МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет компьютерных наук

Применение генетического алгоритма для обучения нейронной сети

Курсовая работа

09.03.04 Информационная безопасность

Кафедра программирования и информационных технологий

Допущено к защите в ГЭК \_\_.\_\_.2023

Зав. КафедройС.Д. Махортов, д. ф.-м. н., профессор

ОбучающийсяР.Т. Халилов, *3 курс, д/о*

Руководитель А.Л. Гавшин, ст. преподаватель

Воронеж 2023

Содержание

[Содержание 2](#_Toc24982)

[Введение 3](#_Toc6819)

[1 Постановка задачи 4](#_Toc7668)

[1.1 Постановка задачи 4](#_Toc16072)

[1.2 Средства реализации 4](#_Toc31074)

[2 Анализ предметной области 6](#_Toc23042)

[2.1 Терминология (глоссарий) предметной области 6](#_Toc4042)

[2.2 Общее понятие 6](#_Toc3243)

[2.3 Составные части нейронной сети 7](#_Toc3020)

[2.3.1 Искусственный нейрон 7](#_Toc31693)

[2.3.2 Функции суммирования и активации 8](#_Toc14222)

[2.4 Архитектура нейронной сети 9](#_Toc9520)

[2.4.1 Перцептрон 10](#_Toc27912)

[2.4.2 Сеть прямого распространения 10](#_Toc18679)

[2.4.3 Автоэнкодеры 10](#_Toc10678)

[2.5 Методы обучения нейронных сетей 12](#_Toc15624)

[2.5.1 Классификация алгоритмов 12](#_Toc26837)

[2.5.2 Метод обратного распространения 13](#_Toc24786)

[2.5.3 Генетический алгоритм 15](#_Toc16539)

[2.5.4 Стохастический алгоритм 17](#_Toc23602)

[2.5.5 Гибридный метод 18](#_Toc25091)

[3 Реализация 19](#_Toc27266)

[3.1 Описание мира 19](#_Toc22563)

[3.2 Модель нейросети 19](#_Toc15125)

[3.3 Реализация генетического алгоритма 21](#_Toc2866)

[3.3.1 Приспособленность 21](#_Toc31410)

[3.3.2 Селекция и скрещивание 22](#_Toc27805)

[3.3.3 Селекция 23](#_Toc30156)

[3.3.4 Мутация 23](#_Toc7388)

[3.3.5 Выход из локального минимума 24](#_Toc389)

[3.4 Выводы о модели 25](#_Toc8428)

[3.5 Реализация интерфейса 28](#_Toc2497)

[Заключение. 29](#_Toc11414)

[Список использованных источников 30](#_Toc21711)

Введение

В последние десятилетия нейронные сети привлекли широкое внимание в области искусственного интеллекта и машинного обучения. Они представляют собой мощный инструмент, способный обрабатывать сложные данные, извлекать полезные закономерности и принимать решения на основе этих данных. Нейронные сети успешно применяются во многих областях, таких как компьютерное зрение, распознавание речи, естественный язык, рекомендательные системы и другие.

Результатом практической части работы будет модель нейронной сети, обученная генетическим алгоритмом.

Исследование нейронных сетей имеет большой потенциал для прогресса и развития самых различных областей. Вместе с развитием вычислительных ресурсов и новых технологий, нейронные сети становятся все более мощными и эффективными. Понимание их возможностей и ограничений становится ключевым фактором для успешного применения в различных сферах жизни и промышленности.

1 Постановка задачи

1.1 Постановка задачи

Целью теоретической части работы является рассмотрение следующих тем:

* Общее понятие нейронных сетей;
* Возможные архитектуры;
* Алгоритмы обучения.

Целью практической части работы является создание приложения симуляции генетического алгоритма, перед которым ставятся следующие требования:

* Наличие графического интерфейса;
* Сбор статистики.

1.2 Средства реализации

В качестве средств реализации был использован язык программирования dart.

Среди преимуществ языка dart можно выделить:

* Объектноориентированность;
* Высокая производительность;
* Простота и читаемость кода;
* Большая экосистема инструментов;
* Поддержка ассинхронного программирования.

Для отрисовки графического интерфейса выбран фреймворк flutter.

Среди преимуществ фреймворка можно выделить:

* Быстрая разработка и простая отладка;
* Высокая производительность, помогающая при рендере картинки в реальном времени.

1. Анализ предметной области
   1. Терминология (глоссарий) предметной области

*Нейронная сеть* — [математическая модель](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C" \o "Математическая модель), а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей [нервных клеток](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD" \o "Нейрон) живого организма

*Нейрон* — узел [искусственной нейронной сети](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C" \o "Искусственная нейронная сеть), являющийся упрощённой моделью [естественного нейрона](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD" \o "). Реализация нейрона - контейнер, хранящий число.

*Вес связи* — число, определяющее как нейрон влияет на следующий нейрон.

*Функция суммирования* — функция, использующая веса и значения нейронов предыдущего слоя. Является входный параметром для функции активации

*Функция активации* — функция, задающая значение нейрона в заданных моделью пределах. Использует функцию суммирования в качестве входного параметра.

*Обучение нейронной сети* — процесс подбора весов и пороговых значений сети для оптимального решения задачи.

*Приспособленность* — показатель, высчитываемый для нейронной сети по каким-либо показателям, которые отслеживаются во время её работы

*Скрещивание* — процесс создания новой нейронной сети путём перемешивания генов 2 других нейронных сетей (родителей).

*Мутация* — процесс случайного изменения весов и/или пороговых значений нейронных сетей

* 1. Общее понятие

Нейронные сети — это алгоритмы машинного обучения, которые состоят из множества связанных между собой нейронов. Эти нейроны работают вместе, чтобы обработать данные и выполнить задачи, такие как классификация, предсказание и распознавание образов.

Каждый нейрон в нейронной сети получает входные данные, которые он обрабатывает и передает на выход. Входные данные проходят через веса, которые определяют, как сильно входное значение влияет на результат. Веса в начале процесса случайно задаются, а затем изменяются во время обучения сети, чтобы улучшить ее результаты.

В нейронных сетях используются различные функции активации, которые позволяют нейронам генерировать нелинейные выходы. Такие функции позволяют сети решать более сложные задачи, чем линейная классификация.

