Санкт-Петербургский государственный университет

Распознавание образов с помощью Pytorch

Выполнил:
студент 221 группы
Коротченок Остап Андреевич

Научный руководитель:
(Профессор, Ph.D.) М. В. Юлдашев

Отметка о зачете:

Санкт-Петербург 2022

Содержание

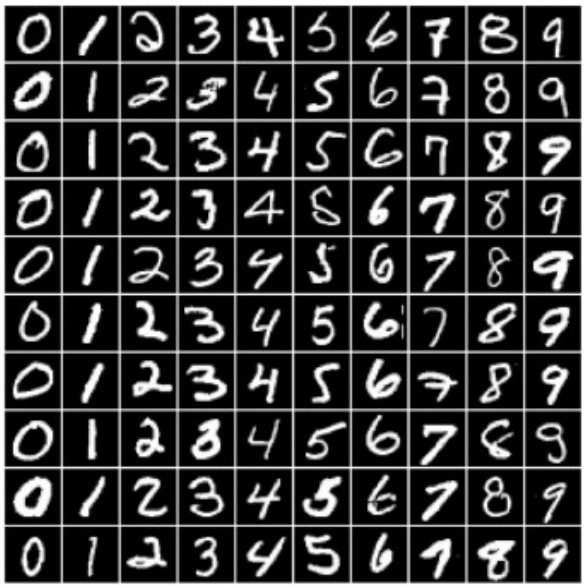
1 Введение

Задачей данной работы является реализация нейронной сети для распознавания рукописных цифр. В качестве тренировочного датасета была выбрана известная база данных MNIST Handwritten Digits Database. В работе рассмотрена модель на основе логистической регрессии и двухслойная модель FNN, последняя из которых была реализована и результаты работы которой приведены далее в отчете.

2 Основная часть

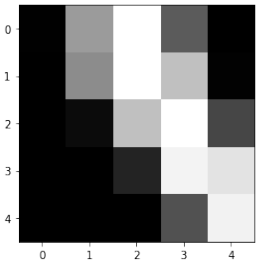
2.1 MNIST Handwritten Digits Database

База данных содержит 60000 grayscale изображений 28px × 28px рукописных цифр (от 0 до 9) с пометками, какая цифра изображена.



Пример изображения из датасета

Библиотека torchvision позволяет конвертировать изображения в тензоры, таким образом любое изображение из серых тонов мы можем преобразовать в матрицу:



$$\downarrow$$

$$\begin{pmatrix} 0.0039 & 0.6039 & 0.9922 & 0.3529 & 0.0000 \\ 0.0000 & 0.5451 & 0.9922 & 0.7451 & 0.0078 \\ 0.0000 & 0.0431 & 0.7451 & 0.9922 & 0.2745 \\ 0.0000 & 0.0000 & 0.1373 & 0.9451 & 0.8824 \\ 0.0000 & 0.0000 & 0.0000 & 0.3176 & 0.9412 \end{pmatrix}$$

Здесь 0 - черный, 1 - белый, а числа между 0 и 1 - оттенки серого.

2.2 Модель

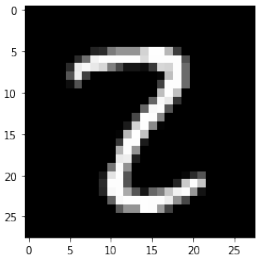
2.2.1 Logistic Regression

Логистическая регрессия классически применяется для предсказания вероятности возникновения события по значениям множества признаков. В нашем случае, по матрице 28×28 с элементами $\in [0, 1]$ будем получать предсказание - число от 0 до 9. Если точнее, то для входной матрицы X будем получать вектор размера 10 (соответствие с цифрами 0-9) по формуле

$$\text{Prediction} = X \times W^T + B \tag{1}$$

где W - матрица весов, B - матрица смещения. К полученному вектору применяется SOFTMAX (см. 2.3.1) и за ответ берется максимум из полученных вероятностей. В качестве функции качества (loss function) используется перекрестная энтропия (Cross Entropy)(см. 2.3.3), в качестве оптимизатора - градиентный спуск.

