Оглавление

[Постановка задачи 2](#_Toc24401714)

[Архитектура нейронной сети 3](#_Toc24401715)

[Метод обратного распространения ошибки 4](#_Toc24401716)

[Вычисление градиентов 4](#_Toc24401717)

[Программная реализация 6](#_Toc24401718)

[Описание класса 6](#_Toc24401719)

[Детали запуска 8](#_Toc24401720)

[Результаты экспериментов 9](#_Toc24401721)

# Постановка задачи

Пусть – множество объектов (входов), каждый элемент которого описывает изображение размером , а – множество ответов (выходов) для каждого изображения, где – кол-во классов, которым данное изображение может принадлежать, причем

|  |  |
| --- | --- |
|  | (1) |

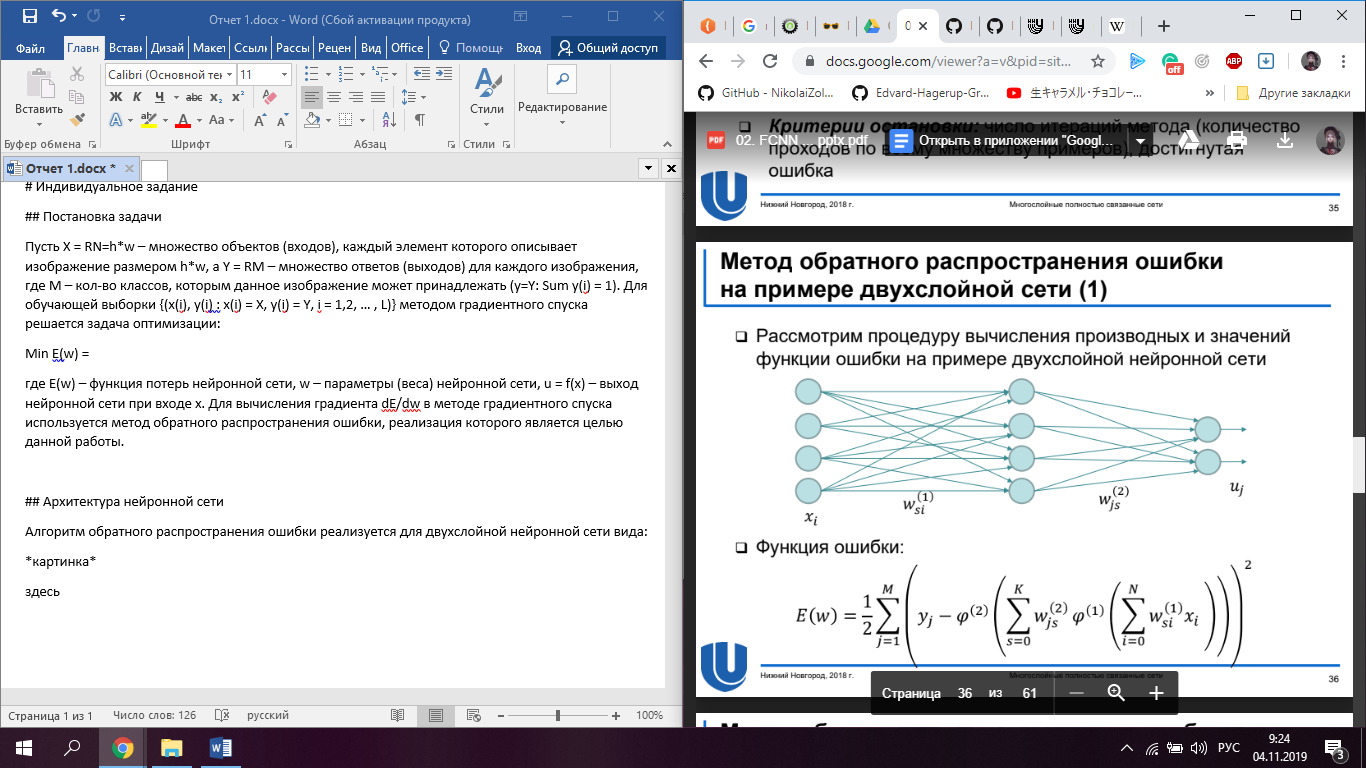
Для обучающей выборки методом градиентного спуска решается задача оптимизации:

|  |  |
| --- | --- |
|  | (2) |

где – функция потерь нейронной сети, – параметры (веса) нейронной сети, u = f(x) – выход нейронной сети при входе . Для вычисления градиента в методе градиентного спуска используется метод обратного распространения ошибки, реализация которого является целью данной работы.

# Архитектура нейронной сети

Алгоритм обратного распространения ошибки реализуется для двухслойной полносвязной нейронной сети вида:



здесь N – размерность входа (N = h\*w), K – количество нейронов скрытого слоя, M – количество нейронов выходного слоя (соответствует количеству классов), – веса первого слоя, – веса второго слоя.

Выходы скрытого слоя обозначим :

|  |  |
| --- | --- |
|  | (3) |

где - функция активации ReLU:

|  |  |
| --- | --- |
|  | (4) |

Выходы нейронной сети (выходного слоя) обозначим :

|  |  |
| --- | --- |
|  | (5) |

где – функция активации softmax:

|  |  |
| --- | --- |
|  | (6) |

В качестве функции потерь используем кросс-энтропию:

|  |  |
| --- | --- |
|  | (7) |

# Метод обратного распространения ошибки

Метод обратного распространения ошибки реализуется путем прямого и обратного прохода по нейронной сети. Прямой проход заключается в вычислении выходов каждого слоя сети по формулам (3) и (5). Обратный проход включает в себя вычисление значения функции потерь и ее градиента. После этого в направлении антиградиента выполняется шаг градиентного спуска для обновления весов:

|  |  |
| --- | --- |
|  | (8) |

где – скорость обучения.

В качестве критерия остановки используется ограничение итераций метода – проходов по всей обучающей выборке. Для повышения производительности шаг градиентного спуска будет выполнятся в направлении усредненного антиградиента по нескольким входам (батчу).

## Вычисление градиентов

1. Вычисление градиента по весам второго слоя :

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |

где

|  |  |
| --- | --- |
|  | (11) |
|  | (12) |
|  | (13) |
|  | (14) |

Подставим (10) и (14) в (9):

|  |  |
| --- | --- |
|  | (15) |
|  |  |
|  |  |

Из условия (1) получаем окончательную формулу вычисления градиента по весам второго слоя:

|  |  |
| --- | --- |
|  | (16) |

1. Вычисление градиента по весам первого слоя :

|  |  |
| --- | --- |
|  | (17) |
|  | (18) |

где

|  |  |
| --- | --- |
|  | (19) |
|  | (20) |
|  | (21) |

Подставим (18) в (17) и разобьем на две суммы:

|  |  |
| --- | --- |
|  | (22) |
|  |  |

Из условия (1):

|  |  |
| --- | --- |
|  | (23) |

Таким образом окончательная формула вычисления градиента по весам первого слоя имеет вид:

|  |  |
| --- | --- |
|  | (24) |

# Программная реализация

## Описание класса

Описанная выше нейронная сеть реализована в классе NeuralNetwork.

Поля класса NeuralNetwork:

|  |  |
| --- | --- |
| Название | Описание |
| w1 | – массив numpy вещественных чисел размера (N, K), хранящий значения весов первого слоям |
| w2 | – массив вещественных чисел длины K\*M, хранящий значения весов второго слоя |
| bias1 | – массив вещественных чисел длины K, соответствующий нейрону смещения первого слоя |
| bias2 | – массив вещественных чисел длины M, соответствующий нейрону смещения второго слоя |