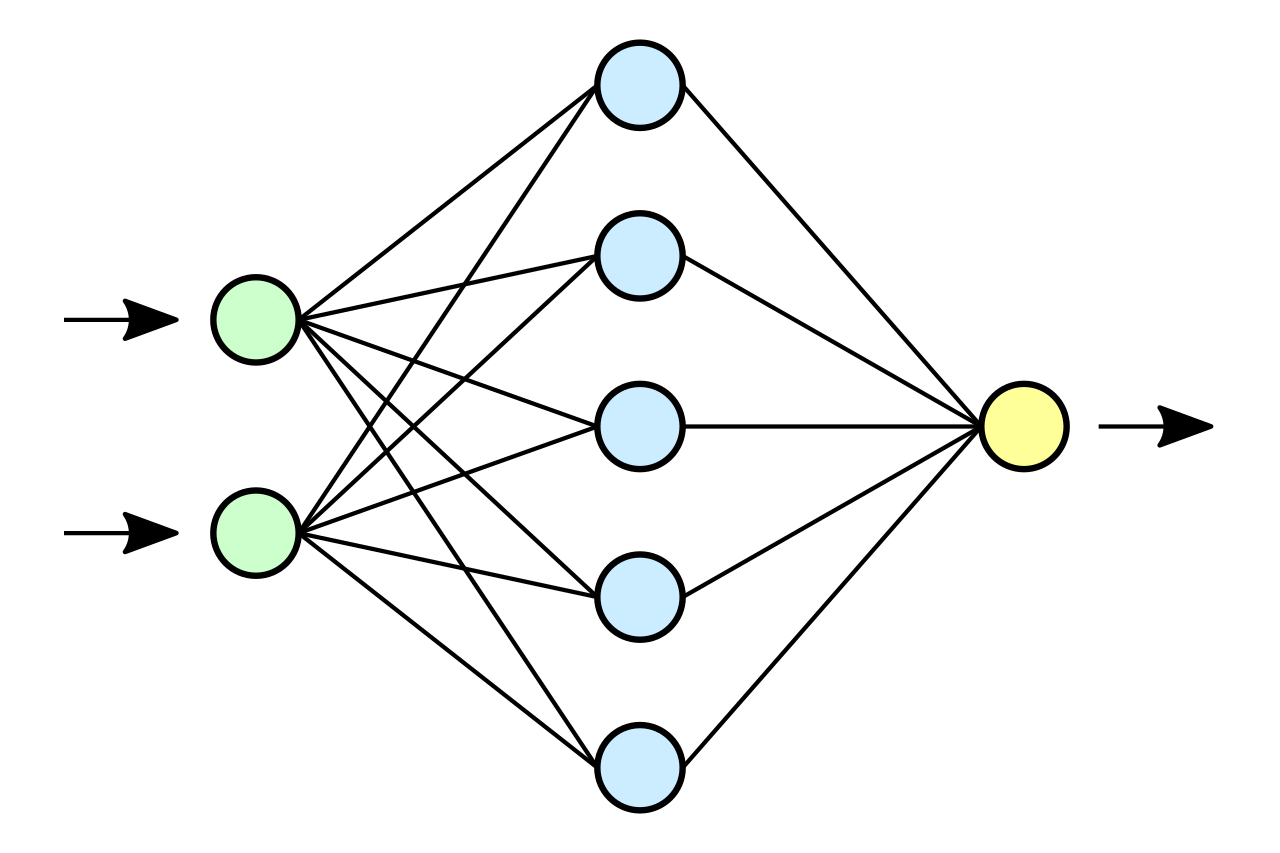
Процесс обучения нейронной сети включает в себя многократное предоставление ей входных данных и корректировку весов в соответствии с ожидаемым выходом. Этот процесс повторяется до тех пор, пока сеть не достигнет необходимой точности в решении задачи.

* 1. Составные части нейронной сети
     1. Искусственный нейрон

Искусственный нейрон, также известный как нейрон МакКаллока-Питтса или нейрон Розенблатта, является базовым строительным блоком искусственной нейронной сети. Он моделирует функциональность биологического нейрона и используется для обработки информации в нейронных сетях.

Искусственный нейрон принимает входные сигналы, взвешивает их и применяет функцию активации для вычисления выходного значения.

С точки зрения реализации нейрон представляет собой вещественное число, обычно от 0 до 1 или от -1 до 1, но в зависимости от задачи пороговые значения могут меняться. Рассмотрим схему простой нейронной сети (Рисунок 1).



1. Схема простой нейронной сети

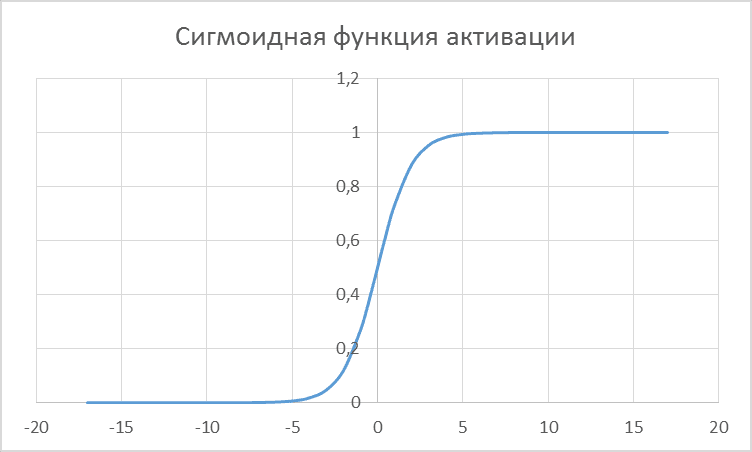
Зелёным цветом обозначены входные нейроны. Их изначальное значение зависит от различных параметров входных данных. Синим цветом обозначены скрытые нейроны. Их значения используются для промежуточных вычислений. Жёлтым обозначены выходные нейроны - решение поставленной перед сетью задачи.

* + 1. Функции суммирования и активации

Каждая связь между нейронами имеет свой вес. Чем больше вес определённой связи по модулю, тем больше она влияет на значение следующего нейрона. Значение следующего нейрона определяется суммой входящих в нейрон связей: вес каждой связи умножается на значение нейрона, из которого она выходит, и прибавляется к общему результату. Таким образом значение нейрона, обозначим его S, представляет собой:

Где - вес i-того входящего нейрона, а - его значение. Значение нейрона должно лежать в границах между 0 и 1, поэтому результат сумматорной функции передается через функцию активации, которая определяет выходное значение нейрона в зависимости от полученного входа. Функция активации может включать в себя нелинейные операции и определяет, должен ли нейрон активироваться или оставаться неактивным. Чтобы свести любое число к промежутку от 0 до 1, полезно использовать сигмоидную функцию:

График сигмоидной функции представлен ниже на рисунке 2:



1. Сигмоидная функция

Теперь активация нейрона, происходит, когда сумма превосходит ноль и чем сильнее она его превосходит, тем ближе значение будет к единице. Однако данная формула не учитывает, что возможно для оптимальной работы алгоритма необходимо, чтобы сумма связей превышала определённое число, не обязательно ноль. Для этого в формуле появляется свободный член - пороговое значение b. Таким образом получаем финальную формулу для вычисления значения нейрона:

* 1. Архитектура нейронной сети

Существует множество моделей нейронных сетей, каждая из которых предназначена для определённого типа задач. Ниже приведены основные модели нейронных сетей:

* + 1. Перцептрон

Простейшая модель нейронной сети. На рисунке жёлтым обозначены входные нейроны, оранжевым - выходные. Простейший перцептрон не имеет скрытый слоёв. Такая модель подходит для решения простейших задач по классификации объектов по их входным параметрам.



1. Простейший перцептрон
   * 1. Сеть прямого распространения

Сеть прямого распространения (Feed forward neural network, FF, FFNN) - усложнённая модель перцептрона, в которой появляется скрытый слой нейронов. На рисунке 4 нейроны этого слоя показаны зелёным цветом.

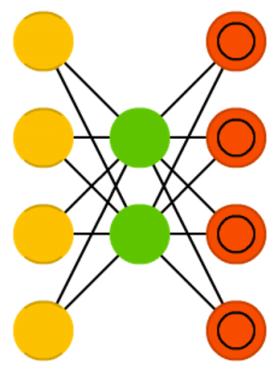


1. Сеть прямого распространения

Нейроны одного слоя между собой не связаны, при этом каждый нейрон этого слоя связан с каждым нейроном соседнего слоя.

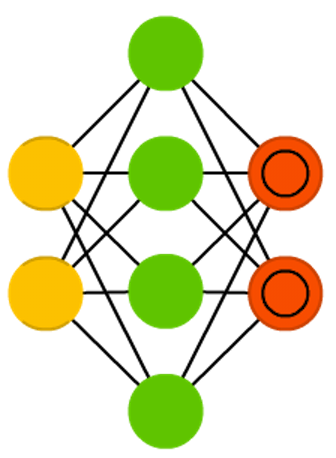
* + 1. Автоэнкодеры

Автоэнкодеры (Autoencoders, AE) - модификация сетей прямого распространения. Основная идея автоэнкодеров — автоматическое кодирование (как при сжатии, а не при шифровании) информации. Самый маленьких слой почти всегда средний, в нем информация максимально сжата. Все, что расположено до середины — кодирующая часть, выше середины — декодирующая.



