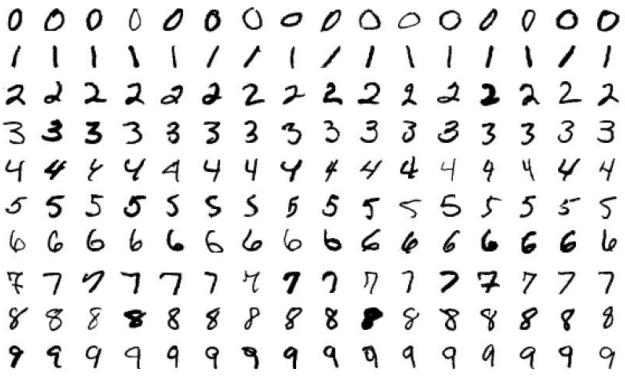
Задание №2.4. Нейронные сети

В этом задании вы создадите простую нейронную сеть, которая будет распознавать чёрно-белые картинки рукописных цифр из датасета MNIST, который содержит 60000 учебных и 10000 тестовых примеров рукописных цифр от 0 до 9. Каждая картинка представляет собой матрицу пикселей размером 28×28, которая хранится в файле в виде одномерного вектора из 784 чисел. Выборка также содержит метки для каждого образца. Вот пример некоторых картинок:



Шаблон кода и сама выборка содержатся в файлах: `nn.py`, `images_train.csv`, `labels_train.csv`, `images_test.csv`, `labels_test.csv`.

Имеющийся в `nn.py` код разбивает учебную выборку на две части – непосредственно учебную из 50000 примеров, и валидационную из 10000 примеров, которая будет использоваться для оценки качества обучения.

Вам необходимо реализовать нейронную сеть с одним скрытым слоем и перекрестной энтропией в качестве функции потерь и обучить ее на входных данных. Используйте сигмоиду в качестве активации нейронов скрытого слоя и функцию софтмакса для выходного слоя. Напомним, что софтмакс – это функция, являющаяся обобщением логистической функции на случай n переменных, которая делает из них распределение[1]:

$$softmax(a_i) = \frac{e^{a_i}}{\sum_{i=1}^{n} e^{a_i}}.$$

Для одного единственного примера (x, y) функция перекрестной энтропии выглядит следующим образом:

$$CE(y, \hat{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k,$$

где $\hat{y} \in \mathbb{R}^K$ – вектор значений функции софтмакса выходного слоя для одного обучающего примера x, а $y \in \mathbb{R}^K$ – истинное значение, которое должна выдавать нейронная сеть для x. В векторе $y = [0, \dots, 0, 1, 0, \dots, 0]^T$ единственная единица стоит на позиции корректного класса (метки). Это так называемое one-hot кодирование классов. Напоминаем, что метка в обучающей выборке – это одно единственное число от 0 до 9, в то время как нейронная сеть будет выдавать распределение $\hat{y} = [\hat{y}_0, \hat{y}_1, \dots, \hat{y}_9]$, где каждое $\hat{y}_i$ соответствует вероятности того, что образец x принадлежит i-му классу.

---

[1] Вы можете заметить, что логистическая функция $g(z) = \frac{1}{1+e^{-z}}$ делает распределение из чисел z и $-z$. Также напомним, что под распределением (вероятности) мы понимаем набор чисел, каждое из которых находится в диапазоне от 0 до 1, и в сумме дающих единицу.

Для n обучающих примеров мы берем среднее значение перекрестной энтропии:

$$J\left(W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}\right) = \frac{1}{n}\sum_{i=1}^{n} CE\left(y^{(i)}, \hat{y}^{(i)}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}.$$

Стартовый код в шаблоне уже конвертирует метки в one-hot формат.

