МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионально образования

“Московский авиационный институт (национальный исследовательский университет)”

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРИКЛАДНОЙ МАТЕМАТИКИ**

**Направление подготовки Фундаментальная информатика и информационные технологии**

Реферат по курсу:

Фундаментальная информатика

на тему:

Генетические алгоритмы

Работу выполнил

студент 1 курса

бакалавриата

очного отделения

Группы М8О-110Б

Гамов Павел Антонович

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. 806 Никулин С. П.

Подпись студента, число

План работы и план написания реферата

1. Общее введение и пояснение что такое машинное обучение
2. Введение в понятие
3. Зачем они нужны
4. Где применяются
5. Какими бывают
6. Принцип
7. Понятие нейрона и активации
8. Активационные функции сигмоиды и зачем они нужны
9. Понятие веса и способы их хранения в памяти
10. Сложение весов по входным данным и активация нейрона
11. Понятие хайден леер
12. Добавляем больше инпутов и уровней и как это будет выглядеть
13. Selection crossover mutation new generation на примере моего кода
14. Selection

* Введем понятие fitness function

Чем лучше проявила себя нейронная сеть тем больше у нее фитнесс таким образом происходит селекция или выборка между элементами поколения нейронных сетей

* На основании сравнения фитнесс функции мы запихиваем все поколение в мусорку и выбираем рандомно тех кто лучше потому что (тут принцип выборки на основании фитнеса и мусорного ведра)

1. Crossover

* Мы получаем некое кол во хороших нейронных сетей с оптимальными весами и нам надо их скрестить
* Привести пример из реальной жизни с животными и их скрещиванием генокода
* Из сайта взять несколько примеров с картинками кроссовера и напистать про них
* Таким образом мы получаем детей из родителей

1. Mutation

* Понятие мутации введение и пояснение что это за херня
* Виды мутаций и их mutation rate с сайта
* Тут можно затронуть насколько можно сильно loss function и мутацию весов в их лучшую сторону для разных видов нейросети

1. New generation

* Построение нового поколения на основании мутированных лучших парней
* Запуск новой селекции
* Достижение цели для безошибочной работы нейронной сети

Генетические алгоритмы

Раздел 1

Основные понятия

Введение

Генетические алгоритмы или машинное обучение – сфера программирования, изучающая способы решения некоторых задач или проблем используя специальные алгоритмы, способные к самообучению и эволюции.

Сама идея таких алгоритмов схожа с эволюционной теорией Дарвина, где существует такое понятие как эволюция - совершенствование организма, адаптация и его развитие.

Ingo Rechenberg – немецкий ученый в области бионики впервые описал понятие “Стратегия эволюции” в 1960 году. Его идея возможности использования таких алгоритмов для решения задач была подхвачена другими учеными, и в 1975 году John Holland и его студенты разработали генетические алгоритмы (Genetic Algorithm’s), которые были позднее описаны в книге под названием “Adaption in Natural and Artificial Systems”.

John Koza в 1992 году применил генетический алгоритм для решения конкретной задачи и назвал этот метод генетическим программированием (Genetic programming).

Области применения

Многие люди испытывают страх по отношению к машинам, которые умеют обучаться. Они считают, что такие программы могут быть опасны, некоторые считают, что такие машины захватят мир, но пока это не очень возможно, позже мы узнаем почему. (написать почему же?)

В то же время эти люди каждый день листают ленту Facebook, видят рекламу на сайтах, находят интересующие их видео в рекомендациях YouTube, используют так популярные нынче языковые поисковики как Siri или Alexa. Все это алгоритмы, которые учатся пока мы ими пользуемся.

У всех бывал такой случай, когда мы ищем, к примеру фотографию котика, а через 10 минут наш Facebook заполнен рекламой ветеринарных клиник, рекламой новых кошачьих причесок и новых видов кормов, а у вас даже кота нет!

Что же случилось? Дело в том, что, когда вы умилялись фотографией котиков, машина, которая определяет какую рекламу вы будете сегодня видеть, получила информацию о том, что вы заинтересованы в данной тематике, а следовательно, вам будет более интересна реклама причесок котиков, чем реклама новых газонокосилок.

Генетические алгоритмы настолько популярны в виду их адаптивности к решениям различных задач, что их применение можно найти буквально везде.