1. Схема автоэнкодера

Если мы попытаемся расширить средние слои такой сети, то мы получим разреженные автоэнкодеры.



1. Схема разреженного автоэнкодера

Разреженный автоэнкодер (Sparse autoencoder, SAE) - противоположность предыдущей сети. Информация не сжимается, а наоборот расширяется и занимает больше места. Сети этого типа могут применяться для извлечения множества маленьких деталей из набора данных

Архитектура вариационных автоэнкодеров (VAE) такая же, как и у обычных, но обучают их другому — приблизительному вероятностному распределению входных образцов. Это в какой-то степени возвращение к истокам, так как VAE немножко ближе к машинам Больцмана. Тем не менее, они опираются на Байесовскую математику касательно вероятностных суждений и независимости, которые интуитивно понятны, но требуют сложных вычислений. Базовый принцип можно сформулировать так: принимать в расчет степень влияния одного события на другое. Если в одном месте происходит определенное событие, а другое событие случается где-то еще, то эти события вовсе не обязательно связаны. Если они не связаны, то распространение ошибки должно это учитывать.

* 1. Методы обучения нейронных сетей

Нейронные сети не программируются в обычном понимании этого слова, они обучаются. Процесс обучения по сути представляет собой алгоритм поиска нужных весов и пороговых значений для каждой связи. Существует множество алгоритмов для обучения нейронных сетей. Рассмотрим самые часто используемые.

* + 1. Классификация алгоритмов

Выделяют 2 главные категории алгоритмов обучения нейронных сетей:

Первая категория - обучение с учителем (Supervised Learning): обучение с учителем является самым простым и распространенным подходом и применяется для решения задач классификации и регрессии. В этом случае, нейронная сеть обучается на основе размеченных обучающих данных, где каждый пример обучающей выборки содержит входные данные и соответствующий эталонный (желаемый) выход.

Самый часто используемый здесь метод обучения - метод обратного распространения ошибки.

Вторая категория - обучение без учителя (Unsupervised Learning): Обучение без учителя применяется в случаях, когда у нас нет размеченных данных или целевой переменной, и мы стремимся найти внутренние закономерности или структуры в данных. Вместо эталонных выходных данных, модель обучается на основе набора неразмеченных данных и стремится кластеризовать данные, выделять скрытые признаки или реконструировать исходные данные.

Обучение без учителя позволяет находить скрытые структуры и закономерности в данных, выявлять новые знания и проводить предобработку данных для последующего обучения с учителем. Этот подход часто используется в задачах кластеризации, сжатия данных, поиске ассоциативных правил и других областях, где целью является понимание данных без явного учителя или целевой переменной.

Общий процесс обучения любой нейронной сети включает в себя следующие шаги:

* Подготовка данных: Набор неразмеченных данных подготавливается для обучения;
* Определение архитектуры сети: Определяется структура нейронной сети, включая количество слоев и функции активации;
* Инициализация весов: Веса сети инициализируются случайными значениями;
* Прямое распространение (Forward Propagation): Входные данные передаются через сеть, вычисляются выходы каждого нейрона и получается преобразованное представление данных;
* Корректировка весов;
* Повторение процесса: Процесс прямого распространения и корректировки весов повторяется до достижения определенного критерия остановки, такого как сходимость кластеров или достижение достаточно хорошего качества реконструкции данных.

Большинство алгоритмов придерживаются этой схемы обучения с исключением шага корректировки весов. Корректировка весов - главное отличие каждого алгоритма от остальных

* + 1. Метод обратного распространения

Метод обратного распространения является одним из наиболее широко используемых методов обучения нейронных сетей с учителем. Он основан на градиентном спуске и позволяет настраивать веса нейронной сети, чтобы минимизировать ошибку между прогнозируемым выходом сети и эталонными выходными данными.

Корректировка весов для данного алгоритма имеет следующие шаги:

* Расчет ошибки: Сравнивается прогнозируемый выход с эталонными выходными данными, и вычисляется ошибка сети, обычно с использованием функции потерь, такой как среднеквадратичная ошибка (MSE) или перекрестная энтропия;
* Обратное распространение ошибки: Ошибка распространяется обратно через сеть с помощью вычисления градиентов функции потерь по весам сети. Это позволяет определить, какие веса сети вносят наибольший вклад в ошибку и как нужно их корректировать.

Алгоритм обратного распространения используется в ряде следующих задач:

* Классификация: требуется отнести входные данные к определенным категориям или классам. Примеры включают распознавание образов, классификацию текстовых документов, определение эмоциональной окраски текста и многие другие;
* Регрессия: необходимо предсказать числовое значение или непрерывную переменную на основе входных данных. Примеры включают прогнозирование цен на недвижимость, предсказание временных рядов, оценку рисков и другие задачи, где требуется предсказание числовых значений;
* Обработка естественного языка: машинный перевод, распознавание речи, генерация текста. Нейронные сети, обученные с помощью обратного распространения, способны извлекать смысл из текстовых данных и выполнять сложные задачи обработки языка;
* Рекомендательные системы: такие системы предлагают пользователям персонализированные рекомендации. Нейронные сети могут анализировать предпочтения пользователя на основе исторических данных и предлагать подходящие товары, фильмы, музыку или другие рекомендации.
  + 1. Генетический алгоритм

Генетический алгоритм является эволюционным методом оптимизации, вдохновленным процессами естественного отбора и генетики. Один из самых часто используемых алгоритмов обучения без учителя. Он может использоваться для обучения нейронных сетей путем эволюции и оптимизации их параметров.

Корректировка весов происходит не столько изменением текущих весов нейронной сети, сколько созданием новых сущностей сети с новыми весами. Общий генетический алгоритм можно записать следующим образом:

* Инициализация популяции: Создается начальная популяция нейронных сетей с случайно установленными весами;
* Оценка приспособленности: Каждая сеть из популяции оценивается на основе их производительности в решении задачи. Приспособленность (Fitness) может измеряться с помощью функции оценки, которая определяет, насколько хорошо сеть выполняет поставленную задачу. Для каждой конкретной задачи функция приспобленности будет уникальной;
* Селекция: Сети с наилучшей приспособленностью имеют больше шансов выжить и передать свои гены следующему поколению. Процесс селекции может включать различные методы, такие как рулеточная селекция или турнирная селекция;
* Скрещивание: Выбранные сети скрещиваются, чтобы создать новые потомки. Это происходит путем комбинирования генетического материала родительских сетей, например, путем смешивания их весов или структуры;
* Мутация: В некоторых случаях в новом потомстве могут происходить случайные изменения или мутации. Это позволяет вносить разнообразие в популяцию и исследовать новые решения;
* Обновление популяции: Новое поколение сетей заменяет предыдущую популяцию, и процесс оценки, селекции, скрещивания и мутации повторяется;
* Повторение процесса.

Генетический алгоритм активно применяется для решения следующих задач:

* Задачи поиска оптимального решения: это может включать поиск оптимальных значений параметров для сложных математических моделей, настройку гиперпараметров алгоритмов машинного обучения или оптимизацию функций с несколькими переменными;
* Проблемы комбинаторной оптимизации: генетический алгоритм используется, если требуется найти лучшую комбинацию из множества вариантов. Примеры включают задачи о рюкзаке, задачи планирования и покрытия, раскраску графов и маршрутизацию;
* Кластеризация и кластерный анализ: ГА может применяться для кластеризации данных, группировки схожих объектов или поиска оптимальных разбиений данных на кластеры.
  + 1. Стохастический алгоритм

Стохастический метод обучения, также известный как метод онлайн-обучения, является одним из способов обучения нейронных сетей, при котором веса обновляются после каждого обучающего примера. В отличие от пакетного метода, где обновления происходят после прохождения всего пакета, стохастический метод обновляет веса непосредственно после каждого примера.

