***Подготовил:***

Скороходов А.И.

***Научный руководитель:***

Тулубенская Е.В.

2023г.

**Использование нейросетей для распознования пожара.**

Муниципальное Бюджетное Образовательное Учереждение

Средняя Образовательная Школа №17

Оглавление

[Введение 3](#_Toc130256249)

[Машинное обучение 4](#_Toc130256250)

[Принципы работы нейронной сети 5](#_Toc130256251)

[Моя модель для решения поставленной задачи 7](#_Toc130256252)

[Процесс обучения 9](#_Toc130256253)

[Результаты и заключение 9](#_Toc130256254)

[Список литературы 11](#_Toc130256255)

[Приложения 12](#_Toc130256256)

# Введение

Пожар – опасное явление, причиняющее вред имуществу и забирающее жизни. За 6 месяцев 2022 г. произошло 197 100 пожаров на которых погибло более 4000 человек. Своевременное выявление пожара является ключом к борьбе с ним. В этом помогают различные датчики. Современные противопожарные системы используют датчики, считывающие такие показатели как: температура воздуха, влажность, атмосферное давление, содержание углекислого газа, молекулярного водорода, этанола, TVOC(общее количество летучих органических соединений) и некоторые другие. Для своевременного реагирования на возгорание все эти данные обрабатываются с помощью компьютора. Одним из способов анализировать эти данные и определять начало возгорания является построение математических моделей и применение методов *машинного обучения*. Этот способ не дает гарантированного результата, однако шанс ошибки незначителен.

# Машинное обучение

**Машинное обучение** — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме[[1]](#b1). Существуют различные модели машинного обучения, среди которых в последнее время стали популярны нейронные сети. Их я и буду использовать.

**Нейро́нная сеть** — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Нейронные сети представляет собой [систему](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) соединённых и взаимодействующих между собой простых процессоров ([искусственных нейронов](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%D1%8B%D0%B9_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD" \o "Искусственный нейрон)). Впервые понятие нейронной сети было предложено в 1943г. У.Маккалоком и У.Питтсом[[2]](#b2). За последущие годы было разработано множество архитектур сетей и алгоритмов обучения. Одной из основных архитектур является сеть прямого распространения с полносвязными слоями. В такой сети нейроны разбиты на слои, внутри которых нейроны не связаны, и каждый нейрон каждого слоя(кроме выходного) связан с каждым нейроном следующего слоя. Выделяют входной слой(на который подаются входные данные), выходной слой(который выводит ответ) и скрытые(все остальные).

# Принципы работы нейронной сети

В памяти компьютора нейронная сеть представлена в виде:

* Списка векторов b. b[l] – вектор смещений(англ. bias) слоя l. b[l][i] – смещение нейрона i в слое l.
* Списка матриц w. w[l] – матрица весов слоя w. w[l][i][j] – вес ребра из j в i.
* Списка векторов act. act[l] – вектор активаций нейронов слоя l. act[l][i] – активация нейрона i в слое l.

Веса и смещения выбираются случайным образом[[Приложение 1]](#app1).

Общий алгоритм работы:

1. На вход подается вектор-столбец вещестенных чисел.
2. Для каждого слоя сети по реккуретному соотношению, заданному формулой под рисунком сверху, вычисляется вектор активаций нейронов.
3. На выходе получаем вектор вещественных чисел: активации последнего слоя.
4. Для оценки точности полученного ответа можно вычислить функцию ошибки(так же известной как функция стоимости) ***C***.(не обязательный шаг)
5. Вычисляется вектор частных производных функции ошибки по каждой из активаций последнего слоя.
6. С помощью вектора частных производных функции стоимости по активациям данного слоя(который мы получили на предыдущем шаге) вычисляются: вектор производных функции стоимости по смещениям данного слоя, матрица производных функции стоимости по весам данного слоя и вектор производных функции стоимости по активациям предыдущего(по нумерации) слоя.
7. Шаг 6 повторяется пока не будут вычеслены производные по всем весам и смещениям в сети(Это называется методом обратного распространения).
8. Веса и смещения обновляются пропорционально их производным(коэффициент пропорциональности называется скоростью обучения).

Реализация на языке Python 3.10: [[Приложение 2]](#app2).

Перед тем, кто обучает нейросеть, стоят следующие задачи:

* Подготовить входные данные(собрать и отформатировать)
* Подобрать скорость обучения
* Выбрать архитектуру(количество слоев и нейронов в каждом из слоев)
* Выбрать функцию активации(f\_act в формуле на рисунке) и функцию стоимости(**C** на рисунке)
* Выбрать количество эпох(количество запусков алгоритма на наборе данных) обучения

Конечной целью является минимизация функции стоимости.