Некоторые задачи, в выполнении которых востребован человек, также могут быть решены с помощью генетических алгоритмов.

Виды генетических алгоритмов

Принцип построения генетического алгоритма схож почти во всех случаях. Отличия можно найти в стиле и способе обучения генетического алгоритма. Можно выделить два вида обучения – контролируемое и неконтролируемое.

# Контролируемое обучение происходит на основании некой базы данных, которая скармливается алгоритму для дальнейшего обучения. Данный способ чаще реализуется в Свёрточных нейронных сетях (Convolutional Neural Network).

# Неконтролируемое обучение – главная тема этого реферата. Такой способ реализуется без участия человека. Можно запустить программу, уйти на какое-то время и вернувшись обнаружить, что наша сеть обучена и готова решать поставленную задачу идеально и безошибочно.

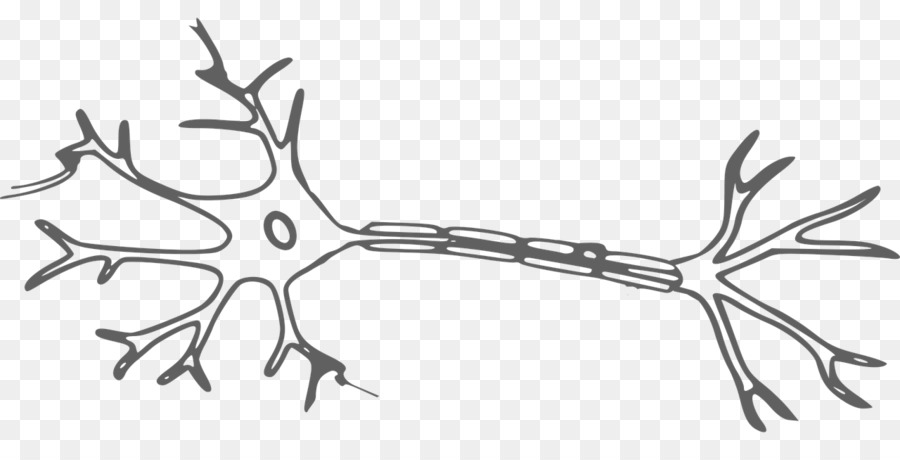
Раздел 2

С чем имеем дело?

Принцип и устройство

Рассмотрим человеческий мозг, он состоит из множества нейронов, по которым бегают заряды электричества. Кто – то увидит в нем чудо, я увижу в нём компьютер, который можно воссоздать, хоть и не полностью.

Спустимся до составляющих компонентов - нейронов.



(найти норм фотку)

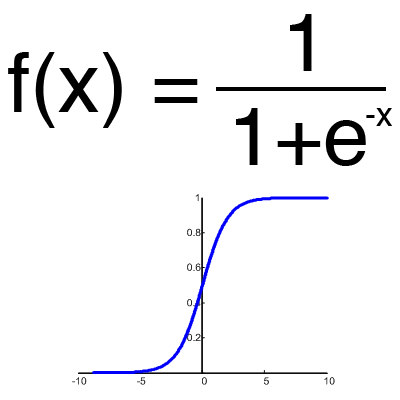
Мы не будем вдаваться в его устройство, все что нам надо знать это то, что он принимает сигналы, обрабатывает и, в зависимости от того, что получил нейрон, активируется или не активируется, соответственно передавая сигнал дальше или не передавая вовсе.

Давайте представим то, что он получает сигнал некоторой мощности. Вопрос – как обозначить его силу? Мы представим его в виде числа от -1 до 1, где -1 отсутствие сигнала, а 1 максимальное его значение. Таким образом мы обозначили входные данные.

Вернемся к нашему нейрону, он, взяв входные данные принимает решение передавать сигнал дальше или нет. Для получения выхода нейрона нам нужно анализировать входные данные, для этого обратимся к логистическим функциям, а именно Сигмоидам

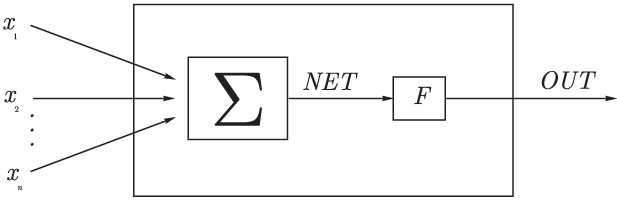
Сигмоида – функция, применяемая в машинном обучении для подсчета результата обработки нейроном входных данных.