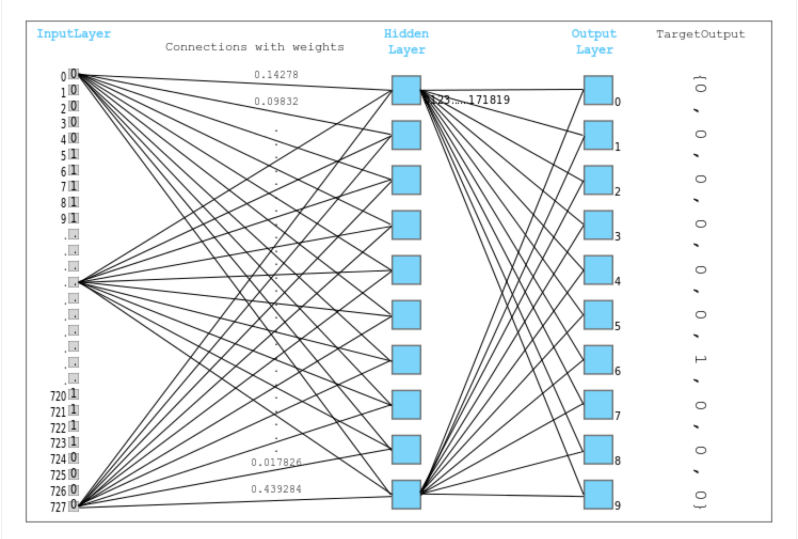
Если на практике применить такую модель, то ее точность не сможет преодолеть 85%. Причиной этому является линейность модели, что приводит к ошибкам как приведено ниже. Поэтому в данной работе представлена реализация более сложной модели, которая будет описана в пункте 2.2.2.
Пример ошибки линейной регресси:



Label: 2 , Predicted: 8

2.2.2 Feedforward Neural Network

Feedforward Neural Network (FNN) или нейронная сеть с прямой связью - самый простой тип нейронной сети, в ней соединения между узлами не образуют циклы.
В нашем случае входными данными будут такие же изображения 28×28, а выходными - вектора размера 10. Однако теперь модель будет состоять из двух слоев, которые будут линейными (действующими по принципу уравнения (1)), переход между которыми будет осуществлять функция активации. Это привносит нелинейность в нашу модель, что поможет преодолеть порог в 85% точности и добиться лучших результатов.
Рассмотрим алгоритм подробнее. Сначала введем обозначение b (batch) - размер партии, т.е. сколько изображений мы берем из тренировочного сета. Тогда первый слой переведет входную матрицу размера $b \times 784$ ($28 \times 28 = 784$) в промежуточную матрицу размера $b \times h$, где за h обозначим глобальный параметр (реализовано при $h = 32$) (h - hidden layer). Далее к каждому элементу промежуточной матрицы применим функцию активации (activation function, реализовано ReLU) (см. 2.3.2). Второй слой преоразует полученную матрицу (все еще $b \times h$) в матрицу $b \times 10$, по которой уже считается потеря (Cross Entropy - см. 2.3.3) и далее веса оптимизируются градиентным спуском.



Визуализация связей между слоями.

2.3 Методы

2.3.1 SOFTMAX

SOFTMAX — это обобщение логистической функции для многомерного случая. Функция преобразует вектор p размерности n в вектор σ той же размерности, где каждая координата $\sigma_i \in [0, 1]$ и $\sum_{i=1}^{n} \sigma_i = 1$.
Координаты вычисляются по формуле:

$$\sigma(p)_i = \frac{e^{p_i}}{\sum_{j=1}^{n} e^{p_j}}$$

Таким образом, SOFTMAX преобразует любой вектор в вектор вероятностей.

2.3.2 ReLU

Rectified Linear Unit — это наиболее часто используемая функция активации при глубоком обучении. Она возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число.
Более формально (с использованием обозначений из предыдущего пункта)

$$\sigma(p)_i = \max(0, p_i)$$

Именно ReLU привносит в модель 2.2.2 нелинейность.

