# **Лабораторная работа №1**

Цель лабораторной работы состоит в том, чтобы изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой). Сеть ориентирована на решение задачи классификации рукописных цифр одноканальных изображений на примере набора данных MNIST.

На входе сети имеется нейронов, в соответствии с разрешением изображения. На выходе сети имеется нейронов – количество классов изображений. Скрытый слой содержит нейронов. В качестве функции активации на первом слое используется функция активации ReLU. В качестве функции активации на втором слое используется функция softmax. В качестве функции ошибки используется кросс-энтропия.

Математическая модель нейрона описывается двумя уравнениями:

где – входной сигнал, – синаптический вес входного сигнала , – смещение, – функция активации.

Метод обратного распространения ошибки

Данный метод определяет стратегию изменения параметров сети в ходе обучения с использованием градиентных методов оптимизации. Градиентные методы на каждом шаге уточняют значения параметров:

Где – скорость обучения – скорость движения в направлении минимального значения функции, , – направление в многомерном пространстве параметров нейронной сети. В классическом методе распространения ошибки направление движения совпадает с направлением антиградиента .

Алгоритм метода обратного распространения ошибки:

1. Инициализация синаптических весов сети (случайным образом из некоторого распределения)
2. Повторение следующих шагов для каждого примера тренировочного набора данных
   1. Прямой проход:
      1. Вычисление значений выходных сигналов нейронов всех слоев ()
      2. Вычисление значений производных функций активации на каждом слое сети ( и )
   2. Обратный проход:
      1. Вычисление значения целевой функции и ее градиента
      2. Корректировка синаптических весов
3. Критерий остановки: число итераций метода (количество проходов по всему множеству примеров).

Один полный цикл предъявления полного набора примеров называется эпохой. Сеть обучается заданное число эпох. В каждой эпохе набор тестовых данных делится на пакеты, и затем происходит обучение на пакете данных по методу обратного распространения ошибки.

Функции активации

* На скрытом слое используется ReLU:
* На выходном слое используется функция *softmax*: , ее производная: и

Функция ошибки

В качестве функции ошибки используется кросс-энтропия:

Где – выход сети, – ожидаемый выход, – число нейронов на выходном слое, – веса выходного слоя, – выход скрытого слоя, – число нейронов на скрытом слоев, – число нейронов на входе сети, – веса скрытого слоя.

Производная функции ошибки по выходному слою:

Производная функции ошибки по скрытому слою:

Описание программной реализации

Данные из библиотеки MNIST считываются с помощью фреймворка keras, преобразуются в вектор и нормируются. Далее происходит инициализация начальных весов случайным образом из равномерного распределения на отрезке [-0.2, 0.2] и [-0.5, 0.5]. Перед обучением происходит перемешивание выборки.

Для каждой эпохи делятся данные на пакеты и для каждого пакета выполняются шаги алгоритма метода обратного распространения ошибки для обучения по тренировочной выборке.

После обучения происходит проверка модели на тестовой выборке и подсчет точности и потери.

Результаты

Для сравнения результатов использовался фреймворк keras. Параметры обучения сети:

* hidden\_dim – размерность скрытого слоя
* training\_epoch – количество эпох для обучения сети
* batch\_size – размер пакета
* learning\_rate – параметр скорости обучения

Тест 1.

* hidden\_dim = 64
* training\_epoch = 20
* batch\_size = 128
* learning\_rate = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Время, с | Тестовая точность, % | Тестовая потеря | Тренировочная точность, % | Тренировочная потеря |
| Backpropagation | 13.942769 | 0.9699 | 0.0951804 | 0.9831166 | 0.0598751 |
| keras | 12.933231 | 0.9777 | 0.0826 | 0.9813 | 0.0660 |

Тест 2.

* hidden\_dim = 128
* training\_epoch = 20
* batch\_size = 128
* learning\_rate = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Время, с | Тестовая точность, % | Тестовая потеря | Тренировочная точность, % | Тренировочная потеря |
| Backpropagation | 16.133751 | 0.9752 | 0.08499 | 0.98598 | 0.0425 |
| keras | 15.648913 | 0.9797 | 0.0768 | 0.9854 | 0.0538 |

Тест 3.

* hidden\_dim = 256
* training\_epoch = 20
* batch\_size = 128
* learning\_rate = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Время, с | Тестовая точность, % | Тестовая потеря | Тренировочная точность, % | Тренировочная потеря |
| Backpropagation | 53.526592 | 0.9772 | 0.07552 | 0.994816 | 0.027224 |
| keras | 18.185767 | 0.9797 | 0.0713 | 0.9879 | 0.0471 |

Тест 4.

* hidden\_dim = 512
* training\_epoch = 20
* batch\_size = 128
* learning\_rate = 0.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Время, с | Тестовая точность, % | Тестовая потеря | Тренировочная точность, % | Тренировочная потеря |
| Backpropagation | 93.588923 | 0.9755 | 0.081129 | 0.998583 | 0.01523 |
| keras | 42.345201 | 0.9807 | 0.0706 | 0.9896 | 0.0425 |