## Моя модель для решения поставленной задачи

В качестве функции активации я выбрал логистическую функцию из семейства сигмоид:

Эта функция обладает очень полезным свойством: она монотонно возрастает от 0 до 1 на промежутке . Таким образом на выходе мы получаем степень “уверенности” нейросети в том что начался пожар: чем ближе к 1, тем вероятней что пожар начался(для получения ответа необходимо выбрать ***порог***, начиная с которого мы будем считать, что пожар начался. Я выбрал порог 0.5). Также полезным свойством является то, что производная данной функции выражается через значение самой функции:

Это позволит упростить вычисления и уменьшить время обучения.

В качестве функции стоимости я использую бинарную энтропию:

Где *y* – это правильный ответ(), и *a* – результат полученный нейросетью (активация последнего слоя)(). Эту функцию можно использовать в качестве функции стоимости по следующим причинам:

* Она положительна на всей области определения.
* Чем больше отличаются y и a, тем больше значение функции(Потому что ).

Преимущество этой функции перед квадратичной() в том что производная функции бинарной энтропии по аргументу функции активации(которую мы будем считать) вычисляется намного быстрее:

Реализация этих функций на языке Python 3.10: [[Приложение 3]](#app3)

В процессе решения задачи лучше всего показала себя модель с архитектурой [12, 100, 200, 1] и скоростью обучения 1e-06. Количество эпох не было подсчитано.

# Процесс обучения

Для начала необходимо подготовить данные. Я взял набор данных [[4]](#b4).

Сначала я изменил значения таким образом, что бы они были не слишком большими(на самом деле этот шаг не так важен и нужен лишь для того, чтобы не возникало переполнение при вычислении экспоненты). Затем взял случайным образом разделил набор данных в отношении 1 к 5. Большая часть – это обучающий набор, на котором модель будет обучатся. А меньшая – тестовый, на нем я буду проверять результаты работы программы и точность(отношение правильных ответов ко всем ответам или, по-другому, шанс получить правильный ответ[[Приложение 4]](#app4).

Далее я начал обучать сеть, запуская функцию fit()[[Приложение 5]](#app5). В процессе обучение я строил график функции стоимости по её значениям в конце каждой эпохи, которые я вычислял с помощью функции iterate(). Переодически я проверял результаты с помощью функции count(). Когда результат, по моему мнению, стал приемлемым, я завершил обучение.

# Результаты и заключение

Перебрав несколько моделей я смог добиться точности в 94%[[Приложение 6]](#app6). То есть с шансом 94% программа правильно определит, начался пожар или нет.

Улучшить этот показатель можно собрав больше данных или построив модель с большим количеством нейронов(однако в этом случае время обучения и занимаемая программой память значительно возрастают и вычислительной мощности моего компьютора уже не хватает, чтобы выполнить достаточное количество эпох за приемлемое время). Можно обрабатывать несколько входных данных параллельно(что может как улучшить результат, так и ухудшить), в таком случае, при многопоточной реализации, время выполнения одной эпохи сократится, пропорционально размеру этого мини-пакета(англ. mini-batch) входных данных. Изменение весов и смещений тогда будет равно среднему арифметическому изменений весов и смещений в каждом случае. Так же ускорить обучение можно, перенеся вычисления на GPU (только для массивов с более 1e8 элементов), но в данном случае это было не эффективно в силу малого размера сети и особенностей моего компьютора.

Изменяя [порог](#tr) округления, можно изменять соотношение ложно положительных и ложно отрицательных результатов(что, однако, не должно сильно изменить точность).

Обученную модель можно использовать для распознования пожара, используя те же показатели, что использовались при её обучении.