2.3.3 Cross Entropy

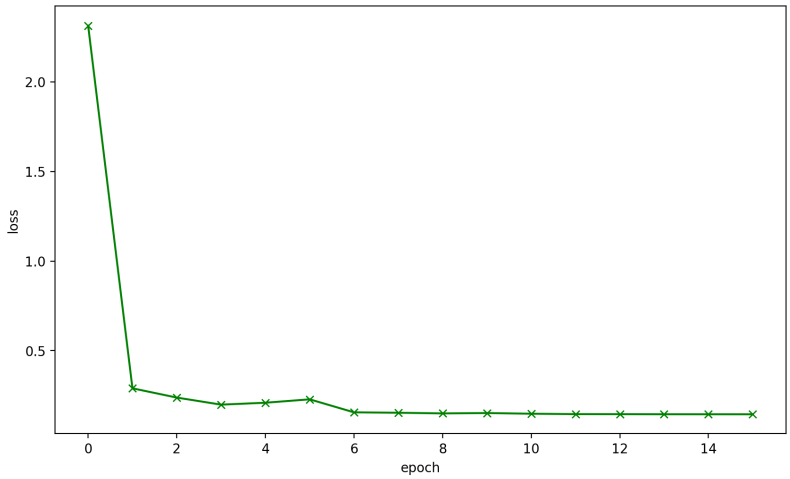
Cross Entropy или Перекрестная Энтропия между двумя распределениями вероятностей измеряет среднее число бит, необходимых для опознания события из набора возможностей, если используемая схема кодирования базируется на заданном распределении вероятностей q, вместо «истинного» распределения p.
В дискретном случае это задается формулой

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

В нашем случае мы сравниваем вектор вероятностей $\hat{y}$ (полученный применением SOFTMAX к результату умножения и сложения матриц в случае логистической регрессии) с вектором $y = (y_0, \ldots, y_9)$, где $\exists! i_{\in\{0,\ldots,9\}} \mid y_i = 1, y_j = 0, i \neq j$. Очевидно, если 1 стоит на месте i в y, то результатом применения функции будет $-\log \hat{y}_i$, так как остальные слагаемые суммы равны нулю. За функцию качества Cross Entropy можно брать, т.к. она дифференцируема, что требуется для градиентного спуска, где мы бы минимизировали ее среднее значение по всем результатам на выборке.

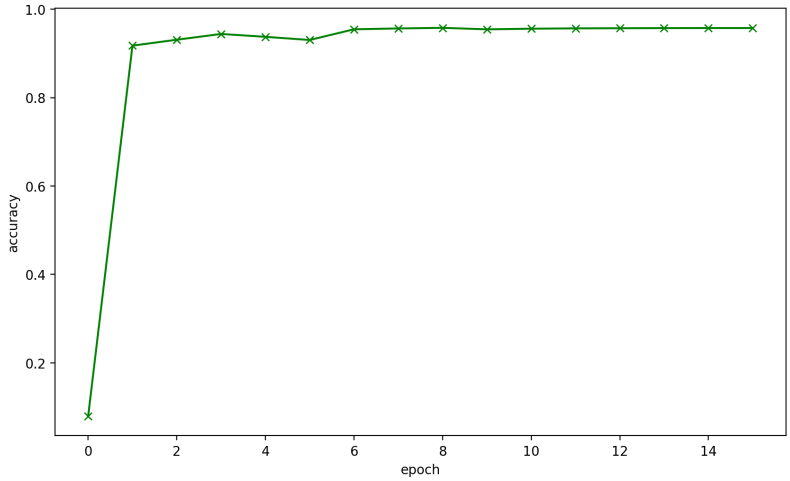
3 Результаты

Реализована двухслойная нейронная сеть, которая обучалась в 15 эпох: 5 со скоростью обучения 0.5, 5 с 0.1 и 5 с 0.01.
В итоге на последней валидации потеря составила 0.1451, а точность - 0.9575, при начальных 2.3132 и 0.0788



Изменение потери по эпохам.

Функцию качества мы на протяжении эпох минимизируем градиентным спуском.



Изменение точности по эпохам.

Точность считается как процент верно классифицированных изображений от всех элементов пакета из валидационного сета.
На тестовом датасете потеря составила 0.1244, а точность - 0.9623, что (как и требуется) близко к результатам на валидационном датасете.

4 Источники

1. Курс "Машинное обучение" на Coursera
2. PyTorch for Deep Learning - Full Course / Tutorial