравець керує птахом, який наполегливо рухається вправо. Гравцеві доручено переміщатися по Фейбі через пари труб, які мають однакові за розміром зазори, розміщені на довільній висоті. Птах автоматично опускається вниз і піднімається лише тоді, коли плеєр натискає на сенсорний екран. Кожен успішний прохід через пару труб присуджує гравцеві одне очко. Зіткнення з трубою або землею закінчує ігровий процес. Під час гри за екран, гравець нагороджується бронзовою медаллю, якщо він досяг десяти і більше очок, срібною медаллю з двадцяти очок, золотою з тридцяти очок і платиновою медаллю із сорока очок.

В нашому випадку роль гравця буде сам компьютер, який буде методом спроб та помилок просуватися в прогресі системи-тренажера.

1.2 Алгоритми машинного навчання

1.2.1 Генетичні алгоритми

Призначення генетичних алгоритмів полягає у вирішинні завдання для оптимізації. Для прикладу подібної задачі можна взяти навчання нейросіток, якщо є підбірки, для яких досягнута мінімальна помилка. При цьому основою данного генетичного алгоритму слугує метод випадкового пошуку. Головним мінусом випадкового пошуку слугує те, що достеменно невідомий час який знадобиться для вирішення задачі.

Для уникннення такого зясовування часу під час вирішенні завдань, застосовуються методи, найдені під час вивчення еволюції та виникнення видів. Як відомо, процес випробування еволюції користуються найбільш єфективними особами. Це має наслідок в тому, що можливість сприятия росту популяції людей зростає, що дозволяє їй краще виживати в змінюваних умовах.

Вперший подібний алгоритм був запропонований у 1975 році Джоном Холландом (Джон Холланд) у Мічиганському університеті. Він отримав назву "репродуктивний план Холланда" і ліг в основу практично всіх варіантів генетичних алгоритмів. Однак, перед тим, як ми його розглянимо детальніше, необхідно зупинитися на тому, яким чином об'єкти реального світу можуть бути закодованні для використання в генетичних алгоритмах.

Представлення об'єктів

Біологія показує, що в реальному світі будь-яка істота може представляти єдине ціле, а генотип містить всю інформацію про організм у наборі хромосом. У цьому випадку, якщо є компонент інформації про генотип, то кожен ген відображається у фенотипі.

Отже, для вирішення проблеми все повинно бути представлене у

належній формі, щоб це можна було використовувати в генетичному алгоритмі. Усі додаткові функції рослини походять від генетичного складу за допомогою генетичних алгоритмів і часто використовуються для різноманітних функцій, оскільки вони не дають інформації про структуру бактерій.

Для представлення генотипу об'єкта застосовуються бітові строки. При цьому кожен приїжджий об'єкт у фенотипі відповідає одному гену в генотипі об'єкта. Ген - бітова строка, що містить все фіксировану довжину, що представляє собою значення цього визнання.

Кодирование признаків (цілих чисел)

Для кодування таких ознак можна використовувати найпростіший варіант - бітове значення цього визнання. Тогда нам буде в цілому просто використовувати ген визначеної тривалості, достатньої для представлення всіх можливих знайомих такого визнання. Але, на жаль, таке кодування не уникає своїх недоліків.

Основний недолік полягає в тому, що співвідношення числа відрізняється за значеннями кількох бітів, наприклад, числа 7 і 8 у бітовому представленні відрізняються в 4-х позиціях, що забезпечує функціонування генетичного алгоритму та збільшує час, необхідний для його ходімостей. Для того, щоб уникнути цієї проблеми, краще використовувати кодування, при якому співвідношення числа відрізняється меншою кількістю позицій, в ідеальному значенні одного біта.

Кодування ознак (дійсні числа)

Найпростіший спосіб кодування, який лежить на поверхні - використовуйте бітове представлення, хоча такий варіант має недоліки, що і для цілих чисел. Тому на практиці зазвичай застосовують наступну послідовність дій:

Розбивають весь інтервал допустимих значень призначень на участі з

необхідною точністю.

Вказують значення гена як цільове число, визначаючи номер інтервала (за допомогою коду Грея).

У якості значень параметрів приймається число, яке відображається середнім інтервалам.

Для того, щоб визначити об'єкт фенотипу (якщо є значення признаків, описуючих об'єкт), нам потрібно лише знати значення генів, що відповідають цим признакам, тому є об'єкт генотипу. При цьому совокупність генів, описуючих генотип об'єкта, представляє собою хромосому.

Таким чином, у реалізації генетичного алгоритму хромосоми представлена бітову структуру фіксованої довжини. При цьому кожен учасник строки відповідає гену. Длина генів у хромосомах може бути однорідною або різною. Чаще всього застосовують гени одинакової тривалості.

Як відомо, в теорії еволюції важлива роль грає це, яким чином визнання родителів передаються потомкам. У генетичних алгоритмах за передачу призначеним родителям потомкам відповідає оператор, який називається скрещиванием (його також називають кроссовер або кроссинговер). Цей оператор визначає передачу ознак предків потомкам.

Діє наступним чином:

з популяції вибираються по 2 особи, які будуть батьками;

точка розриву визначається(зазвичай випадковим чином);

потік визначається як сполучення частин першого та другого батьків.

Наступний генетичний драйвер призначений для підтримки різноманітності людей з популяцією. Це називається мутаційним фактором. Коли використовується цей оператор, кожен біт у хромосомі з певною ймовірністю інверсії.

Більше кажучи, має місце використання так званої оперативної інверсії,

яка входить в том, що хромосома деліться на дві частини, і тоді вони змінюються місцями.

У принципі для функціонування генетичного алгоритму достатньо цих двох генетичних операторів, але на практиці застосовуються ще і деякі додаткові оператори або модифікації даних операторів.

Для більшого розуміння генетичних алгоритмів розглянемо ще одне їх визначення.

Генетичні алгоритми є частиною еволюційних алгоритмів пошуку. Ідея генетичних алгоритмів базується на теорії еволюції Чарльза Дарвіна. Цей алгоритм імітує процес природного відбору, коли сильні особистості страждають і виробляють наступне покоління. При поглибленому навчанні використовуються генетичні алгоритми для оптимізації вимірювання нейронних мереж.

