МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»
Магистерская программа: «Когнитивные системы»

ОТЧЕТ

по Практической работе №1

Реализация метода обратного распространения ошибки для двухслойной полностью связанной нейронной сети

Выполнил:
студент группы 381806-4м
Шульпин Степан

Нижний Новгород
2019

Цель

Цель настоящей работы состоит в том, чтобы изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой).

Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Изучение общей схемы метода обратного распространения ошибки.
2. Вывод математических формул для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов.
3. Проектирование и разработка программной реализации.
4. Тестирование разработанной программной реализации.
5. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Метод обратного распространения ошибки

Метод обратного распространения ошибки определяет стратегию изменения параметров сети w в ходе обучения с использованием градиентных методов оптимизации.

Градиентные методы на каждом шаге уточняют значения параметров:

$$w(k+1) = w(k) + \eta p(w)$$

$\eta, 0 < \eta < 1$ – скорость обучения (learning rate) – «скорость» движения в направлении минимального значения функции,

$p(w)$ – направление в многомерном пространстве параметров нейронной сети.

В классическом методе обратного распространения ошибки направление движения совпадает с направлением антиградиента $p(w) = -\nabla E(w)$.

Общая схема метода обратного распространения ошибки включает несколько основных этапов. Первоначально синаптические веса сети инициализируются определенным образом, например, нулевыми значениями или случайно из некоторого распределения. Далее метод работает для каждого примера обучающей выборки:

1. Прямой проход нейронной сети в направлении передачи информации от входного сигнала к скрытым слоям и выходному слою сети. На данном этапе вычисляются значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие значения производных функций активации на каждом слое сети.
2. Вычисление значения целевой функции и градиента этой функции.
3. Обратный проход нейронной сети в направлении от выходного слоя к входному слою, и корректировка синаптических весов.
4. Повторение этапов 1 – 3 до момента выполнения критериев остановки. В качестве критериев остановки используется число итераций метода (количество проходов), либо достигнутая точность.

Расчеты градиентов и обновления весов для двуслойной полносвязной нейронной сети с функцией активации *Softmax* на выходном слое и *Сигмоидальной* на скрытом слое представлены ниже.

Кросс-энтропийная функция ошибки для задачи классификации с помощью полносвязной сети с одним скрытым слоем

$$E(w) = -\frac{1}{L}\sum_{k=1}^{L}\sum_{j=1}^{M} y_j^k \ln u_j^k ,$$

где $u_j^k = \varphi_2\left(\sum_{s=0}^{K} w_{js}^{(2)}\, \varphi_1\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right)$.

$$p_j = \sum_{s=0}^{K} w_{js}^{(2)}\, \varphi_1\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right), v_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i$$

Производная *Softmax:*

$$\varphi_2(p_j)'_{p_j} = \left(\frac{e^{p_j}}{\sum_{i=1}^{M} e^{p_i}}\right)'_{p_j} = \frac{e^{p_j}}{\sum_{i=1}^{M} e^{p_i}} - \left(\frac{e^{p_j}}{\sum_{i=1}^{M} e^{p_i}}\right)^2 = \varphi_2(p_j)\left(1 - \varphi_2(p_j)\right)$$

$$\varphi_2(p_j)'_{p_{k\neq j}} = \left(\frac{e^{p_j}}{\sum_{i=1}^{M} e^{p_i}}\right)'_{p_k} = \frac{-e^{p_j}e^{p_k}}{(\sum_{i=1}^{M} e^{p_i})^2} = -\varphi_2(p_j)\varphi_2(p_k)$$

$$\varphi_2(p_j)'_{p_k} = \varphi_2(p_j)\left(\delta(p_j, p_k) - \varphi_2(p_k)\right)$$