Стохастический метод по сути представляет собой надстройку над методом обратного распространения. Стохастический метод позволяет более быстрое обновление весов сети, что особенно полезно при обучении на больших объемах данных. Он также позволяет нейронной сети быстрее адаптироваться к изменениям в данных, поскольку обновления происходят после каждого примера. Однако стохастический метод может быть менее стабильным и требует тщательной настройки параметров для достижения хороших результатов.

Стохастический метод предполагает особую подачу входных данных. Набор тестовых значений перемешивается и подаётся в на вход нейросети случайным образом. Один и тот же набор данных может подаваться несколько раз со разным случайным порядком элементов каждый раз. Таким образом, мы увеличиваем шанс, что нейросеть адаптируется под решение задачи, а не под конкретные данные или порядок входных данных.

Большой минус такого подхода заключается в том, что не обходимо обучать нейронную сеть на очень большом наборе данных, а при условии, что вычисления для каждого этапа требуют огромного количества вычислений, на обучение сети по данному методу уходит большое количество времени и вычислительной мощности компьютера.

Поэтому существуют вариации данного алгоритма. Один из способов оптимизации данного метода - это корректировка весов не после каждого примера, а после пакета данных. Пакетом данных условно называется группа из нескольких примеров входных данных. К примеру в одном пакете может быть 10 элементов, после прохождения которых будет совершена одна корректировка весов вместо 10, как было бы при использовании стохастического метода. Такой подход, хотя и понижает точность обучения, но значительно ускоряет обучение, и при правильно подобранном размере пакета, модель нейросети в конце всё равно приходит к тому же локальному минимуму, к которой пришла бы модель с корректировкой после каждого примера.

* + 1. Гибридный метод

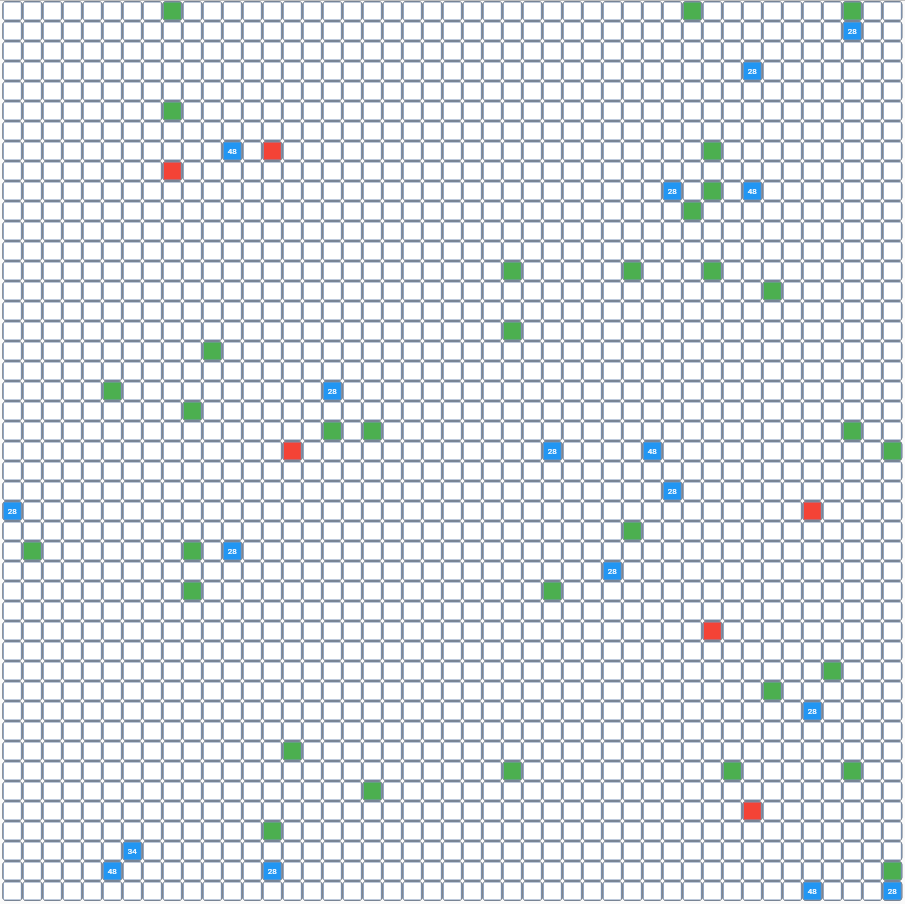
Возможно, что для какой-то специфичной задачи не подходит ни один из существующих методов обучения в их существующем виде. Гибридный метод обучения представляет собой комбинацию различных методов обучения нейронных сетей. Вместо использования только одного метода, гибридный подход объединяет преимущества различных методов с целью улучшения процесса обучения и достижения лучших результатов.

Например, гибридный метод может включать в себя комбинацию методов пакетного обучения и стохастического обучения. В начале обучения может быть использован пакетный метод для быстрой инициализации весов и быстрого приближения оптимальных значений, затем стохастический метод для более точной настройки весов сети.

Гибридные методы также могут включать комбинацию различных алгоритмов оптимизации, функций активации, архитектур сетей и других параметров. Цель состоит в том, чтобы создать оптимальную комбинацию методов, которая учитывает особенности задачи и обеспечивает более эффективное и точное обучение нейронной сети.

1. Реализация
   1. Описание мира

На рисунке представлен мир игры, в которую будет помещена нейросеть.

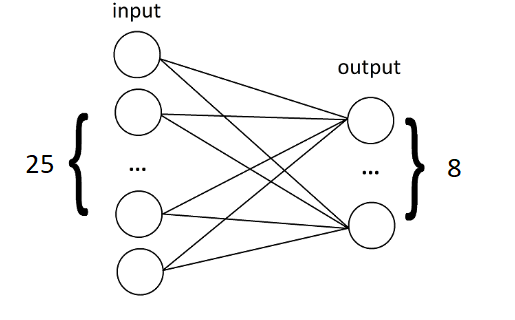


1. Поле игры 45 на 45

Зелёные клетки - это клетки пищи, красные - клетки яда, синие - это боты, то есть сама нейросеть. Число на поверхности бота - его здоровье. Максимум может быть 100 здоровья. Каждый ход бот делает одно передвижение в любом направлении за ход, после этого очередь хода переходит к следующему боту. У мира нет границ: перемещаясь за край экрана, бот появляется на противоположной стороне. Если бот попадает на клетку пищи, то он её «съедает» и восстанавливает 30 здоровья. Попадая на клетку яда, бот «съедает» яд и теряет 30 здоровья. Клетки здоровья и яда, случайным образом появляются на свободных полях каждые несколько ходов. Когда здоровье бота становится меньше 0, он умирает и исчезает с поля. С уменьшением количества ботов клетки пищи появляются реже.

