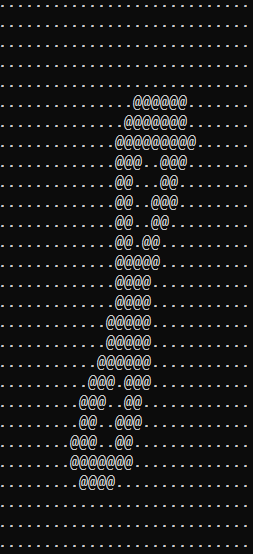
*Отчёт к индивидуальному проекту по теме «Искусственный интеллект»*

*Подготовил студент 17.Б02-пу группы*

*Пченикин Никита*

***Содержание***

1. Описание задачи
2. Архитектура нейронной сети
3. Обучение
4. Решение поставленной задачи, точность решения и ошибки
5. Вывод
6. Используемые источники
7. **Описание задачи**

Построим сеть, распознающую рукописные цифры. Для этого будем использовать готовый набор данных MNIST, включающий 60000 обучающих и 10000 тестовых примеров. Все изображения в MNIST полутоновые и имеют размерность 28x28. Тестовые примеры не используются для обучения и созданы для того, что проверить, насколько хорошо обучилась наше нейронная сеть.

1. **Архитектура нейронной сети**

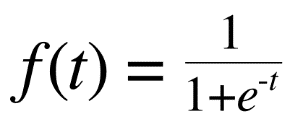
Для реализации нейронной сети был использован однослойный персептрон с 784 входами и одним промежуточным слоем, состоящий из 10 нейронов, которые в последствии нормируются в единичные вектора для решения задачи классификации. Т.е выходной слой имеет 10-мерный вектор ei, где на i-ой позиции стоит 1.0 и 0.0 иначе.

Сама нейронная сеть выглядит следующим образом:

Где массив **Wij** – отвечает за вес между i-ым нейроном входного слоя и j-ым нейроном промежуточного.

**respi** – результат функции активации для i-ого нейрона промежуточного слоя.

Для того, чтобы обучить нейронную сеть, необходимо задать соответствующие веса на ребрах идущие в промежуточный слой. Изначально эти веса задаются произвольными числами от 0.0 до 0.1. В качестве применяемого метода обучения был выбран алгоритм обратного распространения ошибки. Функцией активации выбрана сигмоида:



На каждой итерации обучения, весовые коэффициенты нейронов подгоняются с использованием новых данных из тренировочных примеров.

Изменение весовых коэффициентов и составляют суть алгоритма обратного распространения ошибки.

Каждый шаг обучения начинается с воздействия входных сигналов из тренировочных примеров. После этого мы можем определить значения выходных сигналов для всех нейронов. Этот шаг называется прямым распространением, в ходе которого мы используем взвешенную сумма для нейронов промежуточного слоя.

На следующем шаге алгоритма, выходной сигнала сети **respi** сравнивается с желаемым выходным сигналом **yi**, который хранится в тренировочных данных.

Разница между этими двумя сигналами называется ошибкой **errori** выходного слоя сети.

Невозможно непосредственно вычислить сигнал ошибки для внутренних нейронов, потому что выходные значения этих нейронов, неизвестны.

На протяжении многих лет был неизвестен эффективный метод для обучения многослойной сети.

Только в середине восьмидесятых годов был разработан алгоритм обратного распространения ошибки.

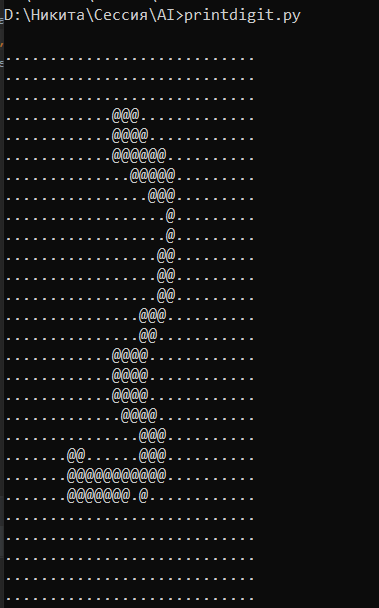
Идея заключается в распространении сигнала ошибки **errori** (вычисленного в шаге обучения) обратно на все нейроны, чьи выходные сигналы были входящими для последних нейронов.