Методы класса NeuralNetwork:

|  |  |
| --- | --- |
| Название | Описание |
| NeuralNetwork | – конструктор класса. Принимает на вход три значения целочисленного типа N, K и M - размеры входного, скрытого и выходного слоев соответственно. Значения по умолчанию: N = 784, K = 50, M = 10. Выполняет начальную инициализацию весов первого и второго слоя нейронной сети случайными числами из нормального распределения и заполнение весов нейрона смещения нулями. |
| phi\_1 | – функция активации скрытого слоя (в данном случае ReLU). Принимает на вход массив numpy вещественных чисел любого размера и возвращает массив numpy (такого же размера) значений активаций, вычисленных по формуле (4) для каждого значения входного массива.  **Примечание**: в рамках алгоритма предполагается, что входной массив двумерный и соответствует произведению матрицы входов и матрицы весов первого слоя . |
| phi\_2 | – функция активации выходного слоя (в данном случае softmax). Принимает на вход двумерный массив numpy любого размера и возвращает двумерный массив (такого же размера), соответствующий значениям активаций второго слоя, вычисленных по формуле (6) для каждого элемента входного массива.  **Примечание**: в рамках алгоритма предполагается, что входной массив является произведением матрицы активаций нейронов скрытого слоя и матрицы весов второго слоя |
| dphi\_1 | – производная функции активации скрытого слоя (в данном случае производная ReLU). Принимает на вход массив numpy вещественных чисел любого размера и возвращает массив numpy (такого же размера) значений производной, вычисленных по формуле (20) для каждого значения входного массива. |
| loss | – функция потерь (в данном случае кросс-энтропия). Принимает на вход двумерный массив numpy , соответствующий матрице значений активаций выходного слоя, и двумерный массив , соответствующий меткам классов входных данных сети. Возвращает вектор значений функции потерь, вычисленных по формуле (7) для каждой пары соответствующих строк из матрицы u и матрицы y. |
| predict | – функция классификации. Принимает на вход двумерный массив numpy, соответствующий матрице входов , и возвращает двумерный массив numpy, соответствующий матрице выходов , посчитанный по формуле (5). |
| get\_metrics | – подсчет ошибки и точности сети. Принимает на вход двумерный массив numpy u, соответствующий матрице значений активаций выходного слоя, и двумерный массив y, соответствующий меткам классов входных данных сети. Возвращает два вещественных значения – ошибку и точность. |
| fit | – подбор весов нейронной сети методом обратного распространения ошибки. Обязательными аргументами являются двумерный массив numpy X – обучающая выборка , и двумерный массив numpy Y – метки классов , соответствующих элементам обучающей выборки. Также на вход принимается целочисленное значение batch\_size – размер батча , целочисленное значение iter – ограничение по эпохам обучения, вещественное значение eta – скорость обучения; так же можно подать на вход тестовую выборку (x\_test и y\_test) для фиксирования метрик качества на ней после каждой эпохи. Значения по умолчанию: batch\_size=1, iter=20, eta=0.1, x\_test=None, y\_test=None. Возвращает словарь, хранящий значения метрик качества после каждой итерации: “err” – список длины iter значений ошибки (усредненных по батчу) после каждой итерации, “acc” – список длины iter значений точности (усредненных по батчу) после каждой итерации, “err\_test” – список длины iter значений ошибки на тестовом наборе (усредненных по батчу) после каждой итерации, “acc\_test” – список длины iter значений точности на тестовом наборе (усредненных по батчу) после каждой итерации.  **Примечание**: при отсутствии тестового набора (x\_test=None, y\_test=None) по ключам “err\_test” и “acc\_test” хранятся пустые списки. |