* 1. Модель нейросети

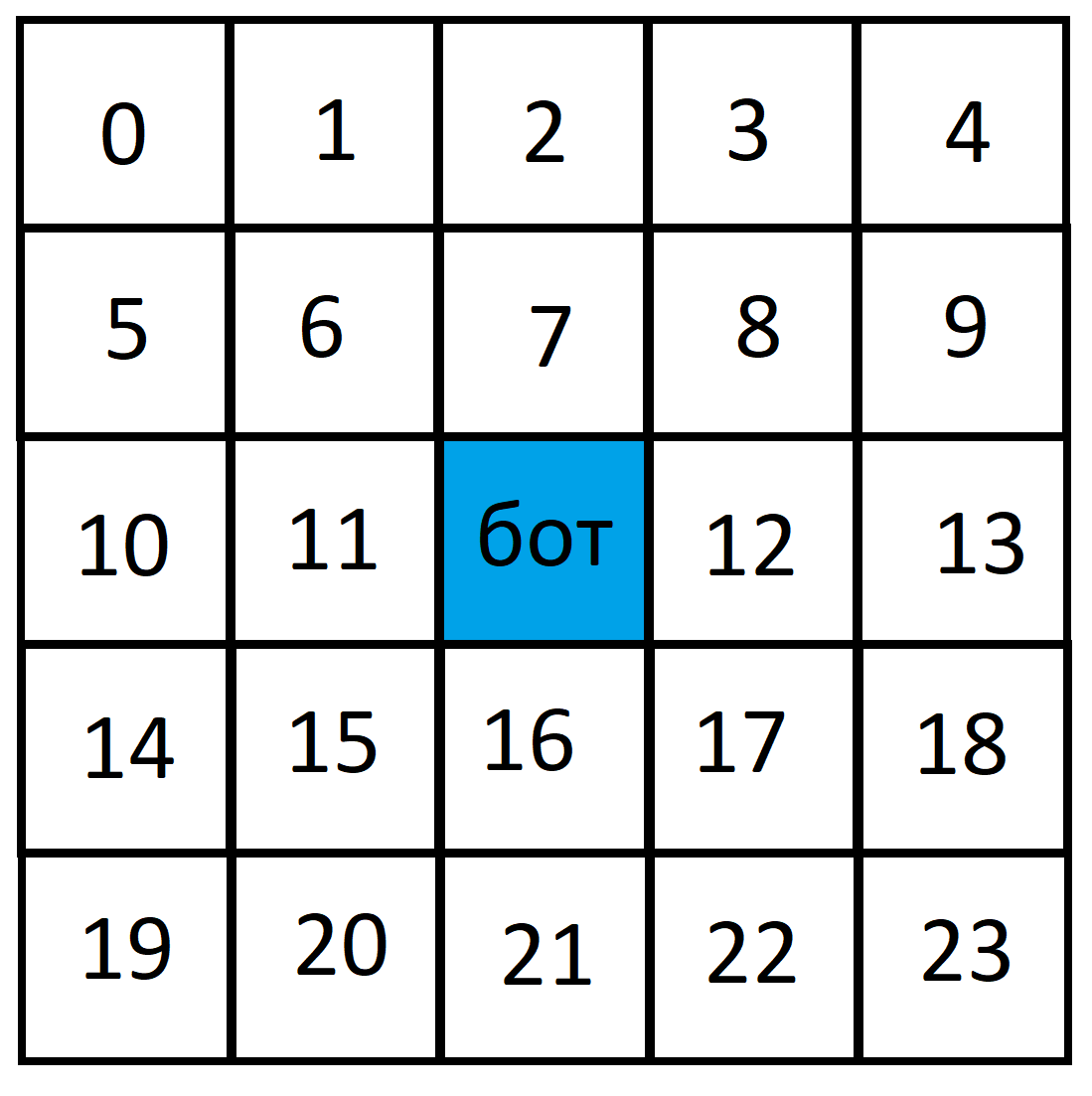
На рисунке представлена модель нейросети



1. Модель нейросети

На вход в первый слой нейросети поступают 25 числа. Первые 24 числа представляют собой 24 поля вокруг бота. Если мы возьмём квадрат 5 на 5 клеток, где центральная клетка - это бот, то начиная слева направо и сверху вниз мы приписываем каждой клетке свой индекс, пропуская клетку самого бота. На вход поступает -1, если на клетке яд, -0.5 если другой бот, 0, если пусто и 1, если на клетке пища. 25 нейрон необходим для того, чтобы избежать локальных минимумов во время симуляции (см пункт 3.3.4).

На рисунке 9 графически показано зрение нейросети.



1. Зрение бота

При случайной инициализации весов и пороговых значений между нейронами связям, исходящим из клеток, непосредственно касающихся бота (клетки 6,7,8,11,12,15,16,17), присваивается значение в 4 раза большее по модулю.

Выходных нейронов 8 - каждый отвечает за одно из направлений, в которое может пойти бот.

В качестве сумматорной функции используется стандартная функция, рассмотренная ранее, в качестве функции активации используется сигмоид.

* 1. Реализация генетического алгоритма

Генетический алгоритм хорошо подходит для решения данной задачи. Мы имеем заданные условия, для которых нам не известен оптимальный алгоритм, следовательно нам уже не подходит алгоритм обратного распространения ошибки. Да и сама задача по сути поставлена, как задача эволюции - вывести лучшего бота для данных условий. Имитируя алгоритм естественного отбора и эволюцию, мы можем вывести такого бота.

Пройдёмся по шагам алгоритма и рассмотрим их реализацию.

* + 1. Приспособленность

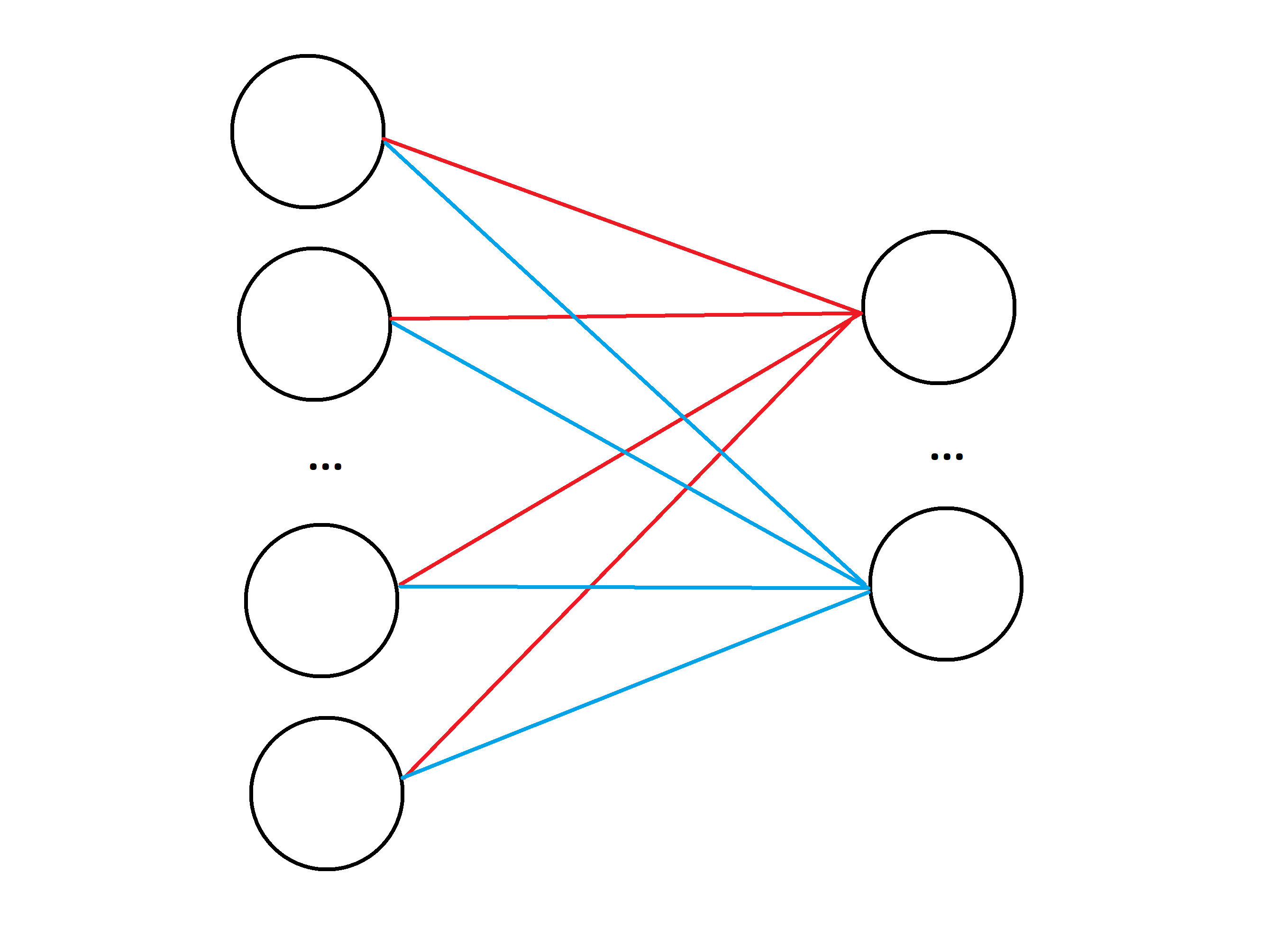
Во время симуляции боты собирают информацию для высчитывания приспособленности. Исходя из поставленной задачи, мы хотим получить бота, способного выжить наибольшее количество ходов. Это задача для ранней стадии, когда боты ещё ничего не умеют и совершают случайные действия. Следующая задача - получить самого приспособленного бота. Показатель на который мы можем посмотреть - среднее здоровье. Чем больше у бота здоровья в любой момент, тем больше его шансы выжить. Как ни странно, оба эти показателя можно учесть одной формулой и учитывая только одно значение. Формула для присобленности F:

В присобленности мы считаем количество пищи, съеденной ботом. Количество ходов, которые проживёт бот полностью зависит от того, сколько еды съест бот, ведь бот, съевший 0 пищи, погибнет через 100 ходов. Что касается второго параметра - чем больше еды ест бот, тем больше будет его серднее здоровье. Таким образом мы стимулируем оба параметра за счёт формулы одной переменной.

* + 1. Селекция и скрещивание

Посчитав приспособленности каждого бота, мы строим новое поколение ботов. Мы отбираем n/2 лучших ботов, где n - это количество ботов в одном поколении, они автоматически попадают в следующее поколение. Оставшие n/2 поколения - это боты образованные скрещиванием ботов из прошлого поколения.

Скрещивание происходит следующим образом: Мы берём половину связей от первого родителя и половину от второго. Причём связи сортируем по принципу, когда мы берём все связи входящие в выходной нейрон родителя.



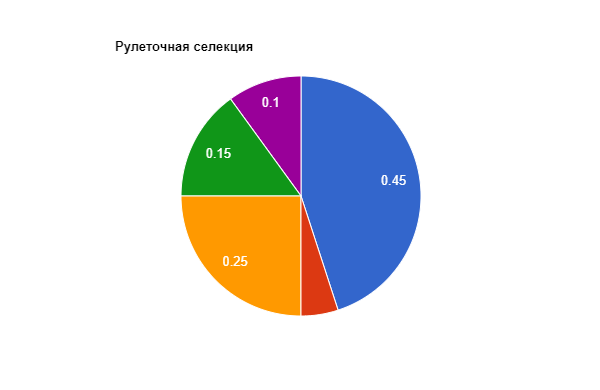
1. Выбор связей для потомка. Синие связи - связи первого родителя, красные - второго родителя

Таким образом мы получаем бота со смешанными характеристиками обоих родителей.

* + 1. Селекция

Чтобы увеличить шансы получения хороших потомков, мы хотим выбрать лучших ботов, но возможно, что у ботов, имеющих низкую приспособленность есть полезные гены, которые могут быть полезны боту с самой высокой приспособленностью. Чтобы получить хорошее распределение ботов в следующем поколении используется рулеточный отбор.

Суть отбора: чем больше приспособленность бота, тем больше шанс, что он будет выбран родителем для скрещивания. Точный шанс стать родителем для k-того бота - , где в числителе присобленность k-того бота, а в знаменателе сумма присобленностей поколения. Таким образом, боты с высокой приспособленностью намного чаще выбираются родителями, но есть шанс, для обновления генетического материала у потомков.



1. Пример рулетки селекции

Однако это не единственный способ обновить генетический материал потомков.

* + 1. Мутация

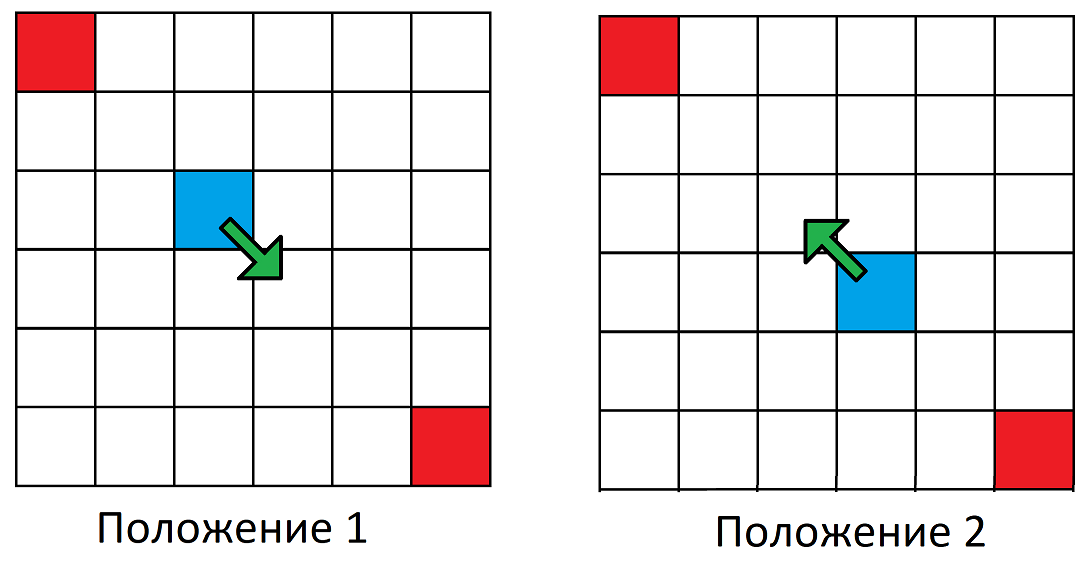
С некоторым шансом, потомок может мутировать: определённая часть его связей может случайным образом увеличиться или уменьшиться. С шансом 50 на 50 происходит одна из двух мутаций:

* Безопасная мутация: меняется малая часть связей, незначительно поменяет поведение бота, необходима для поиска более оптимального решения;
* Экспериментальная мутация: меняется значительная часть связей. Подобная радикальная мутация увеличивает «генофонд» поколения, и с некоторым шансом мы можем получить бота с уникальным качеством, которого нет ни у одного другого бота, и которое трудно бы было получить постепенными изменениями;

Иными словами безопасная мутация - это постепенные поступательные изменения, в то время как экспериментальная мутация - это поиск принципиально новых подходов.

* + 1. Выход из локального минимума

Главная причина гибели ботов - цикличное поведение (Рисунок 10).

