Применение нейронных сетей в робототехнике

# Вступление

В современном мире нейронные сети приобретают все большее значение. Они нашли большое применение в анализе данных, поисковых системах, социальных сетях и прочих сферах. Робототехника также не обходится без применения нейронных сетей. Их применяют для распознавания лиц, речи, обучения роботов ходьбе и другим сложным действиям. В связи с этим мне стало интересно: можно ли найти применение нейросетям в решении задач соревнований по робототехнике.

Объект исследования: нейросети и их применение.

Предмет исследования: компьютерная модель одной из задач конкурса "Юный Робототехник» 2019 года, обученная на решение данной задачи нейросеть.

Цель работы: Исследовать практическое применение нейросетей в робототехнике.

Задачи:

* Создать компьютерную модель задачи и робота
* Обучить нейросеть выполнять поставленную задачу
* Изучить особенности обучения нейросетей

Практическое применение: данный проект доказывает, что нейросети помогают в решении сложных задач, не имеющих очевидного решения алгоритмическим способом.

# О нейронных сетях

История возникновения нейронных сетей

В 1943 году была представлена статья двух выдающихся ученых Уоррена Мак-Каллока и Уолтера Питтса, освещающая математическую модель нейронной сети, а в 1949 году канадский нейропсихолог Дональд Хебб выпустил книгу *"Организация поведения"*, в которой было подробное описание процесса самообучения ИНС.

Позднее, в 1957 году известный американский ученый Фрэнк Розенблатт изобрел персептрон – математическую модель обработки информации человеческим мозгом. Данная разработка уже в те годы умела прогнозировать погоду и распознавать образы, однако вскоре персептрон жестко раскритиковали из-за скандальной работы Марвина Минского, где он разъяснил, в каких задачах персептрон не эффективен. После этого интерес к нейросетям угас, но не на долго.

В 1974 году Пол Вербос разработал алгоритм обратного распространения ошибки, который используется и по сей день для обучения искусственных нейронных сетей. Начиная с 1985 года Джон Хопфилд предлагает миру свое виденье устройства и работы нейросети, которая способна решать некоторые виды задач. Данная работа разогревают нешуточный интерес мировой общественности к искусственным нейронным сетям. В 90-е годы прошлого века алгоритм обратного распространения ошибки получил существенное развитие, за счет чего былая критика Минского касательно неработоспособности сетей была окончательно опровергнута. На сегодняшний день искусственные нейронные сети является главным подразделом машинного обучения.