## Детали запуска

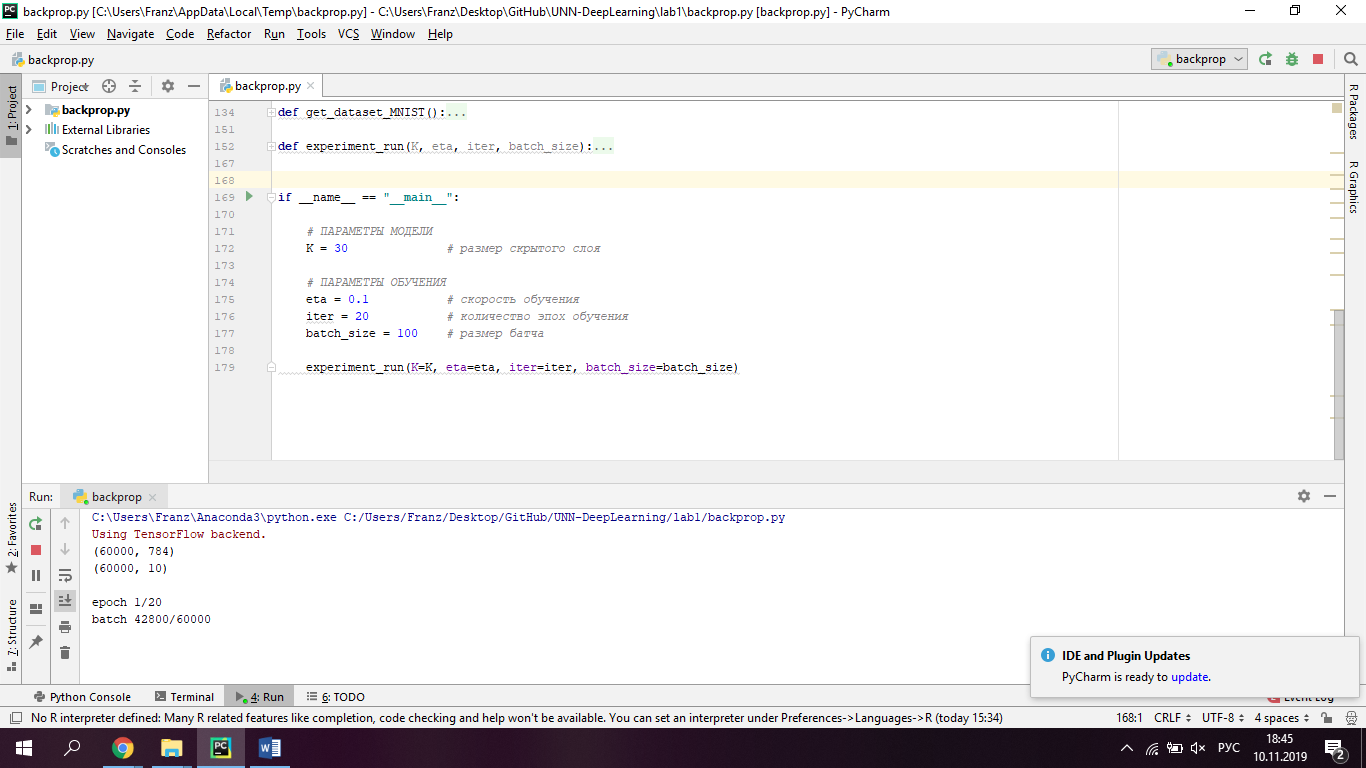
Для проведения эксперимента по обучению нейронной сети на наборе данных MNIST необходимо запустить файл backprop.py, предварительно задав желаемые параметры сети и обучения:

* K – размер скрытого слоя нейронной сети

**Примечание**: размерs входного слоя (N) и выходного слоя (M) не указываются, поскольку соответствуют набору MNIST для которого проводится эксперимент.