Природній відбір

Процес природного відбору починається з відбору сильних особистостей серед людей. Їхні нащадки успадковують здібності своїх предків, і вони є частиною майбутнього покоління. Якщо обидва батьки сильніші, потомство сильніше батьків.

Це інтерактивний процес, який завершується, коли відбираються дуже сильні особистості. Ця ідея працює з функціями пошуку. Просунута група людей вибрала колекцію вирішення проблем.

5 кроків генетичного алгоритму

Процес навчання генетичного алгоритму розділяється на 5 кроків:
Початкова популяція

Функція силової особистості

Відбір найбільш сильних рішень

Обмін характеристиками між двома особами

Мутація

Нова ітерація зі створенням початкової популяризації

Цей процес починається з групи особ, яку називають популяцією. Всі вони є рішеннями проблем. Риса - це сукупність параметрів (змінних), які називаються генами. Гени групуються, утворюючи хромосоми - вирішення проблем.

Генетичний алгоритм ембріонального комплексу представлений людині у вигляді бінарного виразу. Сукупність генів, кодованих хромосомою, називається хромосомою.

Функція сили

Функція силі визначає, ступінь сили однієї окремої особи. Сила визначається як здібність особи нав'язувати конкуренцію іншим особам за заданим вимірюванням. Функція присвоює кожній особі рівень сили. Імовірність того, що особа буде вибрана для виробництва наступної популяції, що визначається на рівні сильних особистостей.

Відбір

Основа ідеї відбору лежить в тому, щоб відібрати найсильніших осіб і передати їх генам наступному поколінню. Х пар осіб (батьків) відсортовується полягаючись на їхні сили.

Схрещування

Схрещування є важливою частиною генетичного алгоритму. Для кожної пари батьків. Точка в дволанцюжковій хромосомі, яка поділяє ген у людини, була обрана випадковим чином. Потім модифіковану особу називають нащадком.

Точка схрещування

Потомство створюється процесом обміну батьківськими генами у випадковому стані з часом. Після обміну генами у батьків потомства з’являється нове вивільнення.

Мутація

Деякий відсоток генів може бут маловірогідним. Для підтримки різноманіття поширення, відокремлено описується стадії змін нових осіб.

Завершення алгоритму

Алгоритм завершує роботу, коли популяція зійшлась, якщо не виробляється потомство, яке значно відрізняється від попереднього покоління. Коли алгоритм вирішився, на виході отримується набір оптимальних рішень заданої проблеми.

1.2.2 Штучна нейронна мережа

ШНМ (штучна нейронна мережа) - це математична модель активності звичайних нейронних мереж для живих організмів з мережевими нервовими клітинами. Згідно з біологічним аналізом, штучна мережа основних компонентів складається з нейронів, взаємопов’язаних та утворюючих шари, кількість яких може змінюватися залежно від нейронної мережі та її призначення (вирішення проблеми).

Найбільш поширеною функцією нейронної мережі є ідентифікація візуальних зображень. У наш час створені мережі, в яких машини можуть успішно ідентифікувати символи на папері та банківських картках, підписи на офіційних документах, ідентифікувати предмети тощо. Ці особливості можуть значно полегшити людську працю, а також підвищити надійність і точність різних робочих процесів через відсутність вантажопідйомності через людські причини.

Під час роботи із зображеннями застосування технології поглибленого вивчення є важливою сферою. Фотографії з різних куточків світу утворюють бібліотеку неструктурованих даних. Ця інформація використовується і використовується для виконання різноманітних завдань із використанням нейронних мереж, технічного навчання та художнього інтелекту:

внутрішнього, соціального, професійного та державного, особливо безпеки.

Теорія

Основою всієї архітектури для відео є аналіз першого етапу, який визначає зображення (об'єкт). Потім штучний інтелект використовує машинне навчання для ідентифікації та класифікації дій.

Щоб знати зображення, нейронна мережа має бути попередньо записаною. Це як нейронні зв’язки в мозку людини - ми володіємо деяким багажем знань, бачимо об'єкт, аналізуємо його та ідентифікуємо.

На практиці це означає, що для ідентифікації кордону, шарів, прихованих у нейронній мережі, зображення буде більш точним. Як це реалізувати?

Зображення розділене на невеликі шматочки, завантажені до декількох пікселів, кожен з яких буде вхідним нейроном. За допомогою синапсів сигнали передаються від одного слова до іншого. На цей момент тисячі нейронів з мільярдами параметрів порівнюють отриманий сигнал з уже обробленими даними.

Простіше кажучи, якщо ми просимо машину ідентифікувати фотографію кота, ми розрізаємо їх на невеликі шматочки і порівнюємо ці шари з перерахованими зображеннями мільйонів котів, які вивчала мережа.

Іноді збільшення кількості шарів стає простим вибором замість навчання. Потім - через цікаву архітектуру.

Нейронна мережа для розпізнавання зображень - це, найпопулярніша можливість застосування НМ(нейронних мереж). При цьому в залежності від особливостей вирішених завдань, вона працює на етапі, найбільш важливих серед яких розповідається нижче.

У якості розподілених зображень можуть виступати найрізноманітніші об'єкти, включаючи зображення, рукописний або друкований текст, звуки та багато іншого. При навчанні мережі пропонуються різні образи з метою того,

до якого саме типу можна віднести. У якості прикладу застосовується вектор знайомих ознак, а сукупність ознак на цих умовах має бути дозволено однозначно визначати, з яким класом утворює справу НМ (нейронна мережа).

Навчаючи, важливо не тільки визначити кількість та значення, необхідні для функцій для гарної точності в нових зображеннях, але також переглянути існування, хоча “не вписуватись” у навчальний метод не дуже добре. Після закінчення належного навчання НМ (нейронна мережа) повинна мати можливість вивчити зображення (того самого класу), щоб воно не зустрічалося в процесі навчання.

Важливо розуміти, що вихідні дані для нейронної мережі не повинні бути одноманітними або суперечливими, тим самим збільшуючи ймовірність належності об'єкта НМ (нейронної мережі) до одного класу.