Определение искусственной нейронной сети

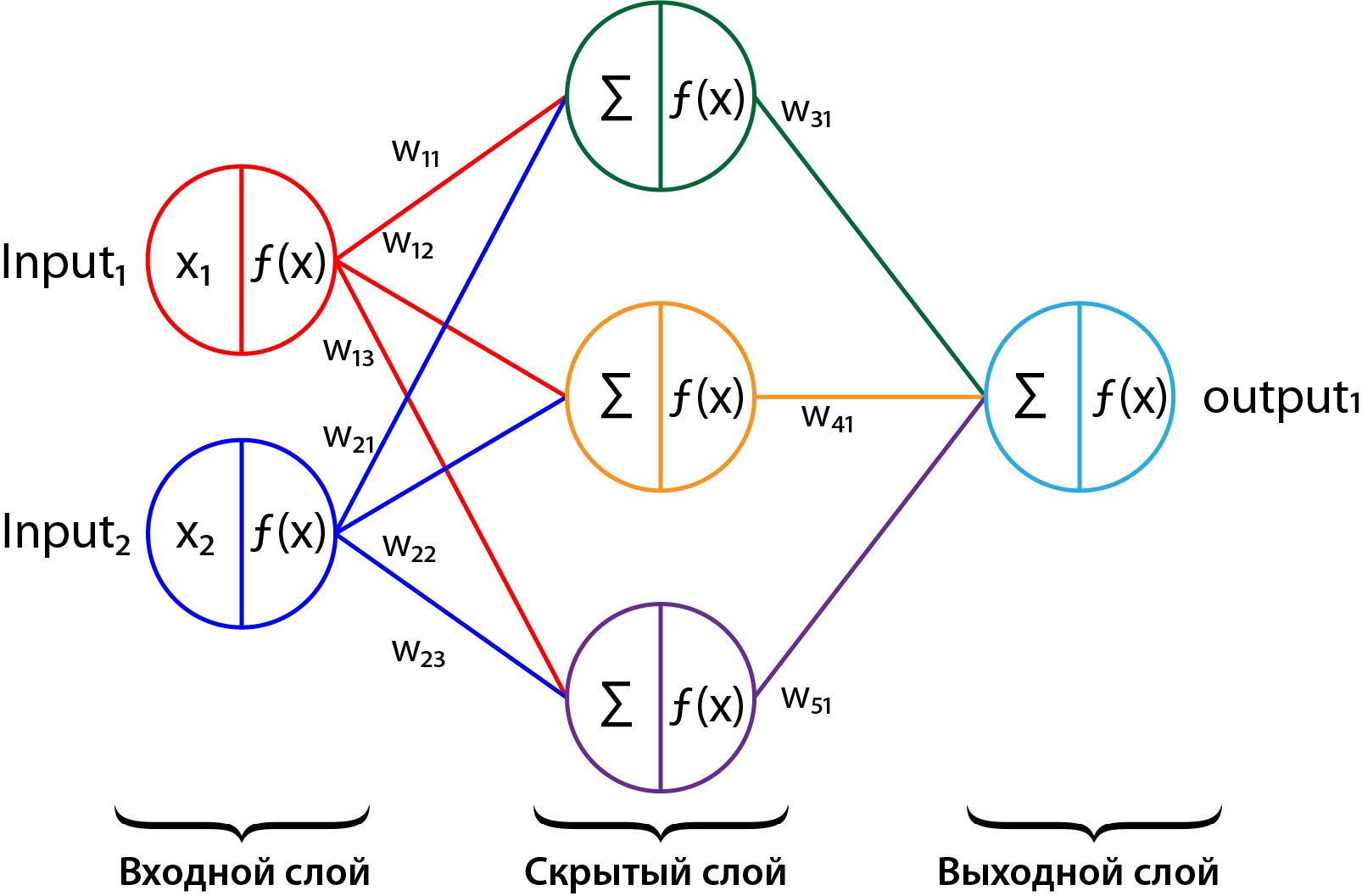
Искусственная нейронная сеть (ИНС) - математическая модель и ее программное воплощение, построенная по принципу биологических нейронных сетей - сетей нервных клеток живого организма.

Принцип работы нейронной сети

Нейронные сети состоят из множества искусственных нейронов, поэтому, чтобы понять принцип работы сети, рассмотрим, как устроен такой нейрон. Строение искусственного нейрона берет свои корни в [работе биологического нейрона](https://dtf.ru/redirect?url=https%3A%2F%2Fhabrahabr.ru%2Fpost%2F214109%2F). Биологический нейрон - это клетка, которая передает электрические разряды в зависимости от поступающей на неё информации, иначе говоря, это простейший вычислительный элемент. Задачу нейрона можно описать так: каждый нейрон решает, насколько важны факты для отдельно взятой небольшой ситуации и передает свое решение другим нейронам.

Искусственный нейрон работает схожим образом. На входе у нас есть некоторые данные. Эти данные передаются по специальным связям-синапсам для дальнейшего суммирования, преобразования и вывода. Все синапсы имеют свой вес (обозначается w) для каждого фактора, на который данный фактор умножается. Таким образом, в сумматоре производится взвешенное суммирование факторов. Затем результат этого суммирования преобразуется с помощью активационной функции (ƒ(x)), и информация передается по специальной связи-аксону для дальнейшего использования. Каждый нейрон может иметь несколько синапсов, но только один аксон.