Простейшая из них



Данная сигмоида принимает в качестве аргумента сумму входных данных и если f(x) больше 0.5 наш нейрон активируется и передает сигнал дальше (output = 1), если сумма меньше 0.5, то нейрон остается не активированным и сигнал соответственно дальше не проходит (output = 0).

Что же получается, теперь нейрон уже не выглядит таким сложным. Его можно представить в виде вот такой схемы.



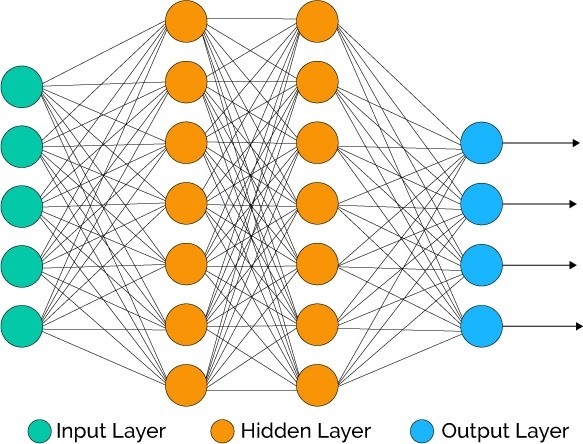
И тут мы вспоминаем про наш Генетический алгоритм. Мы видим данные, которые принимает наш маленький мозг, функция которая обрабатывает сумму этих входных данных и выдает true или false, но где здесь ген нашего мозга? Мы пока что не можем его чему-то научить, потому что учить нечего. Нам требуется ввести понятие веса (weight).

Вес – некое число в любом интервале, обычно берут все тот же [-1, 1], которое определяет значимость входного данного. Количество этих весов как правило равно количеству входных данных.

Если нейрон имеет на входе 2 числа входных данных (х1, х2) то пусть его 2 веса будут записаны в массив [w1, w2]. Таким образом сумма Х, которая будет передаваться в сигмоиду будет находиться как х1 \* w1 + x2 \* w2.

И вот тут начинается самое интересное, теперь мы имеем что менять, у нас есть массив весов каждого входного данного, если наш нейрон активируется тогда, когда не надо, мы всего лишь изменим веса каждого входного данного на что-то и поведение нашего нейрона изменится. Таким образом мы можем сказать, что веса является подобием генетического кода нашего мозга.

Добавив еще больше нейронов и входных данных, мы сможешь сделать довольно сложный мозг, который принимает значения, обрабатывает и дает нам в итоге какой-то выходной массив данных.



Следует пояснить что из себя представляет Hidden layer (скрытый уровень), это те нейроны, которые не видны пользователю, то, что происходит внутри.

Взять человеческий мозг, к примеру, ученик сидит на диктанте и слышит слово программирование, это его входные данные, они сложным образом проходят по всему мозгу, нейроны активируются и вот он написал на листке бумаги слово, которое он услышал. Таким образом hidden layer его мозга это недоступный нам этап принятия им промежуточного решения.

Теперь, когда мы узнали базовые понятия структуры мы можем поговорить про применение машинного обучения и генетического алгоритма для решения некоторой задачи.

Раздел 3

Обучение машины игре Flappy Bird

Подготовка

Для рассмотрения я выбрал задачей научить мою систему научиться играть в Flappy Bird.

Описание игры:

Существует птица и движущиеся ей навстречу прямоугольники в которых есть отверстие.

Существует гравитация, которая постоянно тянет птицу вниз.

Нажатием кнопки “space” птица будет подпрыгивать относительно своего нынешнего положения.

Цель игры:

Используя прыжок, пройти птицей как можно больше отверстий в прямоугольниках.

Если птица касается прямоугольника - игра заканчивается, и начинается заново.

Если птица падает вниз за пределы экрана - игра заканчивается, и начинается заново.

Для написания кода был использован язык программирования Java Script и библиотека p5.js, где представлены инструменты для графического отображения.

(тут будет пикча графики моей проги на белом фоне в черных тонах)

Играть можно, птица прыгает и умирает, прямоугольники двигаются.

Но вернемся к обучению и создадим мозг нашей птицы.

(тут пикча моего мозга моих птиц 5 инт 5 хид 1 аут)

Я использовал 5 inputs, 5 нейронов в hidden layer и 1 output.