Ниже изображен шаг обратного распространения:

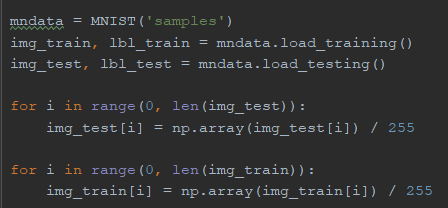


1. **Обучение**

В качестве данных будем использовать известную в мире распознания чисел подборку данных MNIST(<http://yann.lecun.com/exdb/mnist/>). Обучающая выборка состоит из 60000 примеров, тестирующая из 10000.

Для работы с архивом установим утилиту python-mnist, поскольку данные в архиве представляются в специфичном виде.

После того, как архивы с данными были загружены, программе необходимо указать путь к каталогу с ними. Так, переменные img\_train и img\_test содержат изображения для тренировки сети и тестирования соответственно. А переменные lbl\_train и lbl\_test- метки с правильной классификацией (т.е. цифры с изображений). Все изображения имеют размер 28х28.

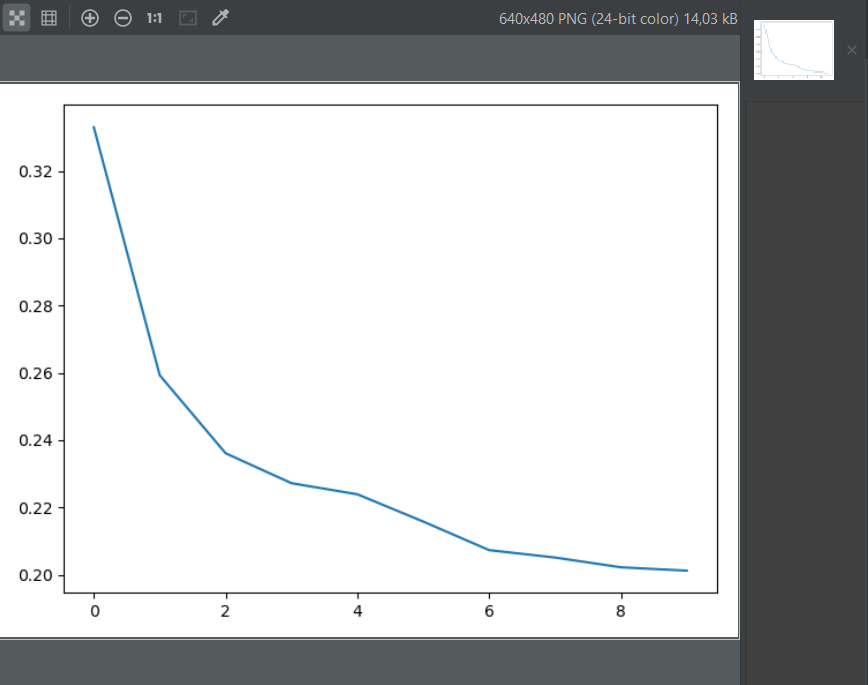
Выполним предобработку данных. Несмотря на то, что с изображениями принято работать в виде двумерного массива, представим изображения в виде одномерного массива и нормализуем данные, чтобы повысить точность вычислений