# Список литературы

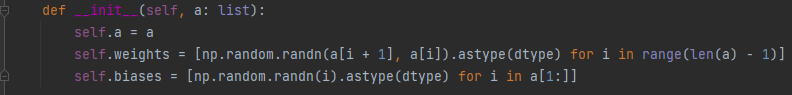
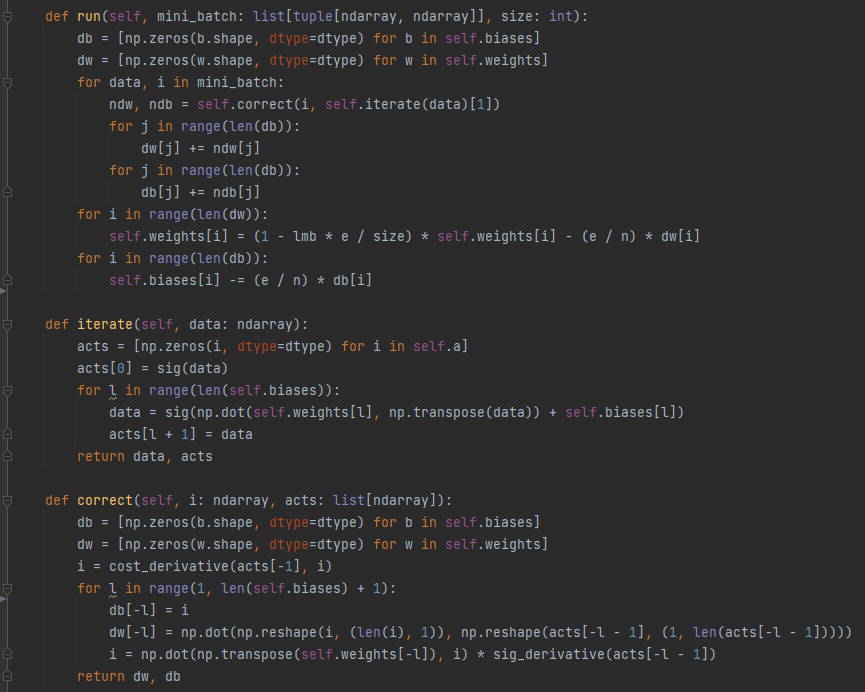
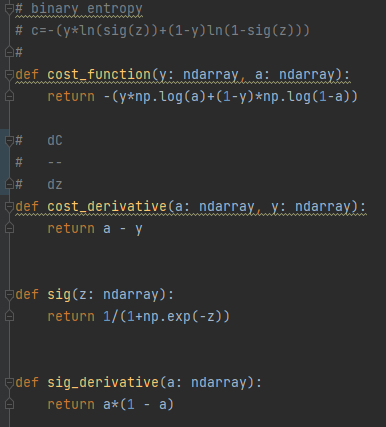
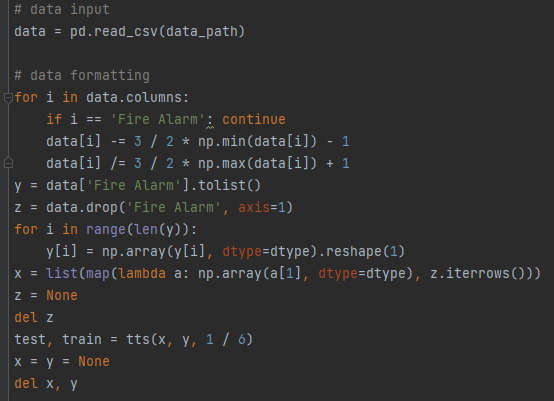
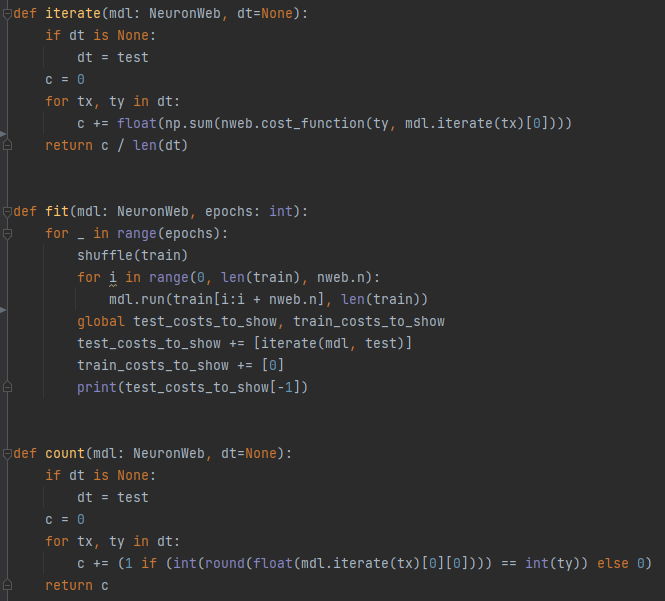
1. <https://ru.wikipedia.org/wiki/Машинное_обучение>

1. <https://ru.wikipedia.org/wiki/Нейронная_сеть>

1. <https://habr.com/ru/post/456738> (перевод книги <http://neuralnetworksanddeeplearning.com/about.html> )

1. <https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset>

# Приложения

1. 
2. 
3. 
4. 
5. 
6. 