1.3 **Постановка завдання**

- Створити систему-тренажер на основі гри “Flappy Bird”;
- Створити гру “Flappy Bird”;
- Підключення нейронної сітки;
- Створення генетичного алгоритму для головного персонажа гри за для проходження далі в сюжеті;
- Система тренажера має грати сама в себе.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАВДАННЯ

2.1 Вибір методів розробки інформаційного порталу

У сьогоденній галузі інноваційних технологій існує безліч технологій та алгоритмів розробки. Це відноситься всього продукту, а також окремих компонентів кінцевої системи. За допомогою новітніх програмних технологій як професійні команди, так і команди початківців можуть створити майже однаковий продукт. І група програмістів, і один ентузіаст. Однак головними відмінностями є час розробки, витрати на людські ресурси, витрати на розробку та характеристики архітектури товару. Тому вибір методу розробки є чи не найважливішим кроком у побудові всієї системи. Однією з основних проблем, з якою стикаються розробники при проектуванні системи, є розробка проекту за допомогою системи управління або безпосередньо мовою програмування. То яка різниця між цими двома методами розробки?

Мова програмування - це офіційна мова, яка використовується для написання комп’ютерних програм. Мова програмування визначає набір словникового запасу, синтаксису та семантичних правил, що визначають тип програми та дії, які комп'ютер виконує під своїм контролем.Мови програмування призначені для створення комп’ютерних програм.

Комп’ютерна програма - це набір правил, що дозволяє комп’ютеру виконувати певні комп’ютерні процеси та керувати різними об’єктами. Різниця між програмуванням та природною мовою полягає в тому, що він призначений для управління вашим комп’ютером. Багато мов програмування використовують спеціалізовані структури для визначення структури даних та управління комп'ютерними процесами.

Бібліотека програмного забезпечення (бібліотека) - це набір кроків або об’єктів, що використовуються для розробки програми. Це набір класів,

компонентів або модулів, які використовуються для виконання різних завдань Бібліотека - це сукупність перевірених кодів, створених кимось. Це готові рішення, які дозволяють програмісту підключатися до додатків, вводити код за допомогою спеціальних алгоритмів та використовувати в різних проектах.

Які переваги користування бібліотекою?

Вони економлять багато часу на кожному етапі. Використання бібліотеки є частиною функціонального способу написання програми. Програма має етапи. Сучасні програмісти використовують загальноприйняті процедури і за потреби використовують повні модулі або бібліотеки.

Сьогоднішня редакція явно складна та потужна для конкуруючих розробників програмного забезпечення. Що стосується темпів зростання, це дорого, трудомістко і неможливо. Один реалізував та опублікував багато алгоритмів, класів та функцій, що входять до коду. Це бібліотеки, якими може користуватися кожен. Це не тільки прискорює розробку та пришвидшує процес, але й зменшує помилки коду.

Фреймворк

Фреймворк або фреймворк - це платформа для розробки програмних додатків. Він забезпечує основу, де розробники програмного забезпечення можуть створювати програми для певної платформи. Наприклад, фреймворк може включати заздалегідь визначені класи та функції, які можуть бути використані для обробки вводу, управління апаратним забезпеченням та взаємодії із системним програмним забезпеченням. Це спрощує процес розробки, оскільки програмістам не потрібно виконувати одній ті самі дії щоразу, коли вони розробляють нову програму.

Фреймворк схожий на інтерфейс прикладного програмування (API), хоча технічно фреймворк включає API. Як випливає з назви, фреймворк служить основою для програмування, тоді як API забезпечує доступ до

елементів, що підтримуються фреймворком. Структура може також включати бібліотеки коду, компілятор та інші програми, що використовуються в процесі розробки програмного забезпечення.

2.2 Вибір засобів програмування системи

Під час розробки я буду використовуватися такі технології як HTML5 з використанням фреймворку Phase. Крім того, ми використовували бібліотеку Synaptic Neural Network для реалізації нейромережі, щоб не створювати її з нуля.

HTML| HTML5

HTML - це мова розмітки, яка використовується для розмітки веб-сторінок в Інтернеті. Браузер отримує документ HTML з веб-сервера або локального сховища і передає документ на мультимедійну веб-сторінку, семантично описує структуру веб-сторінки і спочатку включає сигнали про зовнішній вигляд сторінки.

Кожний HTML елемент - це будівельний блок HTML-сторінки. Використовуючи структуру HTML, блоки можуть бути вбудовані в середину відтвореної сторінки. HTML дає змогу користуватися інструменти для відтворення структурованих сторінок, що представляють собою структурну семантику тексту. Теги описують елемент HTML розмітки, які є записаними в кутові дужки. Мітки, такі як введення вмісту безпосередньо в сторінці. Інші теги забезпечують оточення та інформацію про текст документа і можуть мати інші теги як допоміжні елементи. Браузер не відображає HTML-теги, але використовує їх для пояснення змісту сторінки.

HTML можна включати в програми, написані мовою сценаріїв (наприклад, JavaScript), які впливають на поведінку та зміст веб-сторінок. Увімкнення CSS визначає тип і макет вмісту. З 1997 року Всесвітня павутина (W3C), яка підтримує стандарти HTML і CSS, прямо заохочує використовувати CSS замість відображення HTML. HTML використовує такі інструменти: структуровані документи, вказуючи структуру тексту: заголовки, абзаци, списки, таблиці, посилання тощо; Отримувати інформацію з Інтернету за допомогою посилань; Створення інтерактивних форм; Додайте

до тексту картинки, звуки, відео тощо.

HTML5 - це наступна версія HTML. До робочої групи HTML5 входять AOL, Apple Paul, Google, IBM, Microsoft, Mozilla, Nokia, Opera та сотні інших виробників. Існує певна плутанина щодо контролю версій, оскільки існують дві окремі групи розробників, WAHTWG та W3C. Коли WHATWG прийняла деталі HTML, вона відмовилася від принципу "контролю версій" на користь "постійного розвитку". Рішення полягає в прискоренні впровадження стандартів, тобто розробникам веб-браузерів не потрібно чекати офіційно завіреної копії специфікації (детально рекомендованої), тепер вони можуть виконати деякі частини специфікації. Тому, згідно з WHATWG, постійно розвивається лише одна особливість - HTML.