Вместо пакетного или стохастического градиентного спуска часто пользуются мини-пакетным градиентным спуском, при котором обучение происходит на подвыборке размера B. В этом случае функция стоимости будет выглядеть следующим образом:

$$J_{MB} = \frac{1}{B}\sum_{i=1}^{B} CE\left(y^{(i)}, \hat{y}^{(i)}\right),$$

где B – это размер пакета, то есть количество обучающих примеров в мини-пакете.

Вопрос №1 [6 баллов]

Реализуйте прямое распространение и обратное распространение для вышеприведенной функции потерь. Инициализируйте веса сети случайными значениями, распределенными по стандартному нормальному закону. Инициализируйте веса смещений нулевыми значениями. Количество нейронов на скрытом слое должно быть равно 300, а скорость обучения $\alpha = 5$. Пусть $B = 1000$. Это означает, что на каждой итерации обучение происходит на подвыборке из 1000 примеров. Соответственно, каждая эпоха обучения состоит из 30 итераций, что позволяет покрыть всю исходную обучающую выборку. Картинки уже перемешаны, поэтому вы можете набирать мини-пакеты последовательно.

Обучите модель с помощью мини-пакетного градиентного спуска, описанного выше. Обучение должно продолжаться 30 эпох. В конце каждой эпохи подсчитайте значение функции потерь, усредненное по **всей** обучающей выборке, а затем отобразите эти значения на графике, в котором по оси y идут значения функции, а по оси x – количество прошедших эпох. На этом же графике отобразите значения функции потерь на валидационной выборке.

Аналогичным образом на другом графике выведите точность обучения на обучающей и валидационной выборках, измеряемую как доля корректно классифицированных примеров. По оси x также откладывается количество прошедших эпох. Приложите оба графика к отчету.

Также в конце всего обучения сохраните веса обученной сети в файл, чтобы в дальнейшем их можно было загрузить и использовать в вопросе №3.

Совет: постарайтесь векторизовать свой код максимально возможным образом. В противном случае обучение может стать очень долгим.

Вопрос №2 [2 балла]

Теперь добавим регуляризацию к нашей функции перекрестной энтропии. В результате она приобретет следующий вид:

$$J_{MB} = \left(\frac{1}{B}\sum_{i=1}^{B} CE\left(y^{(i)}, \hat{y}^{(i)}\right)\right) + \lambda\left(\left\|W^{[1]}\right\|^2 + \left\|W^{[2]}\right\|^2\right).$$

Помните, что мы не должны регуляризовать веса смещений. Установите $\lambda = 0.0001$. Запрограммируйте регуляризованную версию сети, обучите ее с помощью мини-пакетного градиентного спуска и постройте такие же два графика, как и в предыдущем вопросе. При этом

помните, что функция потерь, используемая для построения графиков, **не должна** включать регуляризационный терм (регуляризация используется только для обучения сети).

Включите оба графика в ваш отчет и в одном-двух предложениях проведите сравнительный анализ обоих моделей.

Сохраните веса обученной модели в файл, чтобы можно было их загрузить при выполнении следующего задания.

Вопрос №3 [1 балл]

Все это время вы должны были хранить тестовую выборку в неприкосновенности. Теперь, когда вы убедились в том, что ваша модель работает как положено (т.е. регуляризация сети дает тот эффект, который следует из теории), пришло время оценить эффективность сети на тестовой выборке. Обратите внимание, что на основе тестовой выборки мы получаем итоговую оценку эффективности классификатора и какой бы она ни оказалась на этом работа с нашей моделью полностью заканчивается – мы **не возвращаемся** назад и **не пытаемся** улучшить показатели сети. Это стандартный сценарий работы с тестовой выборкой в машинном обучении – она в каком-то смысле является «одноразовой». В противном случае вы начнете подгонять вашу модель под тестовую выборку и в этот самый момент она превратится в валидационную, в результате чего объективной оценки обобщающей способности вашей модели вы не получите.

Загрузив веса сначала нерегуляризованной, а потом регуляризованной сети, вычислите точность модели в обоих случаях. Включите полученные результаты в отчет. Обратите внимание, что результатом будут два числа, а не графики.