Но один нейрон может решить только небольшую задачу, поэтому в нейронных сетях используется множество связанных друг с другом нейронов. Схему простой нейросети можно представить так:



Где применяются нейросети

Нейронные сети применяются в следующих задачах:

1. Распознавание образов и классификация (определение к какому классу принадлежит объект (символ текста, звук, картинка)).
2. Принятие решений и управление (принятие решений по входным данными передача их для дальнейшего использования).
3. Кластеризация (разбиение множества входных сигналов на классы).
4. Прогнозирование (предсказывание будущего значения последовательности на основе нескольких предыдущих значений).
5. Аппроксимация (аппроксимация непрерывных функций).
6. Сжатие данных и ассоциативная память (выражение данных большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Ассоциативная память – обратный процесс).
7. [Анализ данных](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C#%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (извлечение знаний из экспериментальных данных).
8. Оптимизация (процесс максимизации выгодных характеристик и минимизации расходов).

Для каждой задачи применяются различные типы нейронных сетей.

Типы нейронных сетей

Существует множество видов нейронных сетей. Вот некоторые из них:

|  |  |  |
| --- | --- | --- |
| Тип нейронной сети | Область применения | Тип обучение |
| Персептрон Розенблатта | Распознание образов, принятие решений, прогнозирование, аппроксимация, анализ данных | С учителем |
| Хопфилда | Сжатие данных и ассоциативная память | Без учителя |
| Кохонена | Кластеризация, сжатие данных, анализ данных, оптимизация | Без учителя |
| Радиально-базисных функций (RBF-сеть) | Принятие решений и управление, аппроксимация, прогнозирование | Смешанное |
| Свёрточная | Распознание образов | С учителем |
| Импульсная | Принятие решение, распознавание образов, анализ данных | Смешенное |

Как мы видим из таблицы каждый вид нейросетей имеет свое применение. Также в таблице указан тип обучения, еще одна важная характеристика нейронной сети. Рассмотрим некоторые типы обучения нейросетей.

Обучение нейросетей

Существует 3 типа обучения нейронных сетей: с учителем, без учителя и смешанный.

При обучении нейросети с учителем нейросети передаются заранее известные результаты решения задач, и она пытается приблизиться к этим результатам. Данный алгоритм применяется при обучении нейросетей классификации и регрессии.

Обучение без учителя применяется, когда заранее неизвестен правильный ответ для решения задачи. К таким задачам относятся кластеризация, анализ данных, ассоциация, принятие решений и управление, сжатие данных и ассоциативная память. При обучении нейросети без учителя нейросети передаются не ожидаемые результаты, а оценка правильности действий нейросети.

Смешанный алгоритм обучения объединяет в себе свойства обучения с учителем и без. Он применяется, когда имеется только часть ожидаемых решений и большое количество данных. К примеру, для анализа медицинских данных: рентгеновских снимков или МРТ.

# Практическое применение нейронных сетей в робототехнике

Вступление

Изучив тему нейронных сетей, я заинтересовался возможностью их применения в робототехнике. Конечно, нейросети и так повсеместно используются для распознавания лиц, обучения роботов ходьбе и прочим действиям, но мне хотелось найти им применение в более простых задачах, например, решение заданий конкурса по робототехнике.

В качестве задания я решил взять одну из дисциплин конкурса «Юный Робототехник 2019» - ориентирование на местности. Обобщая, можно свести данное задание к нахождению фигур на карте, идентификации их и возвращении в гараж. Так же на карте расположены препятствия и стенки, которые робот должен объезжать.