Дві команди працювали разом: WATWG писав функції в режимі реального часу, а W3C сприймав ці функції як "фотографію" та включав їх у конкретні версії своїх функцій. W3C працював дуже повільно, оскільки вимагав не лише веб-браузера, але й широкого кола користувачів.

28 жовтня 2014 р. W3C Alliance оголосив про надання набору функцій HTML5 у рекомендованому стандартному стані. Особливістю є те, що у цій формі специфікація HTML 5.0 була створена два роки тому, і подальша робота зосереджена на тестуванні та оцінці сумісності існуючих реалізацій.

На момент стандартизації HTML5 вже давно став справжнім стандартом і широко використовувався у веб-додатках. Лише фактична автентифікація стандартів закінчила просування HTML5 і підтвердила універсальність та точність його впровадження. Особливості HTML5 не обмежуються описом, але включають широкий спектр веб-технологій, інтегрованих для створення відкритих веб-платформ, програмного середовища для програмування платформ, яке може взаємодіяти з інструментами та пристроями підтримки відео та графіки та анімації.

Javascript

JavaScript (JS) - це динамічна та об'єктно-орієнтована мова програмування прототипів. Впровадження стандарту ECMAScript. Він в основному використовується для створення сценаріїв веб-сторінок, які дозволяють клієнтам (пристроям кінцевих користувачів) взаємодіяти з користувачем, керувати браузером, безперебійно взаємодіяти з сервером та змінювати дизайн та зовнішній вигляд веб-сторінки.

JavaScript класифікується як прототип (підмножина об’єктів), мова програмування сценаріїв, що передбачає динамічне введення тексту. На додаток до прототипування, JavaScript частково підтримує інші парадигми програмування (важливі та частково функціональні), а також деякі архітектурні особливості, зокрема: динамічне та повільне введення тексту, автоматичне управління пам’яттю, успадкування прототипів та об’єкти першого класу.

Фреймворк Phaser

Phaser - це 2D-ігровий фреймворк, який використовується для створення ігор HTML5 для настільних та мобільних пристроїв. Це безкоштовне програмне забезпечення, розроблене Photon Storm.

Phaser використовує як візуалізатор Canvas, так і WebGL і може автоматично переключатися між ними на основі підтримки браузера. Це дозволяє швидко здійснювати візуалізацію на настільному та мобільному пристроях. Він використовує бібліотеку Pixi.js для візуалізації.

Ігри можна розгорнути на iOS, Android та рідних робочих програмах за допомогою сторонніх інструментів, таких як Apache Cordova та phonegap.

Бібліотека Synaptic Neural Network

Synaptic - це бібліотека нейромереж Javascript для node.js та браузера, яка дозволяє навчати архітектури нейронних мереж першого та навіть другого порядку. Проект включає кілька вбудованих архітектур, таких як багатошарові персептрони, багатошарові мережі довгострокової пам'яті,

автомати з рідким станом та інструктор, здатний навчати справжність мереж.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 **Алгоритм машинного навчання**

За формулюванням Артура Самуеля 1959, машинне навчання - це спосіб змусити комп'ютери працювати без програмування в явній формі. У загальному випадку, це процес тонкої настройки навчання, поступово поліпшує вихідну випадкову систему.

Тобто метою тут є створення штучного інтелекту, який зможе знайти правильне рішення з поганої системи тонким налаштуванням параметрів моделі. Для цього в алгоритмі машинного навчання використовується безліч різних підходів.

Саме в цьому проекті, основний підхід до машинного навчання (machine learning algorithm, ML) заснований на нейроеволюції. У цій формі машинного навчання використовуються еволюційні алгоритми, такі як генетичний алгоритм (genetic algorithm, GA), для навчання штучних нейронних мереж (artificial neural networks, ANN).

Тобто в нашому випадку можна сказати, що ML = GA + ANN.

3.2 **Штучна нейронна мережа**

Штучна нейронна мережа - це підмножина алгоритму машинного навчання. За основу в ній взято структура і функції біологічних нейронних мереж. Ці мережі створені з безлічі нейронів, що передають сигнали один одному.

Тобто для створення штучного мозку нам потрібно симулювати нейрони і з'єднати їх, щоб вони сформували нейронну мережу.

Стандартна штучна нейронна мережа складається з шару вхідних даних, одного або декількох прихованих шарів і шару вихідних даних. У

кожному шарі є кілька нейронів. Нейрони вхідних і вихідних даних приєднані безпосередньо до зовнішнього середовища. Приховані нейрони з'єднуються між ними

У цьому проекті кожен об'єкт (птах) має власну нейронну мережу, яка використовується в якості II-мозку для проходження гри. Вона складається з наступних трьох шарів:

1) шар вхідних даних з двома нейронами представляє те, що бачить птах:

1.1) горизонтальне відстань до найближчого проміжку.

1.2) різниця висот з найближчим проміжком.

2) прихований шар з шістьма нейронами

3) шар вихідних даних з одним нейроном, що створює дію:

3.1)якщо вихідні дані> 0,5, то зробити ривок, в іншому випадку не робити нічого

На малюнку (Рисунок 3.1). нижче показана архітектура нейронної мережі для цього прикладу:

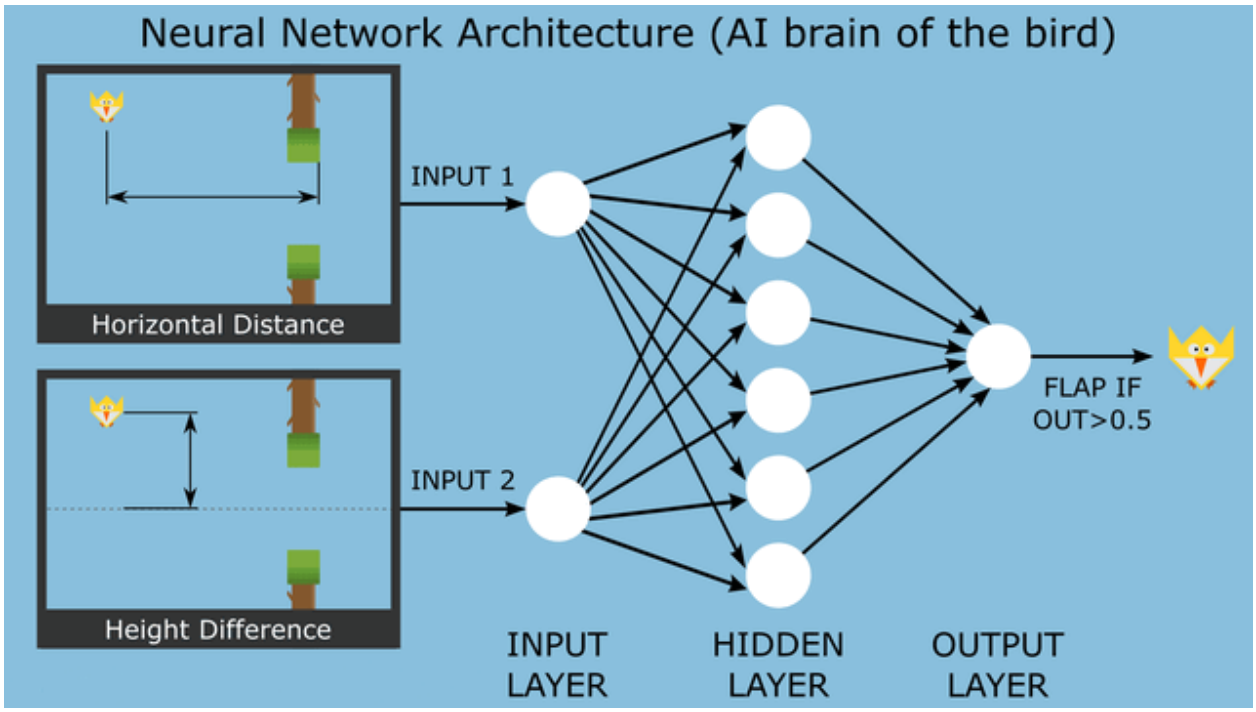


Рисунок 3.1 - Архітектура нейронної мережі

3.3 **Генетичний алгоритм**

Під час обговорення алгоритму машинного навчання, то говорили, що для навчання і вдосконалення нейронних мереж використовується генетичний алгоритм.

Генетичний алгоритм - це техніка оптимізації на основі пошуку, яка копіює природний відбір і генетику. У ній використовується таке ж поєднання відбору, кросинговеру і мутації для зміни вихідної випадкової популяції.

Ось основні етапи реалізації нашого генетичного алгоритму:

створюємо вихідну популяцію з 10 об'єктів (птахів) з випадковими нейронними мережами

даємо всім об'єктам грати одночасно з використанням їх власних нейронних мереж

у кожного об'єкта обчислюємо його функцію пристосованості для оцінки його якості (докладніше див. у розділі Функція пристосованості)

після смерті всіх об'єктів оцінюємо поточне покоління для створення нового за допомогою генетичних операторів (докладніше див. у розділі Стратегія заміни)

повертаємося до етапу 2

3.4 Функція пристосованості

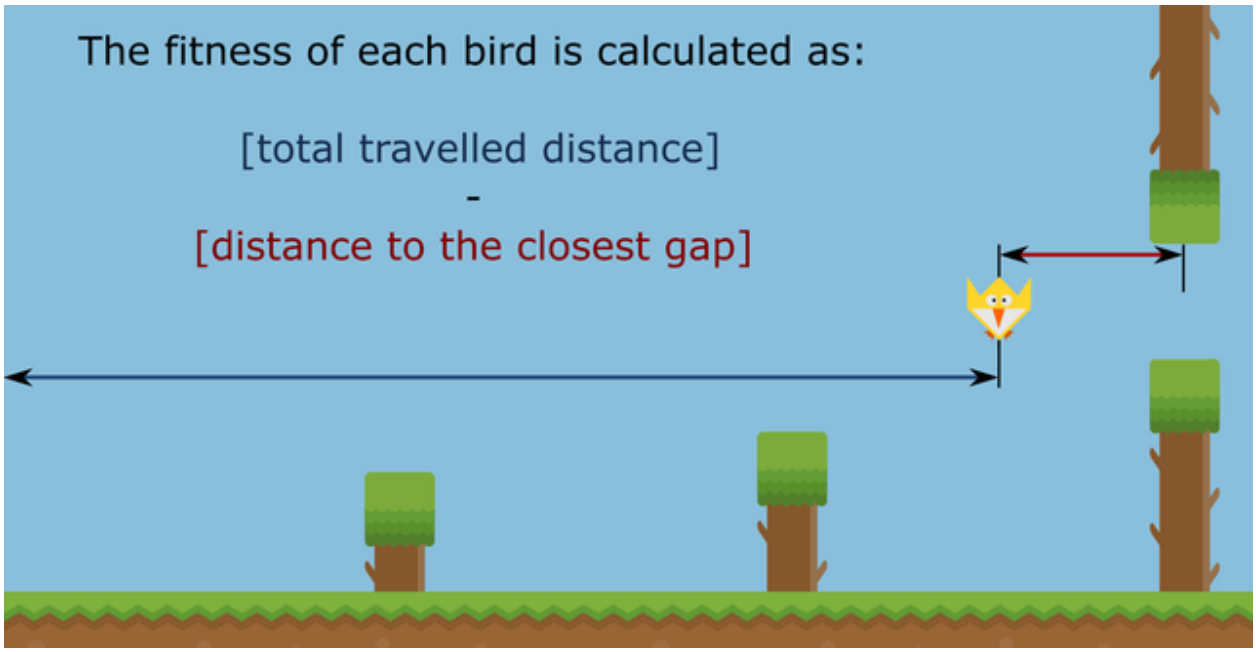
На додаток до генетичного алгоритму (етап 3), розглядаючи тут докладніше функцію пристосованості - що це таке і як її визначити.

Оскільки, щоб популяція еволюціонувала з найкращих об'єктів, нам необхідно визначити функцію пристосованості.

У загальному випадку, функція пристосованості - це метрика, що вимірює якість об'єкта. Якщо у нас буде метрика якості кожного птаха, з’являється можливість вибрати найбільш пристосовані об'єкти і використовувати їх для відтворення наступного покоління.