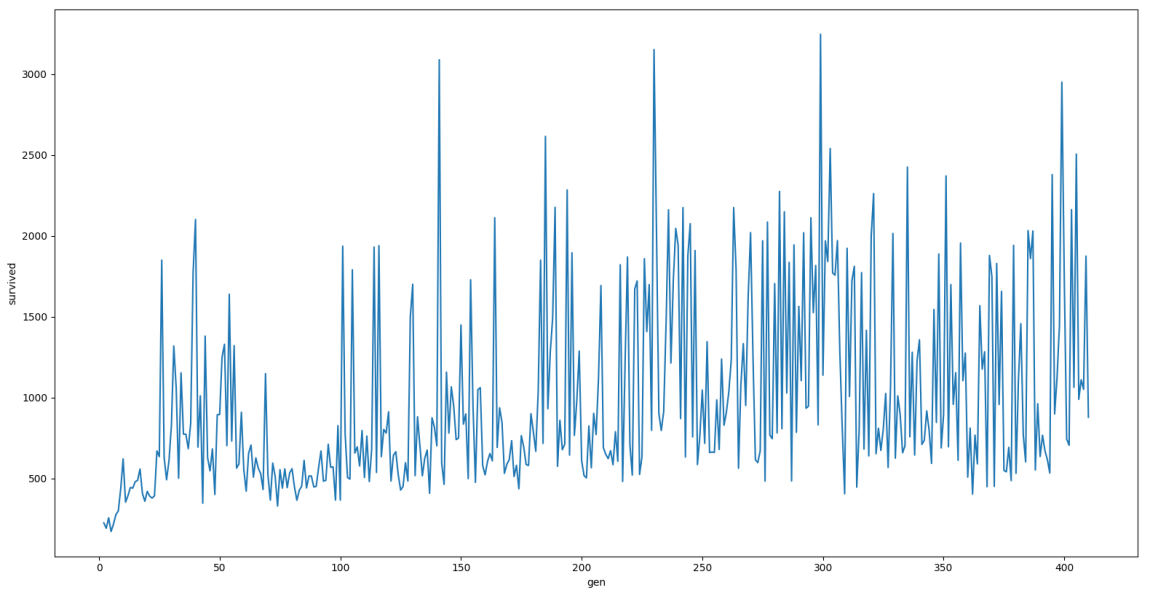


1. Цикличное поведение

Нейросеть находит лучший выход из положения 1 - переместиться в положение 2, из положения 2 - в положение 1. Таким образом бот застревает в бесконечном цикле, пока не погибает, или пока не изменится обстановка вокруг, например, если в поле зрения бота появится новая клетка пищи, клетка яда или другой бот, и входные данные меняются, позволяя выбраться из цикла. 25 входной нейрон работает примерно по такому же принципу: в обычных условиях, нейрон имеет значение 0, если распознаётся цикличное поведение, то значение нейрона уменьшается на 0.1 с каждым ходом, пока бот не сделает другое действие, тогда значение снова станет равно 0

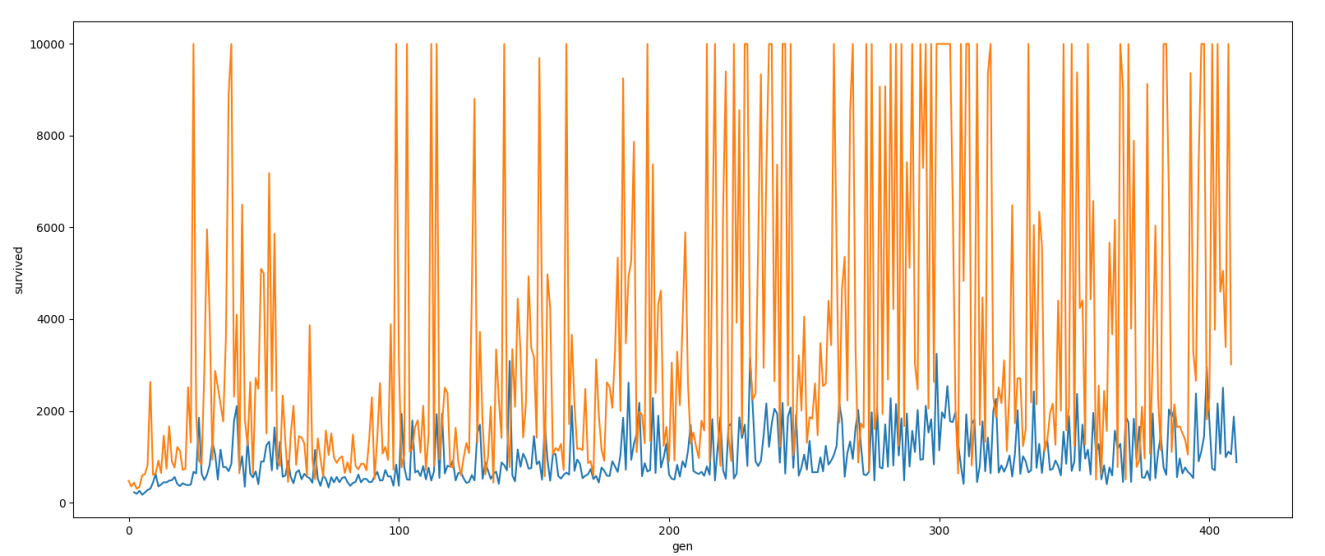
* 1. Выводы о модели

На рисунке 13 ниже представлен график обучения нейросети. Ось OX отражает поколение, ось OY - среднее количество ходов, пережитых ботами. График очень многое говорит о заданной модели, неравномерность объясняется следующим образом: после взрывного улучшения показателей, когда у одного или нескольких ботов значительно вырастает показатель приспособленности, во время селекции будут значительно чаще выбираться именно эти боты, и всего через пару симуляций поколение уже заполнено их потомками, а так как поколение не способно полностью прокормиться с начальными ресурсами, в один момент клетки пищи начинают заканчиваться и появляющихся клеток не хватает на всех, поэтому боты медленно умирают от голода.



1. График обучения

Эта гипотеза доказывается следующим графиком (Рисунок 14).



1. Среднее (синий цвет) и максимальное (оранжевый цвет) количество пережитых ходов

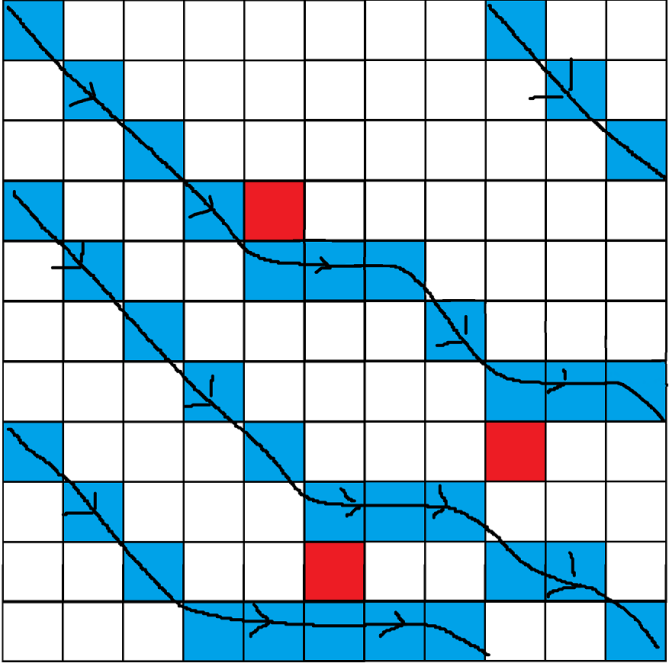
Важное замечание: после 10000 ходов симуляция останавливается и запускается новое поколение.

Как видно из графика максимальное количество ходов многократно больше среднего количества ходов. Объясняется это тем, что бот, достигший лучшего результата остался один, благодаря чему, он может жить бесконечно, так как ресурсов более чем хватает на одного или нескольких оставшихся ботов, поэтому мы считаем, что по достижении отметки в 10000 ходов, бот может жить бесконечно и останавливаем симуляцию.