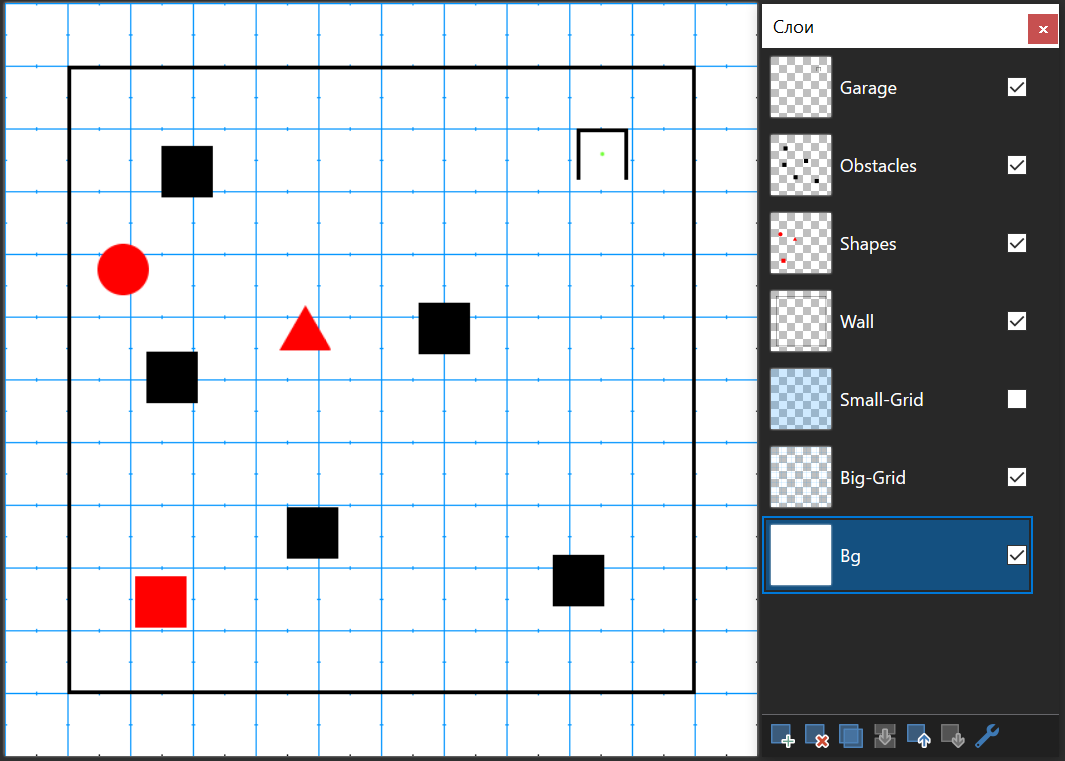
Так как реализовывать данный проект я начал на каникулах, а продолжил во время карантина, я не мог использовать физические модели роботов и карту, поэтому мне необходимо было создать их компьютерные модели.

Создание моделей

В качестве языка программирования для реализации программы я был выбран Python, так как довольно понятный по сравнению с другими языками, имеет отличную производительность при обработке данных и множество библиотек и фреймворков.

Для создания моделей машины и карты (дальше окружения) использовалась библиотека pygame. Pygame — это «игровая библиотека», но подойдет и для создания модели задания и машины.

Приступив к созданию моделей, я нарисовал шаблон карты, где на каждом слое находились различные объекты: гараж, препятствия, фигуры. Таким образом, я облегчил задачу по созданию разнообразных карт.



Затем я нарисовал модель робота и приступил к программированию. Мне необходимо было создать класс робота, обладающего функциями для передвижения, получения данных и взаимодействия с окружением. Вот некоторые из них:

# Класс робота

*class* Car():

#   Объявление переменных робота

*def* \_\_init\_\_(*self*):

#   Вычисление длин радаров - показаний датчиков расстояния

*def* compute\_radars(*self*, *degree*, *road*):

#   Вычисления значений датчиков столкновения с

*def* compute\_collision\_points(*self*):

#   Поворот робота вокруг свой оси, используется при передвижении по карте

*def* rotate(*self*, *angle*):

#   Поворот относительно одного из колес, используется при определении фигур

*def* pivot\_rotate(*self*, *side*):

#   Преобразование показаний датчиков для

*def* get\_data(*self*):