У цьому проекті ми винагороджуємо птицю в прямій залежності від зробленого відстані. Крім того, вона карається за поточним відстані до найближчого проміжку. Таким чином розрізняються птахи, які пролетіли однакова відстань.

Підіб'ємо підсумок: наша функція пристосованості - це різниця між загальною відстанню, пророблених птахом, і поточним відстанню до найближчого проміжку(Рисунок 3.2).



4 Рисунок 3.2 - Функція пристосованості

3.5 Стратегія заміни

На додаток до генетичного алгоритму (етап 4), ось етапи застосування природної еволюції до вмираючого поколінню. По суті, виживають кращі об'єкти, а їхні нащадки заміняють найгірші об'єкти наступним чином:

1)сортуємо об'єкти поточного покоління по їх рівню пристосованості

2)вибираємо чотири кращих об'єкта (переможців) і передаємо їх безпосередньо в наступне покоління

3)створюємо одного нащадка як результат кросинговеру між двома найкращими переможцями

4)створюємо трьох нащадків як результати кросинговеру двох випадкових переможців

5)створюємо двох нащадків як прямі копії двох випадкових переможців

6)застосовуємо до кожного нащадку випадкові мутації, щоб додати варіативності.

В остаточному вигляді гра- тренажер має вигляд (Рисунок 3.3) :

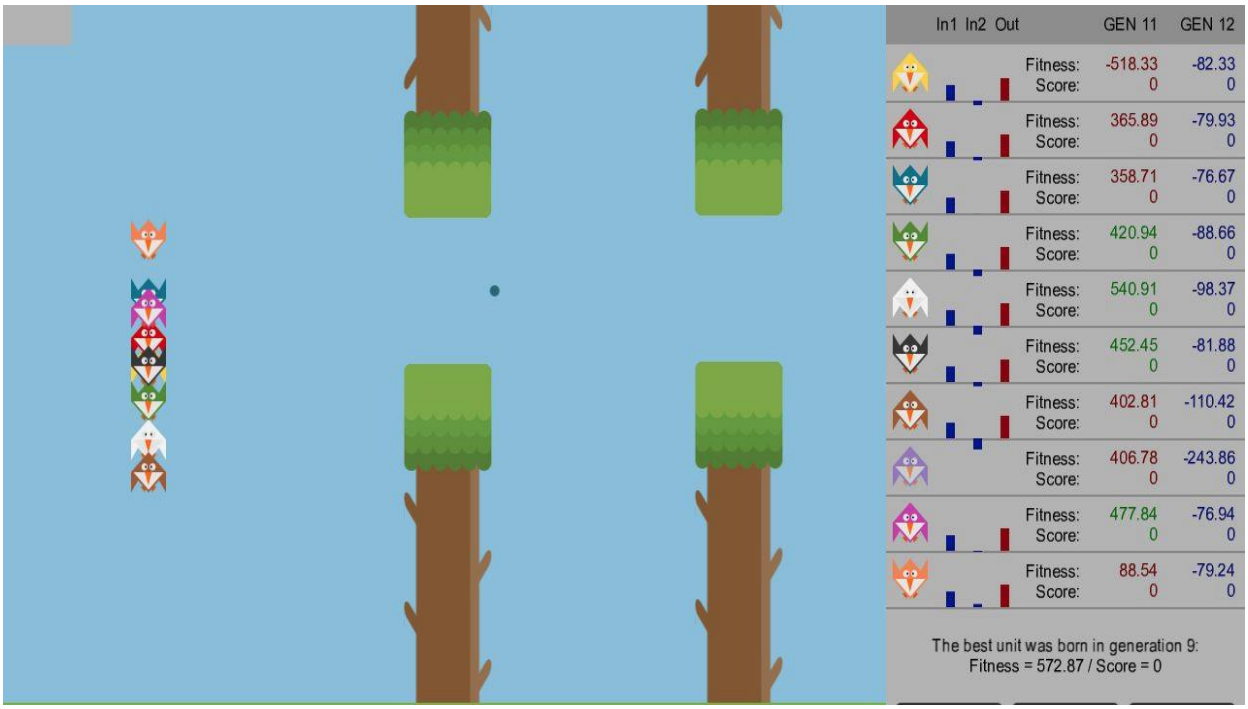


Рисунок 3.3 - Остаточний вигляд

3.6 Тестування

1) Тестування розпочинається з запуску самої системи тренажера (Рисунок 3.4).