Также заметна тенденция: после того, как бот достигает отметки высокой максимальной отметки - сразу после максимальное количество пережитых ходов рекордно низкое. Это очередное подтверждение, что всё поколение теперь является потомками самого приспособленного бота и теперь в поколении нет явного фаворита, который остаётся один в конце симуляции. После чего, снова идёт пиковое значение - благодаря удачной мутации в поколении появился новый фаворит, и так по кругу.

Таким образом, развитие идёт большими скачками. В поздних поколениях боты намного агрессивнее и быстрее - выживает бот, который за наименьший срок съедает наибольшее количество клеток пищи. В первую очередь это обеспечивает не выживание бота, а смерть его конкурентов, ведь чем меньше еды останется, тем больше конкурентов погибнет, и тем выше шансы на выживание.

Кроме того ботами выработана интересная оптимальная стратегия для квадратного поля, на котором производилась симуляция. Боты не имеют никакого понятия о том, как выглядит поле и где они на нём находятся, но выработанное поведение является самым эффективным в данной модели: боты движутся по диагонали, съедая все клетки пищи, которые они находят по пути и обходя клетки яда, но обходят они их по особенному - когда клетка яда находится справа от бота, бот обходит клетку снизу или сверху смещаясь не несколько клеток вправо, и таким образом перемещаясь на соседнюю диагональ и тем самым охватывая большую часть карты. Удивительный и неожиданный факт - бот использует клетки яда, как ориентир для ориентации в пространстве.

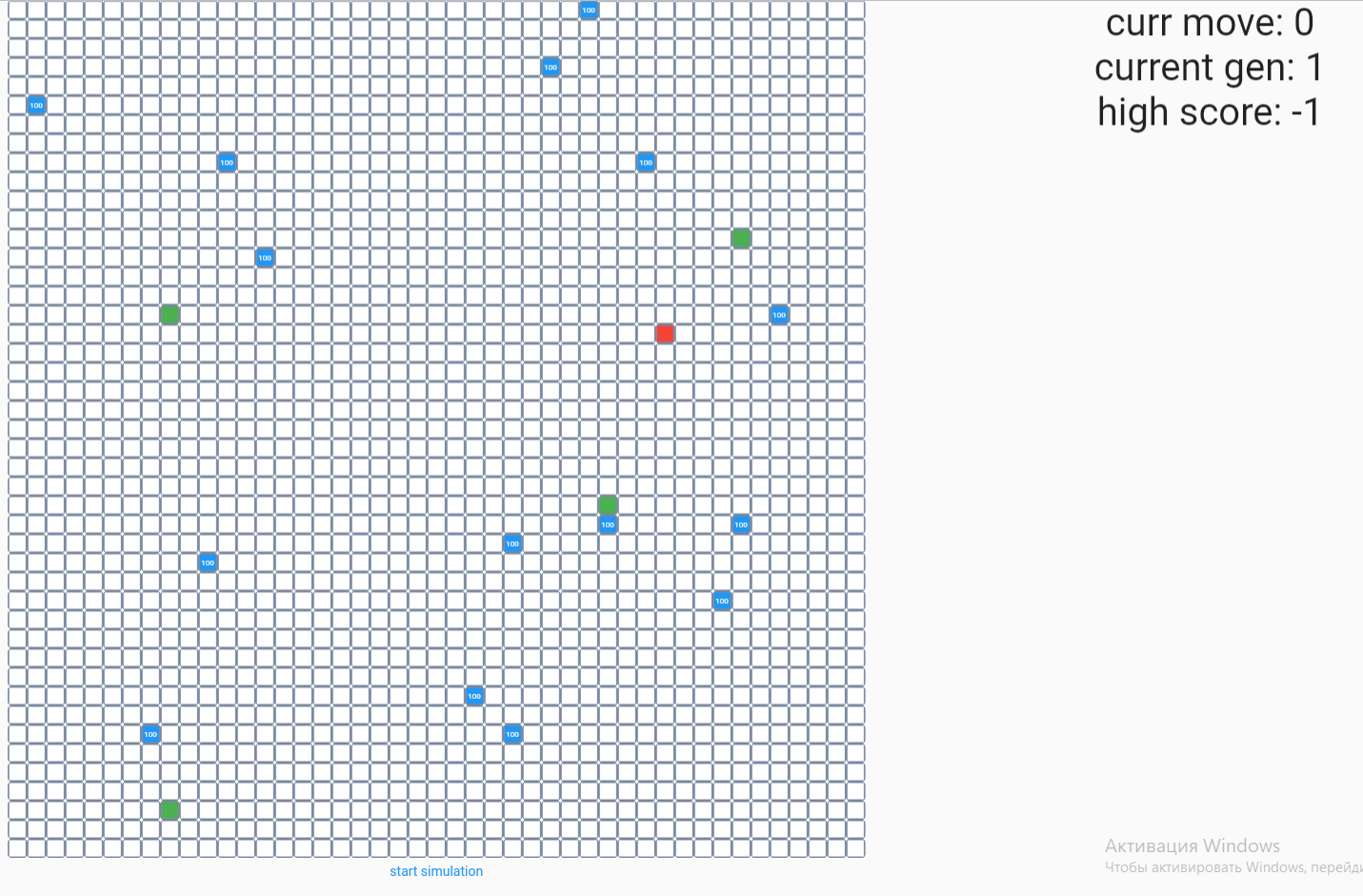


1. Оптимальный маршрут бота для поля 11 на 11

Можно сделать интересный вывод, что любое изменение модели, даже то, которое должно ухудшить ситуацию - это информация, которую нейросеть может использовать для своего преимущества.

* 1. Реализация интерфейса

На рисунке ниже представлен интерфейс программы



1. Интерфейс программы

Слева представлено поле симуляции, под которым находится кнопка «start simulation».

Кнопка запускает цикл программы:

* Симуляция;
* Новое поколение;
* Перезапуск.

Справа на экран выводится статистика: текущий ход, максимальный ход, до которого дожил бот и текущее поколение.

Заключение.

В данной курсовой работе были исследованы основные принципы работы нейронных сетей, рассмотрены различные методы обучения, включая метод обратного распространения, генетический алгоритм и метод упругого распространения. Была предложена и реализована модель на основе генетического алгоритма для решения конкретной задачи.

В процессе работы были поставлены следующие задачи: описание основных принципов работы нейронных сетей, разбор общего принципа работы алгоритмов обучения, реализация генетического алгоритма для решения конкретной задачи. Для достижения поставленных целей был проведен обзор литературы по нейронным сетям, изучены основные алгоритмы обучения и реализован соответствующий программный код. Полученные результаты могут быть полезны для дальнейших исследований и применения нейронных сетей в различных областях.

В заключении можно отметить, что нейронные сети являются мощным инструментом для решения сложных задач, и дальнейшее их развитие и исследование имеют большую перспективу.

Список использованных источников

1. Царьков Михаил, Создание «искусственной жизни» [Электронный ресурс] / Царьков Михаил // Хабр: науч. журн. – 2018 – Режим доступа: https://habr.com/ru/articles/418545/ (Дата обращения: 05.03.2023)
2. 3Blue1Brown: Neural networks [Электронный ресурс].– Режим доступа: URL: https://www.3blue1brown.com/topics/neural-networks – Заглавие с экрана.–(Дата обращения 10.04.2023).
3. Арнис, Нейронные сети для начинающих [Электронный ресурс] / Арнис // Хабр: науч. журн. – 2016 – Режим доступа: https://habr.com/ru/articles/312450/ (Дата обращения: 10.04.2023).
4. Олег Бунин, Введение в архитектуры нейронных сетей [Электронный ресурс] / Олег Бунин // Хабр: науч. журн. – 2017 – Режим доступа: https://habr.com/ru/companies/oleg-bunin/articles/340184/ (Дата обращения: 20.04.2023).