#   Вычисление награды, полученной роботом, необходимо для обучения нейросети

*def* get\_reward(*self*):

#   Проверка нахождения фигуры

*def* check\_shape(*self*, *road*):

#   Поворот к фигуре

*def* rotate\_to\_shape(*self*, *side*, *road*):

#   Определение фигуры

*def* identify\_shape(*self*, *side*, *road*):

#   Обновление положения робота, показаний датчиков и остальных переменных

*def* update(*self*, *road*):

Постановка задач

Когда модель робота и окружение были созданы, необходимо было выбрать тип нейросети и метод обучения. Для этого я разбил задачу робота на части:

* + 1. Нахождение фигур
       1. Передвижение по карте, объезд препятствий
       2. Определение фигуры
    2. Возвращение в гараж
       1. Передвижение по карте, объезд препятствий
       2. Нахождение гаража

Теперь необходимо было определить, какие подзадачи имеют алгоритмическое решение, а какие лучше решать при помощи нейронных сетей. Участвуя в конкурсе в прошлом году, я уже составлял алгоритм определения фигуры, для данной подзадачи нейронная сеть не нужна.

А вот передвижение по карте и нахождение гаража довольно сложно запрограммировать, имея только датчики расстояния. Для них я и решил использовать нейросеть.

Теперь необходимо определить тип задач. Что необходимо выполнить роботу в каждой из подзадач? Для передвижения по карте, робот должен анализировать данные с датчиков и в соответствии с ними осуществлять движение по карте. Эта задача подходит под определение принятия решения и управления. Следующая задача – нахождение гаража. Выполняя ее, робот должен по данным датчиков определить гараж перед ним или нет. Данная задача является классификацией.

Обучение передвижению по карте

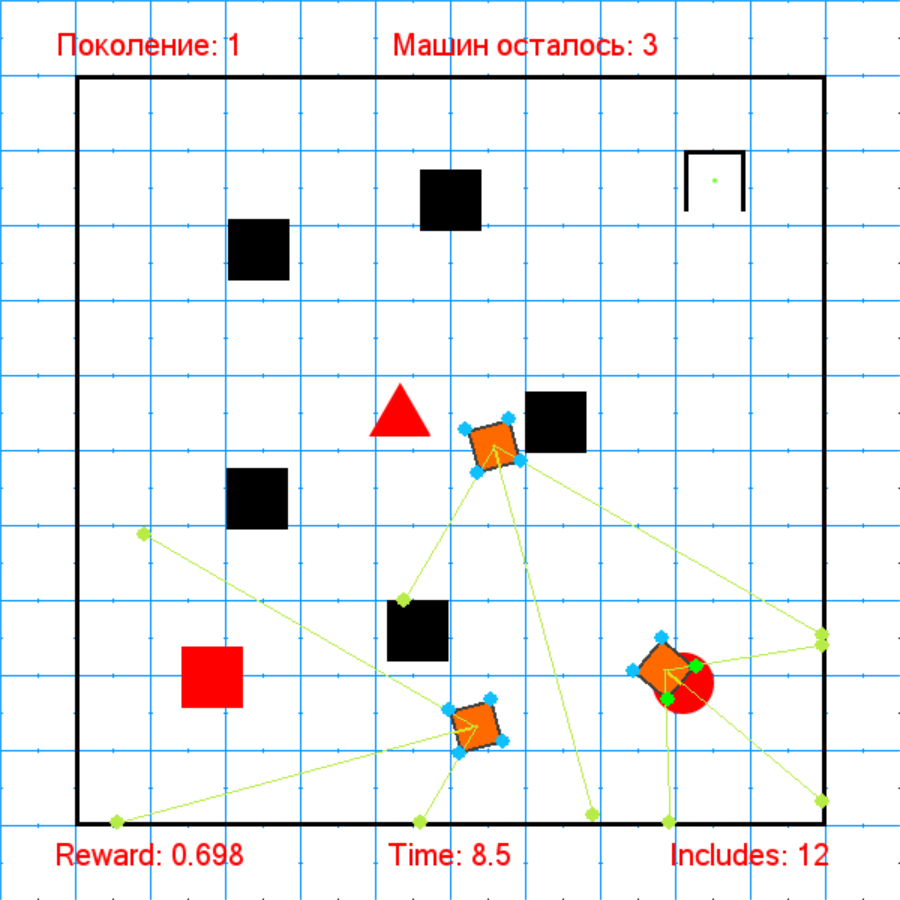
Определив типы задач, я решил начать обучение робота с передвижения по карте. Так как это задач принятия решения и управления, необходим алгоритм обучения без учителя. Одним из таких алгоритмов является эволюционный алгоритм или NEAT.