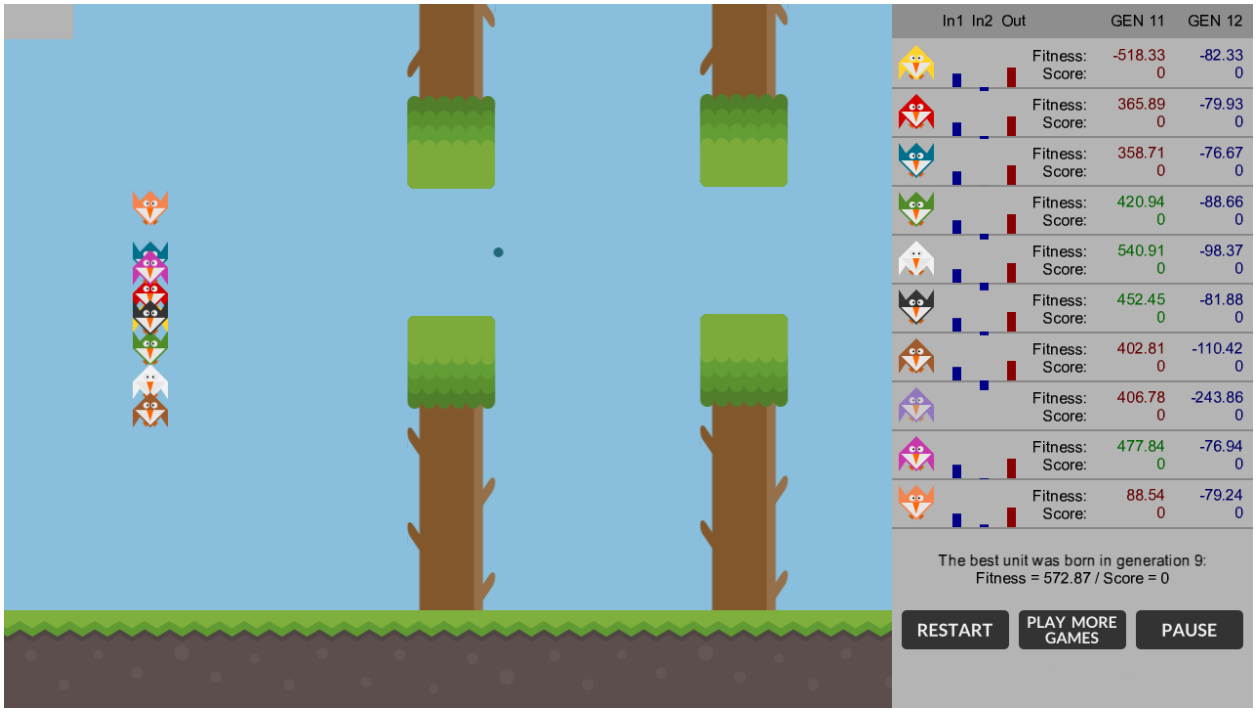


Рисунок 3.4 — Запуск системи-тренажера

2) На рисунку 3.4 продемонстровано роботу випускання першого покоління пташок, якщо все покоління пташок програє випускається наступне на основі зібраних даних першого покоління.

3) Кнопка “Restart” перезапускає гру і покоління пташок запускаються заново (Рисунок 3.5).

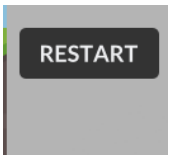


Рисунок 3.5 — Кнопка “Restart”

4) Кнопка “Pause” ставить систему-тренажер на паузу(Рисунок 3.6).

Рисунок 3.6 - Кнопка “Pause”

5) На Рисунку 3.7 зображена бокова панель,на якій виводиться для користувача інформація про пташок з кожного покоління та рекорд найкращого птаха і з якого він був покоління.

Рисунок 3.7 — Бокова панель

ВИСНОВОК

В даній дипломній роботі було спроектовано та реалізовано систему-тренажер для “Теорії ігор” з використанням генетичних алгоритмів машинного проєктування. Проведена робота з аналізу методів проектування, програмування, вибору інструментів для реалізації проекту.

Для розробки елементів сайту було використано фреймворк Phaser, використовується для створення ігор HTML5 для настільних та мобільних пристроїв. Використовується Бібліотека Synaptic Neural Network для ствоврення нейронної мережі що б не створювати її з нуля.

Як результат, було розроблено систему, яка відповідає всім вимогам замовника. Основний функціонал було перевірено дев-тестуванням та відправлено на оцінку якості замовнику.

СПИСОК ЛІТЕРАТУРИ

1) Bayesian Reasoning and Machine Learning David Barber c ,2010 c. 100-128
2) Шпетний, І.О. Інформатика [Електронний ресурс] / І.О. Шпетний, С.І. Проценко, К.В. Тищенко. - Електронне вид. каф. Електроніки, загальної та прикладної фізики. - Суми: СумДУ, 2018. - 187 с.
3) Булашенко, А.В. Інформатика і комп’ютерна техніка: конспект лекцій для студ. спец. 6.030601 "Менеджмент і адміністрування" заочної форми навчання / А.В. Булашенко. - Суми: СумДУ, 2012. - 232 с.
4) Демиденко, М.Г. Крос-платформні мови програмування: конспект лекцій / М.Г. Демиденко, О.В. Федченко. - Суми: СумДУ, 2010. - 88 с.
5) Шендрик, В.В. Web-програмування: конспект лекцій для студ. спец. (7) 8.05010102 "Інформаційні технології проектування" денної та заочної форм навчання / В.В. Шендрик, О.В. Бондар, Ю.В. Парфененко. - Електронне видання каф. Комп'ютерних наук, сек. ІТФ. - Суми: СумДУ, 2013. - 124 с.
6) Прикладне машинне навчання з допомогою Scikit-Learn, Keras і TensorFlow. Орельєн Жерон, 2015 ст. 25-40.
7) Розпізнавання образів і машинне навчання. том 1. Кристофер М. Бишоп
8) Adaptation in Natural and Artificial Systems,An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence,John H. Holland 2011 c. 140-156.
9) Офіційний сайт документації з Phaser.io [Електронний ресурс] - https://phaser.io/
10) Інформаційний портал Synaptic Neural Network [Електронний ресурс] - https://caza.la/synaptic/#/

ДОДАТКИ

Додаток А – головна сторінка тренажера

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8" />
                <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1" />

                <meta name="application-name" content="Flappy
Bird" />
                <meta name="author" content="Viktor Malezhyk" />
                <meta name="description" content="An HTML5
simple simulation of an artificial intelligence for the Flappy
Bird game using Neural Networks and a Genetic Algorithm which
can teach a little bird how to flap optimally in order to fly
safely through single gaps placed between vertical barriers as
long as possible." />
                <meta name="keywords" content="html5, game,
simulation, artificial, intelligence, flappy, bird, neural,
network, genetic, algorithm" />
                <meta name="copyright" content="Viktor Malezhyk"
/>

                <title>Flappy Bird</title>
                <style>body {margin: 0; background: #333;
overflow: hidden}</style>
                <script src = "phaser.min.js"></script>
                <script src = "synaptic.min.js"></script>
                <script src = "gameplay.js"></script>
                <script src = "genetic.js"></script>
        </head>
        <body>
        </body>
</html>
```

Додаток Б – модуль генетичного алгоритму

```
var GeneticAlgorithm = function(max_units, top_units){
	this.max_units = max_units; // max number of units in
population
	this.top_units = top_units; // number of top units
(winners) used for evolving population

	if (this.max_units < this.top_units) this.top_units =
this.max_units;

	this.Population = []; // array of all units in current
population

	this.SCALE_FACTOR = 200; // the factor used to scale
normalized input values
}

GeneticAlgorithm.prototype = {
	// resets genetic algorithm parameters
	reset : function(){
		this.iteration = 1;	// current iteration
number (it is equal to the current population number)
		this.mutateRate = 1; // initial mutation rate

		this.best_population = 0; // the population
number of the best unit
		this.best_fitness = 0;  // the fitness of the
best unit
		this.best_score = 0;	// the score of the best
unit ever
	},