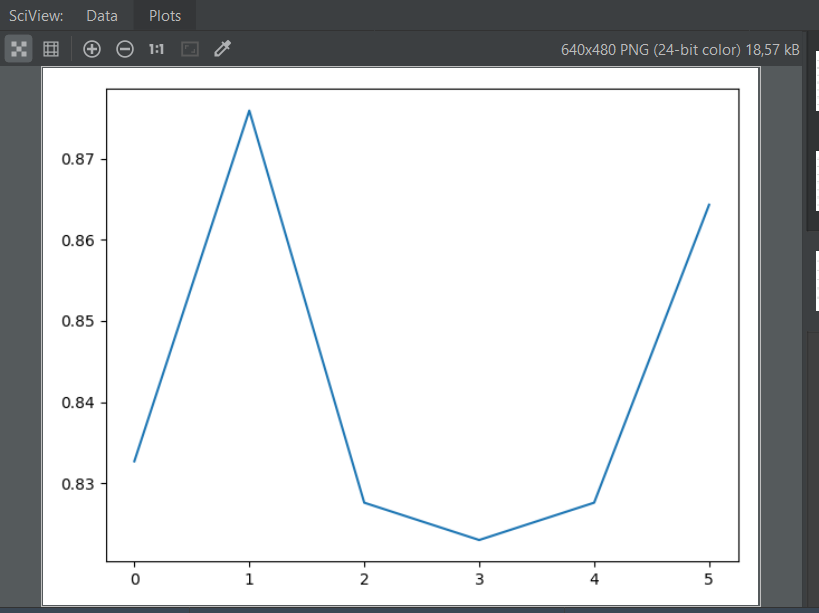
1. **Решение поставленной задачи, точность решения и ошибки**

Нейросеть обучалась 10 эпох при среднем времени на эпоху около 5 минут. Алгоритмическая сложность алгоритма , т.к взвешенная сумма считается для каждого нейрона промежуточного слоя, суммируя соответствующие веса входного слоя, где m – кол-во нейронов на промежуточном слое, в данном случае m = 10.Точность «предсказаний» варьируется в районе 88.4% к общему число примеров на тестовом множестве

Для выявления количественной характеристики, показывающий число несовпадения на всей выборке используется 0-1loss функция:

Где X – тестовое множество, операция применяется ко всему вектору их выходного слоя.

Была проанализирована функция ошибки на всех 10 эпохах, по которой видно, что она сходится к асимптоте

Для проверки нейросети на переобучение была сделана «кросс-проверка», она же «перекрёстная проверка», суть которой заключается в разбиении имеющихся данных на k частей, k – 1 разбиений используются для обучения, и оставшаяся часть используется для тестирования. Процедура выполнятся k раз. В нашем случае k = 6, сеть тестировалась 6 раз на примерах с k \* 10000 по (k + 1) \* 10000 – 1 и обучалась на остальных. Приведенный график демонстрирует точность предсказаний в каждом из 6 случаев. Из графика можно заключить, что точность в среднем при данной выборке варьируется в пределах 84.1%.

1. **Вывод**

При количестве нейронов в скрытом слое, равном 10, достигнута точность 85% при однократном выполнении алгоритма обратного распространения ошибки и 88% при выполнении на 10 эпохах. Была сделана «перекрёстная проверка», показывающая что переобучение нашей модели не возникло и в среднем точность на k = 6 тестах варьируется в пределах 84.1%. При увеличении кол-ва нейронов увеличится предполагаемая точность, но и потребуется большие вычислительные мощности. Но уже можно сделать вывод, что такая простая в реализации нейросеть способна различать цифры намного лучше встроенного в Microsoft Word механизма распознавания символов при рукописном вводе формул.

1. **Используемые источники**

[1] <http://robocraft.ru/blog/algorithm/560.html>

[2] <http://py-algorithm.blogspot.com/2014/10/blog-post_8.html>

[3] <http://qaru.site/questions/1259290/extract-images-from-idx3-ubyte-file-or-gzip-via-python>

[4] <https://ru.coursera.org/lecture/supervised-learning/mnoghosloinaia-nieironnaia-siet-funktsiia-oshibki-5kFhz>