Кратко работу данного алгоритма можно описать так: алгоритм берёт группу особей, которые пытаются выполнить поставленную задачу. По итогу выполнения задачи каждая особь получает награду. Особи с самой большой наградой «скрещиваются» для создания следующего поколения. Этот процесс продолжается до тех пор, пока не появится поколение, способное выполнить задачу.

Для обучения роботов я использовал библиотеку NEAT-Python – реализацию алгоритма NEAT на языке python. В ходе обучения передо мной встали следующие вопросы: какие данные передавать нейросети, какие данные получать от нейросети и как награждать роботов за выполнение задания.

Передача входных данных

Чтобы определить, как передавать нейросети данные, необходимо понять какие данные у нас есть. Наш робот может оперировать только датчиками расстояния, касания и цвета. Датчик цвета в передвижении не участвует, датчики касания нужны только для определения столкновения. Тогда, для получения данных, нам остается использовать только датчики расстояния. Но какое количество датчиков будет наиболее эффективным? Это можно узнать только опытным путем. Запуская поколения с разным количеством и расположением датчиков, я выявил, что быстрее и лучше приспосабливаются особи с тремя датчиками, расположенными под углами -45, 0 и 45 градусов относительно центра.



Выходные данные нейросети

Чтобы понять, какие данные должна выдавать нейросеть, необходимо определить действия, необходимые роботу для движения. Для передвижения по карте будем учитывать 3 параметра движения: скорость, направление поворота и угол поворота. Так как роботу необходимо пройти задание за минимальное время скорость будет максимальной и постоянной, поэтому на нее решение нейросети влиять не должно. Остается два параметра: угол и направление поворота – следовательно, у нейросети должно быть два выходных нейрона. В соответствии с их показаниями робот будет выбирать направление и угол поворота.

Награждение особей

В данном алгоритме правильная награда является одним из главных аспектов обучения. В ходе обучения роботов я испробовал различные способы награждения роботов:

1. За пройденное расстояние. Данный способ награждения привел к тому, что все роботы стали кружить вокруг одной точки.
2. За площадь, пройденную на карте. Данный подход дал больше результатов, роботы перестали стоять на месте и начали активнее проезжать по карте.
3. За площадь и найденные фигуры. Такой способ поощрения еще больше стал стимулировать роботов на исследование карты.
4. Поощрение за площадь и найденные фигуры и наказание за пройденное расстояние. Данный способ оказался самым эффективным, заставляя роботов находить все фигуры за короткое время.

Итоговая формула для получения награды выглядит так:

, где S – пройденная площадь, n – количество найденных фигур, d – пройденное расстояние.

В программном коде данная формула выглядит так:

return ((Counter(self.includes)[True])\*\*2)\*(len(self.shapes)\*\*2+1)/

math.sqrt(self.distance)/10

Итоги обучения передвижению

Обучив нейросеть с учетом вышеуказанных факторов, я получил роботов, способных находить все три фигуры на большей части карт и как минимум одну фигуру на остальных. Таким образом, применение нейросетей в решении конкурсных задач по робототехнике уже можно считать эффективным.

Обучение распознаванию гаража

Следующей подзадачей является нахождение гаража. Для ее решения роботу необходимо передвигаться по карте и проверять пространство перед собой на наличие гаража. На выполнение последнего действия нам и необходимо обучить нейросеть.

