**Лабораторная работа №2**

НЕЙРОННЫЕ СЕТИ

По дисциплине «Машинное обучение»

Выполнил студент

группы 3530904/80102: Шерман М.Л.

Преподаватель: Селин И.А.

Оглавление

[Задачи 3](#_Toc69497213)

[Пункт 1 4](#_Toc69497214)

[Пункт 2 7](#_Toc69497215)

[Пункт 3 8](#_Toc69497216)

[Вывод 11](#_Toc69497217)

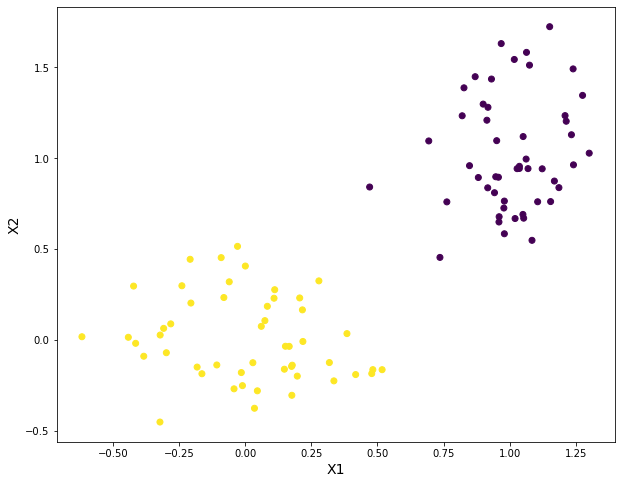
[Приложение 12](#_Toc69497218)

# Задачи

1. Постройте нейронную сеть из одного нейрона и обучите её на датасетах nn\_0.csv и nn\_1.csv. Насколько отличается результат обучения и почему? Сколько потребовалось эпох для обучения? Попробуйте различные функции активации и оптимизаторы.
2. Модифицируйте нейронную сеть из пункта 1, чтобы достичь минимальной ошибки на датасете nn\_1.csv. Почему были выбраны именно такие гиперпараметы?
3. Создайте классификатор на базе нейронной сети для набора данных [MNIST](http://yann.lecun.com/exdb/mnist/) (так же можно загрузить с помощью torchvision.datasets.MNIST, tensorflow.keras.datasets.mnist.load\_data и пр.). Оцените качество классификации.

# Пункт 1

Основной темой данной работы являются нейронные сети и их возможности. Для более детального погружения в материал было предложено начать рассмотрение возможностей нейронной сети, состоящей из одного нейрона. Для исследования были даны два датасета. Каждый объект датасетов имеет по два признака и относится к одному из двух классов. Для построения нейронных сетей воспользуемся библиотекой PyTorch для языка программирования Python.

Рассмотрим «расположение» классов на плоскости для первого набора данных:

Видно, что оба класса хорошо обособлены друг от друга (линейно разделимы). Модели классического машинного обучения (например KNN, SVC) с правильно подобранными параметрами смогли бы без труда разделить данную выборку. Посмотрим, сможет ли с данной задачей справиться нейронная сеть из одного нейрона.

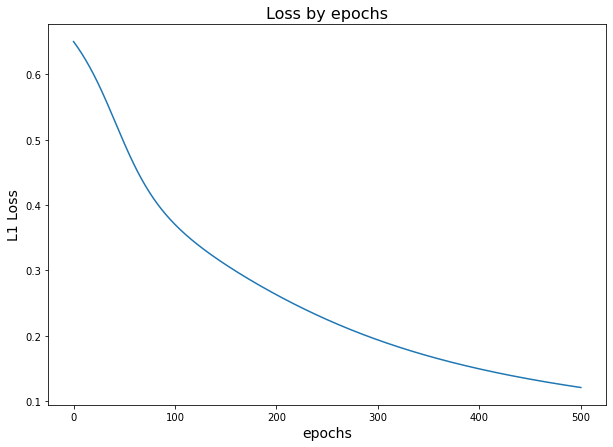
Наша нейронная сеть будет состоять из одного линейного слоя и функции активации. Функция активации необходима для нормализации выходных данных.

Первым делом после создания нейронной сети необходимо выбрать функцию ошибок, которая будет подсчитывать ошибку на каждой итерации, а также функцию оптимизации, которая будет изменять веса модели в соответствии с полученными ошибками. Для различных оптимизаторов понадобятся различные значения параметра скорости обучения (learning rate), поэтому в зависимости от выбранной функции мы будем выбирать наилучшее значение этого параметра. Также важным параметром является количество эпох для обучения.