* eta – скорость обучения
* iter – ограничение по эпохам обучения
* batch\_size – размер поднабора обучающей выборке (батча), по которому будет усредняться шаг градиентного спуска

Пример инициализации параметров:

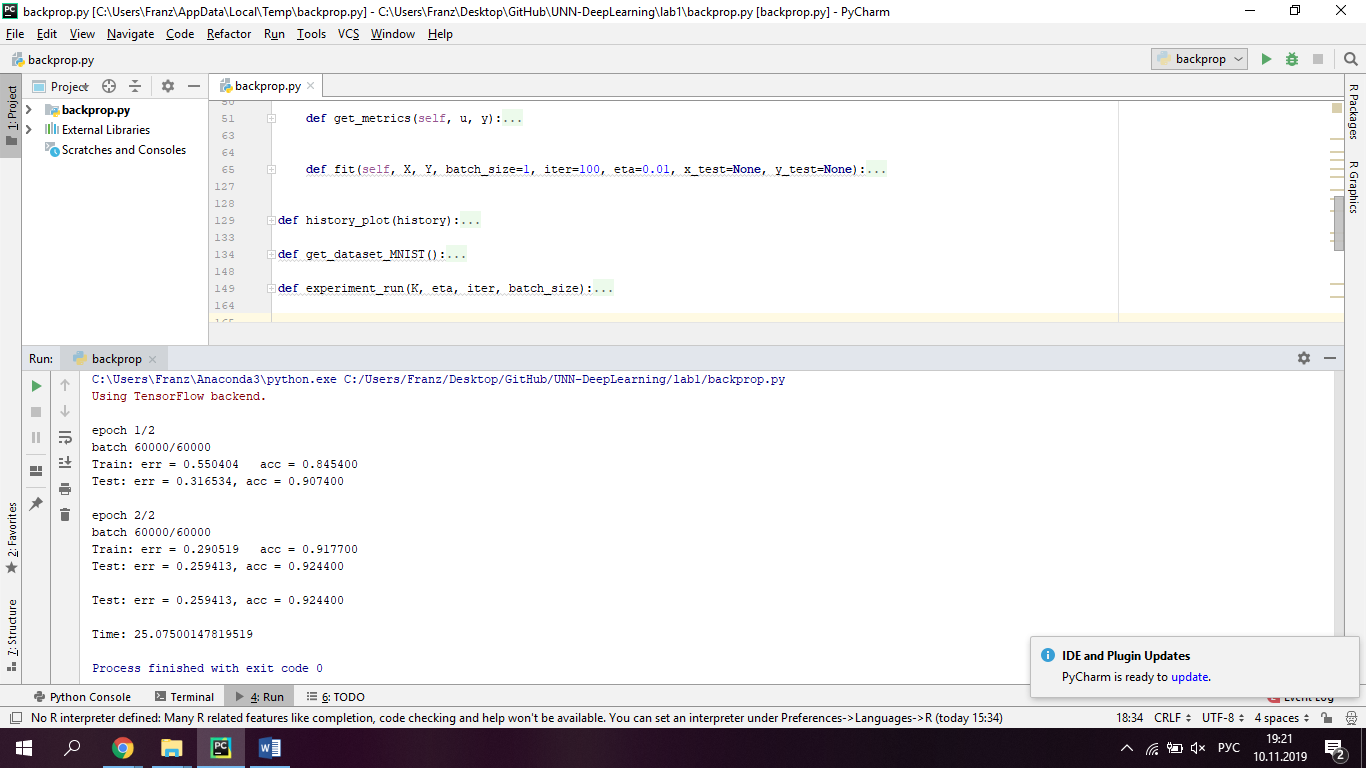


Эксперимент заключается в последовательном выполнении следующих действий:

1. Загрузка обучающего и тестового наборов данных MNIST.
2. Преобразование набора картинок X к набору нормированных векторов.
3. Преобразование набора категориальных признаков Y методом One-Hot Encoding к набору векторов.
4. Инициализация нейронной сети, описанной выше архитектуры, с заданным размером скрытого слоя.
5. Обучение нейронной сети методом обратного распространения ошибки с заданными параметрами на тренировочном наборе (загруженном в пункте 1) с выводом значений метрик качества на тестовом наборе (загруженном в пункте 1). Время обучения фиксируется.
6. После завершения обучения строятся графики изменения ошибки и точности на обучающем и тестовом наборах.