Для начала определим тип задания и нейросети, а также способ обучения. Задача робота – определить, находится ли перед ни гараж. Данную задачу можно отнести к распознаванию образов и классификации. Следовательно, для обучения необходимо использовать персептрон.

Для обучения персептрона применяется обучение с учителем. Одним из таких алгоритмов обучения является метод обратного распространения ошибки. Данный алгоритм корректирует каждый вес, пропорционально тому, насколько он способствует общей ошибки.

Написание нейросети

Для обучения я решил написать собственную модель персептрона, способную работать с любым количеством нейронов и обучатся методом обратного распространения ошибки. Созданный класс нейронной сети обладет следующими функциями:

#   Класс нейросети

*class* NeuralNetwork:

#   Инициализация нейронной сети, объявление переменных

*def* \_\_init\_\_(*self*, *neurons*=[1], *weights*=None):

#   Активационная функция

*def* f(*self*, *x*):

#   Производная активационной функции

*def* df(*self*, *x*):

#   Нормализация входных данных

*def* norm\_value(*self*, *inputs*):

#   Возвращение решения нейросети по полученным данным

*def* get\_response(*self*, *inputs*):

#   Обучение нейросети

*def* train(*self*, *epoch*, *duration*):

Подбор входных и выходных данных

Чтобы максимально эффективно обучить нейросеть необходимо правильно подобрать входные данные. В качестве информации нейронная сеть будет получать расстояние до препятствий перед машиной. Для повышения точности роботу необходимо остановится и сделать несколько замеров, поворачиваясь вокруг своей оси. Эффективность обучения будет определятся правильностью подбора количества замеров и угла поворота.

Для нахождения оптимального угла обзора и шага поворота необходимо учесть то, что на максимальной дальности работы датчика (255 см) расстояние между замерами не должно превышать ширину гаража (40см), а также количество замеров не должно быть мало. Учитывая все эти факторы, я нашел общий угол замера - 20°. Также экспериментальным путем был выявлен оптимальный шаг между замерами - 2°30´. Замеры датчиков можно обозначить следующим образом:

Так же входные данные должны включать ответ, ожидаемый от нейронной сети: 1 - перед роботом есть гараж и 0 - гаража нет. Таким образом входные данные будут иметь следующий вид:

Входные данные нейросети

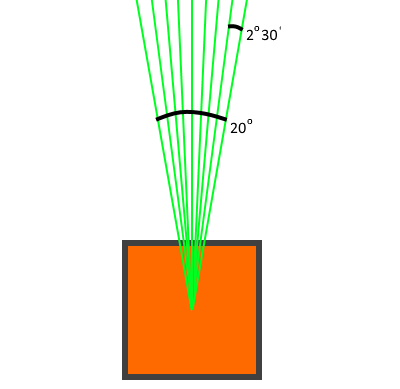
Результаты измерений робота     Ожидаемый ответ

             ↓                   ↓

[[254, 254, 254, 255, 186, 195, 204, 156], 0]

[[255, 255, 255, 255, 254, 255, 254, 254], 1]

[[170, 175, 181, 187, 254, 254, 254, 254], 1]



Выходные данные нейронной сети будет содержать всего одно число: 0 при отсутствии гаража и 1 при наличии.

Итоги обучения распознаванию гаража

По итогам обучения робот смог научится определять наличие гаража перед собой. Таким образом все задачи по обучению нейросетей выполнены

# Заключение

В ходе проделанной работы, мной были выполнены следующие задачи:

* Изучены особенности обучения нейросетей
* Созданы компьютерная модель задачи и робота
* Обучены две нейросети решающие подзадачи задания

Изучив тему нейронных сетей, написав и обучив их на выполнение поставленных задач, я пришел к выводу, что нейросети имеют практическое применение в школьной робототехнике и могут использоваться для решения конкурсных заданий.