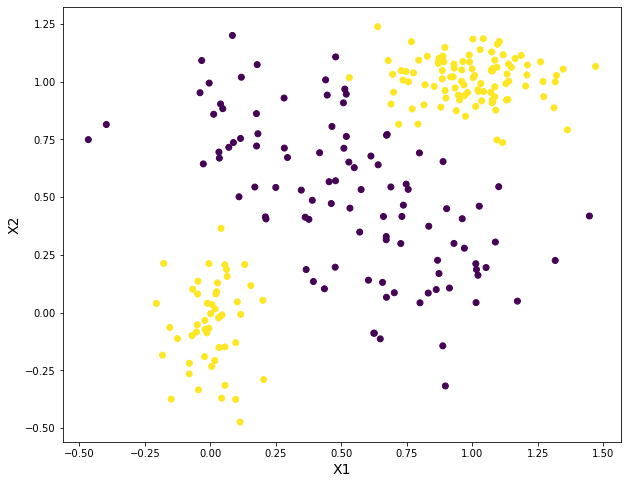
В первом пункте работы будем использовать функции потерь, ориентированные на регрессию, так как они наиболее понятны и просты при начале работы с нейронными сетями.

После нескольких тестов удалось определить наилучший набор функций и признаков для нейронной сети:

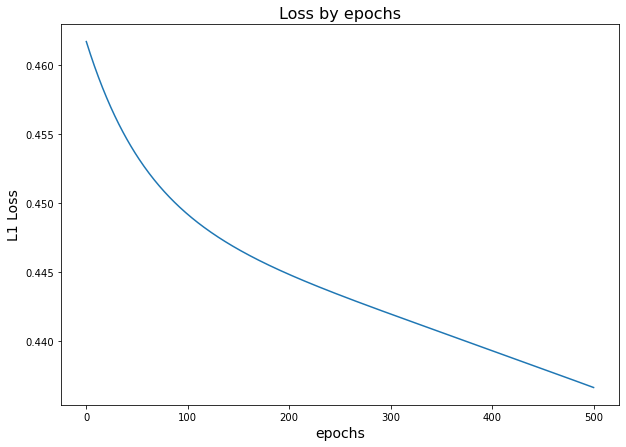
* Функция потерь - L1Loss,
* Оптимизатор – SGD
* Скорость обучения - 0.1
* Количество эпох - 500

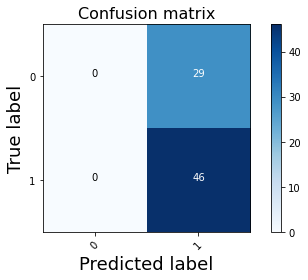
Где L1Loss – функция ошибок на основе метода средней абсолютной ошибки, SGD – метод стохастического градиентного спуска.

Точность предсказания составила 100%. Этот результат безусловно радует, стоит помнить, что мы рассматривали линейно разделимую выборку, на которой хорошие результаты могут показать и более простые алгоритмы.

Рассмотрим «расположение» классов на плоскости для второго набора данных:

В данной выборке представленные классы линейно неразделимы, что позволит более детально оценить возможности нейронных сетей.

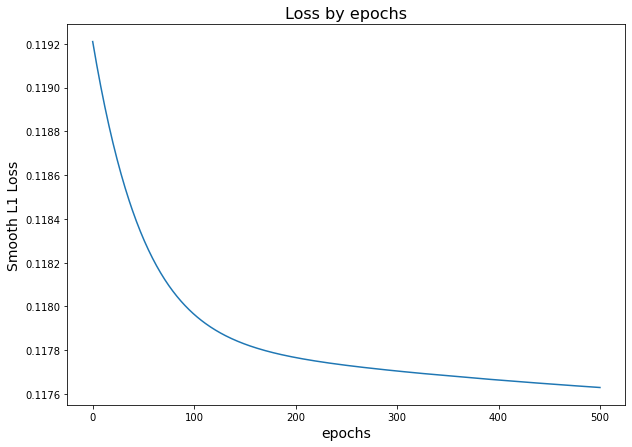
Воспользуемся той же моделью, что и для предыдущей выборки, совместно с теми же функциями потерь и оптимизации.

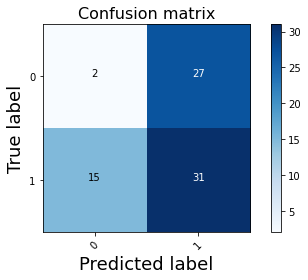
Точность составила 61%. С одной стороны, может показаться неплохим результатом для линейно неразделимой выборки, но стоит взглянуть на матрицу ошибок:

Нейронная сеть определила все объекты, как экземпляры первого класса. Такой результат никак нельзя назвать успешным. Основной причиной такого поведения модели стали функция потерь и оптимизатор. Безусловно, само устройство нейронной сети тоже влияет на точность, но в связи ограничениями задания, попробуем улучшить результаты другими способами.