Производная *ReLu*:

$$\varphi_1(p)'_p = \left(p\&(p \geq 0)\right)'_p = 1\&(p \geq 0)$$

Тогда частная производная по весу скрытого слоя равна:

$$\frac{\partial E}{\partial w_{mn}^{(2)}} = \left(-\sum_{j=1}^{M} y_j \ln u_j\right)'_{w_{mn}^{(2)}}$$

$$= -\sum_{j=1}^{M} y_j \frac{1}{\varphi_2(p_j)} \varphi_2(p_j)\left(\delta(p_j, p_m) - \varphi_2(p_m)\right)\frac{\partial p_m}{\partial w_{mn}^{(2)}}$$

$$= -\sum_{j=1}^{M} y_j \left(\delta(p_j, p_m) - \varphi_2(p_m)\right)\varphi_1(v_n)$$

$$= \{-y_m(1 - \varphi_2(p_m)) + \sum_{j=1\neq m}^{M} y_j\, \varphi_2(p_m)\}\, \varphi_1(v_n)$$

$$= \{\varphi_2(p_m)(y_m + \sum_{j=1\neq m}^{M} y_j) - y_m\}\varphi_1(v_n) = (\varphi_2(p_m) - y_m\ )\varphi_1(v_n)$$

Частная производная по весам входного слоя:

$$\frac{\partial E}{\partial w_{nl}^{(1)}} = \left(-\sum_{j=1}^{M} y_j \ln u_j \right)'_{w_{nl}^{(2)}}$$

$$= -\sum_{j=1}^{M} y_j \frac{1}{\varphi_2(p_j)} \varphi_2(p_j) \left(\delta(p_j, p_j) - \varphi_2(p_j) \right) \frac{\partial p_j}{\partial w_{nl}^{(1)}}$$

$$= \sum_{j=1}^{M} \left(y_j - \varphi_2(p_j) \right) \sum_{s=0}^{K} w_{js}^{(2)} \varphi_1(v_s)'_{w_{nl}^{(1)}}$$

$$= \sum_{j=1}^{M} \left(y_j - \varphi_2(p_j) \right) \sum_{s=0}^{K} w_{js}^{(2)} \varphi_1'(v_s) v_s{}'_{w_{nl}^{(1)}}$$

$$= \sum_{j=1}^{M} \left(y_j - \varphi_2(p_j) \right) w_{jn}^{(2)} \varphi_1'(v_n) x_l \sum_{i=0}^{N} w_{ni}^{(1)} x_i$$

Программная реализация

Была написана программа на языке python, решающая задачу классификации рукописных цифр на основе базы данных рукописных цифр MNIST.
Класс *Network* решает задачу обучения нейронной сети. Экземпляр класса создается со следующими параметрами:

- **input_layer_size** – число нейронов входного слоя
- **hidden_layer_size** – число нейронов скрытого слоя
- **output_layer_size** – число нейронов выходного слоя

Метод *train* тренирует сеть и принимает на вход:

- objects – объекты для тренировки сети
- labels – разметка для объектов
- epochs – число эпох
- learning_rate – скорость обучения
- batch_size – размер пакетов

Метод *test* проверяет качество сети и принимает на вход:

- objects – объекты для проверки сети
- labels – разметка для объектов

Результаты работы программы:

Количество эпох	Количество нейронов скрытого слоя	Скорость обучения	Размер пакета	Точность на тренировочных данных	Точность на тестовых данных	Время обучения, с	Точность на тренировочных данных (keras)	Точность на тестовых данных (keras)	Время обучения (keras), с
10	16	0.1	128	0.9366	0.9313	07.18	0.9508	0.9488	05.51
20	100	0.1	128	0.9800	0.9726	20.70	0.9869	0.9775	15.35
10	100	0.1	256	0.9460	0.9465	09.49	0.9593	0.9571	07.45
10	300	0.1	128	0.9656	0.9621	30.63	0.9794	0.9714	12.53
20	784	0.1	128	0.9834	0.9758	2:17.43	0.9922	0.9805	56.88