После запуска файла backprop.py процесс обучения нейронной сети будет отображаться по средствам вывода номера эпохи, пройденного объема тренировочного набора данных и метрик качества на тренировочном и тестовом наборах, усредненных по всему набору.

Пример выводимых данных (при ограничении в две эпохи обучения):



# Результаты экспериментов

Ниже приведены результаты экспериментов с разными комбинациями параметров обучения и сети.

1. Изменение размера скрытого слоя:

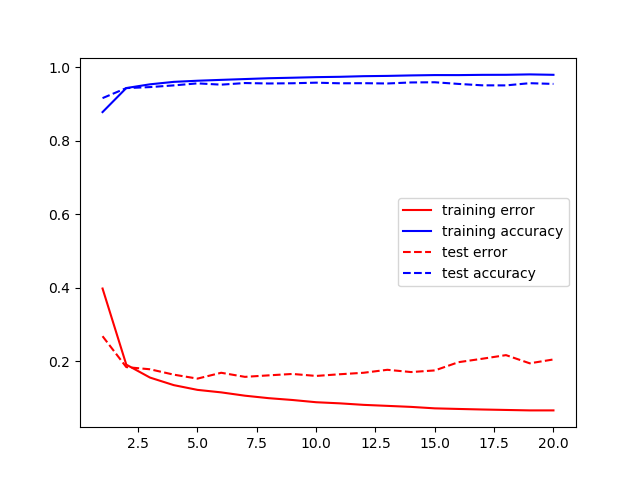
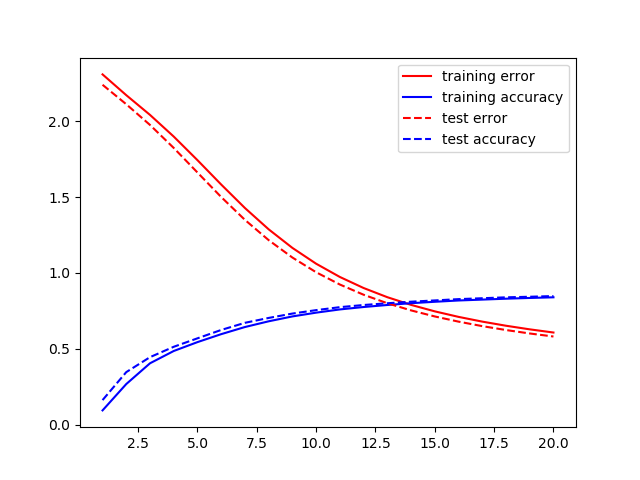
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер скрытого слоя (K) | Ошибка | | Точность | |
| Тренировочный набор | Тестовый набор | Тренировочный набор | Тестовый набор |
| 30 | 0.087120 | 0.116840 | 0.974633 | 0.966600 |
| 50 | 0.066745 | 0.102559 | 0.980900 | 0.970300 |
| 70 | 0.057679 | 0.090380 | 0.984067 | 0.972400 |

Чем больше размер скрытого слоя, тем больше информации о входных данных можно сохранить, что объясняет лучшие метрики качества.

1. Изменение скорости обучения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Скорость обучения (eta) | Ошибка | | Точность | |
| Тренировочный набор | Тестовый набор | Тренировочный набор | Тестовый набор |
| 1 | 0.067074 | 0.205593 | 0.979267 | 0.954500 |
| 0.1 | 0.092072 | 0.117558 | 0.973483 | 0.962900 |
| 0.01 | 0.272220 | 0.262699 | 0.922400 | 0.925700 |
| 0.001 | 0.607965 | 0.581690 | 0.839833 | 0.848000 |

При недостаточной скорости обучения изменение ошибки плавное, но медленное, что мешает достигнуть желаемого минимума за относительно небольшое число эпох (рис. 1а), избыточная – может стать причиной “проскакивания” минимума на каждом шаге градиентного спуска, что может привести к удалению от решения (рис. 1б).