	// creates a new population
	createPopulation : function(){
		// clear any existing population
		this.Population.splice(0,
this.Population.length);

		for (var i=0; i<this.max_units; i++){
			// create a new unit by generating a
random Synaptic neural network
			// with 2 neurons in the input layer, 6
neurons in the hidden layer and 1 neuron in the output layer
			var newUnit = new
synaptic.Architect.Perceptron(2, 6, 1);
```

```
                        // set additional parameters for the new
unit
                        newUnit.index = i;
                        newUnit.fitness = 0;
                        newUnit.score = 0;
                        newUnit.isWinner = false;

                        // add the new unit to the population
                        this.Population.push(newUnit);
                }
        },

        // activates the neural network of an unit from the
population
        // to calculate an output action according to the inputs
        activateBrain : function(bird, target){
                // input 1: the horizontal distance between the
bird and the target
                var targetDeltaX = this.normalize(target.x, 700)
* this.SCALE_FACTOR;

                // input 2: the height difference between the
bird and the target
                var targetDeltaY = this.normalize(bird.y -
target.y, 800) * this.SCALE_FACTOR;

                // create an array of all inputs
                var inputs = [targetDeltaX, targetDeltaY];

                // calculate outputs by activating synaptic
neural network of this bird
                var outputs =
this.Population[bird.index].activate(inputs);

                // perform flap if output is greater than 0.5
                if (outputs[0] > 0.5) bird.flap();
        },

        // evolves the population by performing selection,
crossover and mutations on the units
        evolvePopulation : function(){
                // select the top units of the current
population to get an array of winners
                // (they will be copied to the next population)
                var Winners = this.selection();

                if (this.mutateRate == 1 && Winners[0].fitness <
0){
```

```
                        // If the best unit from the initial
population has a negative fitness
                        // then it means there is no any bird
which reached the first barrier!
                        // Playing as the God, we can destroy
this bad population and try with another one.
                        this.createPopulation();
                } else {
                        this.mutateRate = 0.2; // else set the
mutatation rate to the real value
                }

                // fill the rest of the next population with new
units using crossover and mutation
                for (var i=this.top_units; i<this.max_units;
i++){
                        var parentA, parentB, offspring;

                        if (i == this.top_units){
                                // offspring is made by a
crossover of two best winners
                                parentA = Winners[0].toJSON();
                                parentB = Winners[1].toJSON();
                                offspring =
this.crossOver(parentA, parentB);

                        } else if (i < this.max_units-2){
                                // offspring is made by a
crossover of two random winners
                                parentA =
this.getRandomUnit(Winners).toJSON();
                                parentB =
this.getRandomUnit(Winners).toJSON();
                                offspring =
this.crossOver(parentA, parentB);

                        } else {
                                // offspring is a random winner
                                offspring =
this.getRandomUnit(Winners).toJSON();
                        }

                        // mutate the offspring
                        offspring = this.mutation(offspring);

                        // create a new unit using the neural
network from the offspring
                        var newUnit =
synaptic.Network.fromJSON(offspring);
```

```
                        newUnit.index =
this.Population[i].index;
                        newUnit.fitness = 0;
                        newUnit.score = 0;
                        newUnit.isWinner = false;

                        // update population by changing the old
unit with the new one
                        this.Population[i] = newUnit;
                }

                // if the top winner has the best fitness in the
history, store its achievement!
                if (Winners[0].fitness > this.best_fitness){
                        this.best_population = this.iteration;
                        this.best_fitness = Winners[0].fitness;
                        this.best_score = Winners[0].score;
                }

                // sort the units of the new population in
ascending order by their index
                this.Population.sort(function(unitA, unitB){
                        return unitA.index - unitB.index;
                });
        },

        // selects the best units from the current population
        selection : function(){
                // sort the units of the current population
in descending order by their fitness
                var sortedPopulation = this.Population.sort(
                        function(unitA, unitB){
                                return unitB.fitness -
unitA.fitness;
                        }
                );

                // mark the top units as the winners!
                for (var i=0; i<this.top_units; i++)
this.Population[i].isWinner = true;

                // return an array of the top units from the
current population
                return sortedPopulation.slice(0,
this.top_units);
        },

        // performs a single point crossover between two parents
        crossOver : function(parentA, parentB) {
```

```
                // get a cross over cutting point
                var cutPoint = this.random(0,
parentA.neurons.length-1);

                // swap 'bias' information between both parents:
                // 1. left side to the crossover point is copied
from one parent
                // 2. right side after the crossover point is
copied from the second parent
                for (var i = cutPoint; i <
parentA.neurons.length; i++){
                        var biasFromParentA =
parentA.neurons[i]['bias'];
                        parentA.neurons[i]['bias'] =
parentB.neurons[i]['bias'];
                        parentB.neurons[i]['bias'] =
biasFromParentA;
                }

                return this.random(0, 1) == 1 ? parentA :
parentB;
        },

        // performs random mutations on the offspring
        mutation : function (offspring){
                // mutate some 'bias' information of the
offspring neurons
                for (var i = 0; i < offspring.neurons.length;
i++){
                        offspring.neurons[i]['bias'] =
this.mutate(offspring.neurons[i]['bias']);
                }

                // mutate some 'weights' information of the
offspring connections
                for (var i = 0; i <
offspring.connections.length; i++){
                        offspring.connections[i]['weight'] =
this.mutate(offspring.connections[i]['weight']);
                }

                return offspring;
        },

        // mutates a gene
        mutate : function (gene){
                if (Math.random() < this.mutateRate) {
                        var mutateFactor = 1 + ((Math.random() -
0.5) * 3 + (Math.random() - 0.5));
```

```
                        gene *= mutateFactor;
                }

                return gene;
        },

        random : function(min, max){
                return Math.floor(Math.random()*(max-min+1) +
min);
        },

        getRandomUnit : function(array){
                return array[this.random(0, array.length-1)];
        },

        normalize : function(value, max){
                // clamp the value between its min/max limits
                if (value < -max) value = -max;
                else if (value > max) value = max;

                // normalize the clamped value
                return (value/max);
        }
}
```