(поясняю про 5 инпутов)

Что касается весов в данной системе то они случайны и находятся в интервале между -1 и 1. Для каждого нейрона я записал веса его входных данных и получил массив из 6 элементов, каждый из которых является массивом, состоящим из 5 элементов. Массив в массиве позволяет представить генетический код птица в виде матрицы, это удобная структура хранения значений.

Я использовал 1 output потому что птице надо решить, основываясь на полученных данных, прыгать ей или нет, если выходной нейрон активен тогда наша птица сама нажимает на пробел и тем самым прыгает, если же ее мозг решил не прыгать, значение её выходного нейрона будет равно 0 и пробел не будет нажат и птица не прыгнет.

Имея птицу, ее мозг и генетический код мы можем приступить к ее обучению.

Selection

Вернемся к теории эволюции Дарвина, организмы непрерывно эволюционируют, отбрасывая плохое и приобретая хорошее. Но как понять, что организм построен хорошо? Все познается в сравнении. Поэтому, пусть у нас будет не одна птица, а целая популяция совершенно разных птиц, генетический код которых на начальном этапе будет сгенерирован абсолютно случайно. Понятно, что в одних и тех же условиях они будут вести себя по-разному и когда все наши птицы умрут мы сможем понять какая из них была самая лучшая, в этом нам поможет Fitness function.

Fitness function – функция подсчета эффективности некого мозга по сравнению с другими в популяции. Пусть у каждой нашей птицы будет переменная которая называется fitness, пока птица живая, ее fitness увеличивается на единичку. И когда все поколение умрет, мы сравним fitness всех наших птиц и найдем птицу с наилучшим этим показателем. Её генетический код будет использоваться в создании новой популяции, так как она показала себя лучше всех в выполнении поставленной задачи: пройти в игре как можно дальше.

Допустим в нашей популяции 3 птицы, 1-ая прожила 100 секунд, 2-ая прожила 200 секунд, 3-яя прожила 400 секунд. Нам нужно найти 2 наилучшие птицы (маму и папу). (почему две мы узнаем дальше.) Это можно сделать очень просто взяв две птицы с наибольшим fitness значением, это простой способ можно сделать более похожим на естественный отбор, предствив некую емкость куда мы положим 100 копий первой птицы, 200 копий второй птицы и 400 копий третей птицы. Закрыв глаза, мы вытащим одну птицу из этой емкости, у нас будет ~14% достать 1-ую птицу, ~29% достать 2-ую птицу и ~57% достать 3-ью птицу. Уже больше похоже на реальную жизнь, ведь теперь даже самый неудачник имеет шансы дать потомство.

Так мы дела ем столько раз, сколько генетических кодов мы хотим перенести в следующее поколение птиц.

Crossover

Что мы имеем? Две (или более) птицы, отобранные с помощью selection function, с оптимальными (или пока еще нет) генетическими кодами, весами. Можно создать Offspring.

Offspring – новая машина, полученная скрещиванием генетического кода двух (или более) машин с предыдущего поколения.

Принцип базируется на реальном примере скрещивания генетического кода живых организмов, когда часть генетического кода одного представителя дополняет часть генетического кода другого организма.

(фотки с днк)

Применим принцип для создания нового генетического кода offspring. Из генетического кода папы, представляющего массив весов, мы возьмем 1/2 часть этих данных и заполним ими первую половину генетического кода наследника, применим тот же для мамы. Таким образом мы получаем новый генетический код, полученный путем скрещивания предыдущих двух.

(пример кода скрещивания функция, принимающая 2 массива и мутирующая их).

Mutation

Теперь, получив новый генетический код нам следует затронуть еще один аспект развития реальных организмов – мутацию.

Мутация – некое действие, произведенное в отношении генетического кода, направленное на его модернизацию.

Мутации как правило случайны, в моем примере получив новый генетический код путем скрещивания я заполняю им новую популяцию, одну птицу я оставляю без мутаций, на случай если все остальные мутированные гены окажутся неудачными и можно будет начать заново (?). Для каждых остальных путем рандомного выбора я меняю некое значение весов на другой в том же интервале.

Как итог, имея задачу, некую популяцию и поочередно выполняя selection, crossover и mutation, мы можем найти через некое количество итераций найти оптимальные веса для решения конкретной задачи