а)

б)

Рис. 1. График зависимости ошибки (красный) и точности (синий) на тренировочном (сплошная линия) и тестовом (пунктирная линия) наборах при обучении нейронной сети с размером скрытого слоя K=30 на 20 эпохах с размером батча 100 со скоростью обучения: a) 0.001 (недообучение), б) 1 (метод расходится).

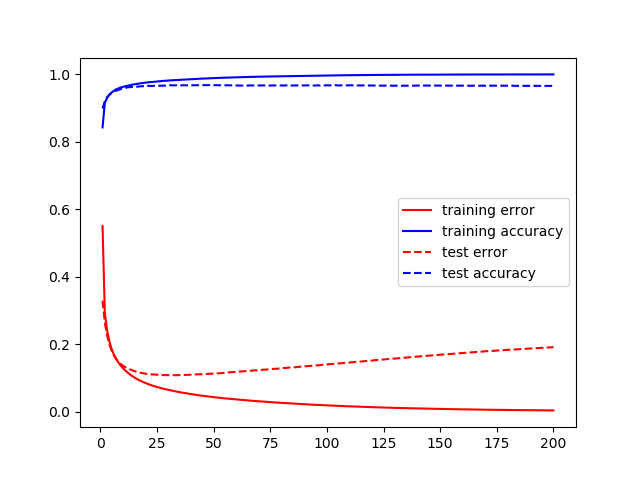
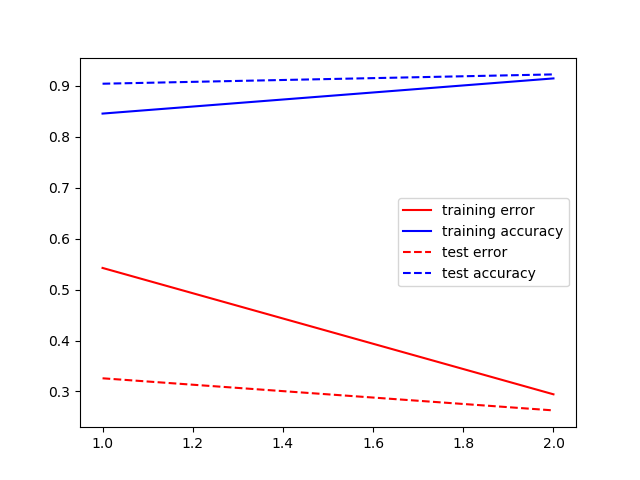
1. Изменение размера батча

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер батча (batch\_size) | Ошибка | | Точность | |
| Тренировочный набор | Тестовый набор | Тренировочный набор | Тестовый набор |
| 10 | 0.051975 | 0.181970 | 0.983200 | 0.961900 |
| 100 | 0.089794 | 0.116622 | 0.973867 | 0.966500 |
| 1000 | 0.265435 | 0.256556 | 0.924200 | 0.927400 |

1. Изменение количества эпох обучения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество эпох (iter) | Ошибка | | Точность | |
| Тренировочный набор | Тестовый набор | Тренировочный набор | Тестовый набор |
| 2 | 0.294344 | 0.262734 | 0.914833 | 0.922700 |
| 20 | 0.097652 | 0.130103 | 0.971867 | 0.961000 |
| 200 | 0.004693 | 0.191990 | 0.999933 | 0.965900 |

Слишком большое количество эпох приводит к переобучению (рис. 2б), слишком мало эпох – не позволяет сети обучиться вовсе (рис. 2а).



а)

б)

Рис. 2. График зависимости ошибки (красный) и точности (синий) на тренировочном (сплошная линия) и тестовом (пунктирная линия) наборах при обучении нейронной сети с размером скрытого слоя K=30 со скоростью обучения 0.1 и размером батча 100 на: a) 2 эпохах (недообучение), б) 200 эпохах (переобучение).