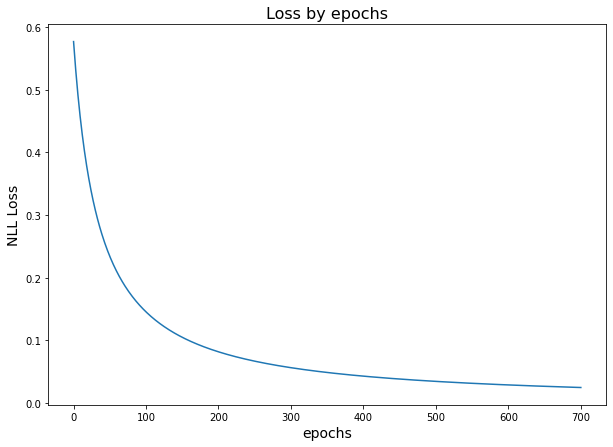
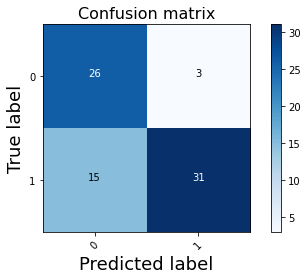
# Пункт 2

Попробуем использовать иную функцию потерь: вместо L1Loss воспользуемся SmoothL1Loss. При большом абсолютном значении аргумента данная функция ведёт себя как L1Loss, а при близком к 0 как MSELoss (функция ошибок на основе метода средней квадратичной ошибки).

В результате получаем следующие данные от функции потерь:

Точность составила 44%, однако теперь нейронная сеть хотя бы определяет некоторые классы как 0.

Воспользуемся функцией ошибки NLLLoss, которая специально используется для классификации. Также для корректной работы модели с выбранной функцией потерь заменим функцию активации nn.Sigmoid на nn.LogSigmoid.

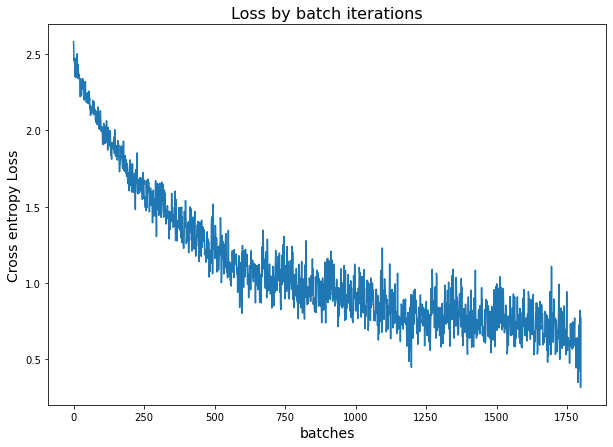
Точность составила 76%, что можно считать успехом. Матрица ошибок выглядит гораздо лучше, чем в предыдущих случаях:

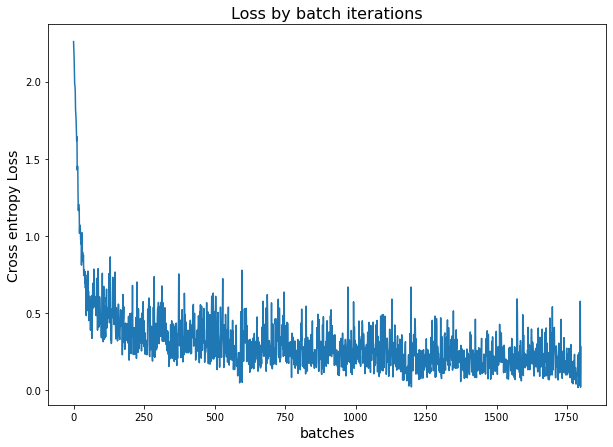
Для дальнейшего улучшения точности предсказания потребуется более кардинальное улучшение структуры нейронной сети.

# Пункт 3

В данном пункте необходимо создать классификатор на основе нейронной сети для набора данных MNIST. Этот набор содержит чёрно-белые фотографии цифр от 0 до 9 размера 28 \* 28 пикселей. Для многоклассовой классификации будем использовать функцию потерь CrossEntropyLoss. Данная функция объединяет в себе функцию активации LogSoftmax и функцию потерь NLLLoss. Оптимизатор оставим прежним – SGD.

Поскольку в данном датасете достаточно много данных (60000 тренировочных значений и 10000 тестовых), изменим подход к обучению модели. Раньше из-за небольшого объёма данных мы обучали модель на всём датасете и для приемлемой точности задавали большое количество эпох. Теперь на каждой эпохе мы будем разделять данные на пакеты (батчи) размером 100 элементов. Благодаря такому подходу модель гораздо лучше обучается.

Для более тщательного обучения нейронной сети установим скорость обучения 0.001.

Точность полученной модели составила 85%. Результат впечатляющий, учитывая, что нейронная сеть имеет лишь один скрытый слой. Попробуем повысить точность добавив ещё один слой, а между ними применить функцию активации для нелинейного преобразования. Также теперь будем использовать в качестве оптимизатора функцию Adam, которая чаще всего используется в работе над данным датасетом.

В результате полученная точность составила 94%.

# Вывод

В результате проделанной работы были исследованы базовые принципы работы и построения нейронных сетей. Удалось изучить различные алгоритмы оптимизации, а также функции подсчёта ошибок.

Таким образом, был получен опыт построения и применения нейронных сетей для базовых наборов данных.

# Приложение

Весь код и графики можно найти в следующем репозитории:

<https://github.com/Mark-Sherman-SE/ML-Labs>