МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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

Кафедра Математического обеспечения и суперкомпьютерных технологий

Направление подготовки: «Прикладная математика и информатика»
Магистерская программа: «Вычислительная математика и суперкомпьютерные технологии»

ОТЧЕТ

по практической работе №1:

Реализация метода обратного распространения ошибки для двухслойной полностью связанной нейронной сети

Выполнила:
студентка группы 381703-3м
Береснева Юлия Васильевна

_____

Нижний Новгород
2018

Содержание

Цели и задачи

Цель: изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой).

Задачи:

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.

Вывод математических формул

Выведем математические формулы для вычисления градиентов функции ошибки по параметрам нейронной сети и формулы коррекции весов.

В качестве функции ошибки используется кросс-энтропия.

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

Найдем производную целевой функции по параметрам последнего слоя:

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

$$u_j = \varphi^{(2)}(g_j) = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} v_s\right)$$

$$g_j = \sum_{s=0}^{K} w_{js}^{(2)} v_s$$

$$v_s = \varphi^{(1)}(f_s) = \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)$$

$$f_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i$$

$$u_j = \varphi^{(2)}(g_j) = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} v_s\right) = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} \varphi^{(1)}(f_s)\right) =>$$

$$u_j = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right)$$

Тогда подставим в производную:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \frac{\partial\left(-\sum_{j=1}^{M} y_j \ln \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right)\right)}{\partial w_{js}^{(2)}} =$$

$$= -\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j}\frac{dg_j}{\partial w_{js}^{(2)}} = -\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j}v_s = \delta_j^{(2)}v_s$$

$$\delta_j^{(2)} = -\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j} = \frac{\partial E(w)}{\partial g_j}$$

То есть:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \delta_j^{(2)}v_s$$

Найдем производную целевой функции по параметрам скрытого слоя:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \frac{\partial\left(-\sum_{j=1}^{M} y_j \ln \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)}\varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)}x_i\right)\right)\right)}{\partial w_{si}^{(1)}} =$$

$$= -\sum_{j=1}^{M}\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j}\frac{dg_j(v_s)}{dv_s}\frac{d\varphi^{(1)}(f_s)}{df_s}\frac{df_s}{w_{si}^{(1)}} =$$

$$= -\sum_{j=1}^{M}\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j}\, w_{js}^{(2)}\frac{d\varphi^{(1)}(f_s)}{df_s}x_i = \delta_s^{(1)}x_i$$

$$\delta_s^{(1)} = -\sum_{j=1}^{M}\frac{y_j}{u_j}\frac{d\varphi^{(2)}(g_j)}{dg_j}\, w_{js}^{(2)}\frac{d\varphi^{(1)}(f_s)}{df_s} = \frac{\partial E(w)}{\partial f_s}$$

То есть

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \delta_s^{(1)}x_i$$

Описание метода обратного распространения ошибки

Метод обратного распространения ошибки является градиентным методом минимизации функции ошибки для полностью связанной нейронной сети, веса которой изменяются по формуле:

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

где η – скорость обучения, $p(w)$ – направление сдвига в пространстве параметров сети. Опишем общую схему метода:

1. Прямой проход. Во время прямого прохода вычисляется
   - выход сети для некоторого входа путем послойного вычисления значений в нейронах
   - производные от функций активации нейронов для вычисленных значений
2. Вычисление функции ошибки и ее производных по весам выходного слоя
3. Обратный проход. В время обратного прохода
   - вычисляются градиенты функции ошибки на слоях
   - корректируются веса

Шаги повторяются до тех пор, пока не будет выполняться критерий останова – достижение необходимой точности / достижение необходимого числа итераций.

Распишем формулы для нашей задачи. Если принять за $\varphi^{(1)}(u)$ сигмоидальную функцию: $\varphi^{(1)}(u) = \frac{1}{1+e^{-u}}$, а за $\varphi^{(2)}(u_j)$ – функцию *softmax* $\varphi^{(2)}(u_j) = \frac{e^{u_i}}{\sum_{k=1}^{M} e^{u_k}}$, то получим следующие значения для производных по параметрам:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = -\frac{y_j}{u_j}\frac{e^{g_j}(\sum_{k=1}^{M} e^{g_k} - e^{g_j})}{(\sum_{k=1}^{M} e^{g_k})^2} v_s = \delta_j^{(2)} v_s$$

$$\frac{\partial E}{\partial w_{si}^{(1)}} = -\sum_{j=1}^{M} \frac{y_j}{u_j}\frac{e^{g_j - f_s}(\sum_{k=1}^{M} e^{g_k} - e^{g_j})}{(\sum_{k=1}^{M} e^{g_k})^2 (1+e^{-f_s})^2} w_{js}^{(2)} x_i = \delta_s^{(1)} x_i$$

Программная реализации

В ходе лабораторной работы реализовали программу для обучения двуслойной нейронной сети, решающей задачу классификации, на языке программирования Python 3.6.

Класс *Classifier* решает задачу обучения нейронной сети. В процессе работы программа принимает на вход следующие значения:

- число итераций, выполняемых в ходе обучения – число эпох
- значение параметра скорости обучения

которые передаются в конструктор класса при создании объекта вместе с другими параметрами:

- *hidden_neurons* - число нейронов скрытого слоя
- *batch_size* - размер батча
- *eps* - точность

Для обучения созданной сети на примерах из обучающей выборки используем метод *fit()*, на вход которому подается вектор с метками класса. Затем вызывается метод *predict()*, принимающий на вход данные из тестовой выборки и предсказывает по ней выход с помощью ранее обученной сети. Также класс использует следующие методы:

- *fit_batch* – функция, осуществляющая обучение сети на пачке (batch)
- *count_error* – функция, считающая ошибку предсказания для обучающей выборки, выводит высчитанную ошибку.
- *softmax* – функция активации, включая производные
- *cross_entropy_error* – функция ошибки, включая производные
- *mix* – функция возвращающая случайную перестановку элементов выборки

Тестирование на наборе данных MNIST

Созданная программная реализация тестировалась на обучающем наборе данных MNIST, представляющего из себя 70 000 одноканальных изображений, размера 28x28 пикселей, с рукописными цифрами.

Обучение сети происходило на тренировочной выборке размера 60000 изображений, тестирование - на тестовой выборке набора данных MNIST размера 10000 изображений.

Измерение точности классификации (как отношение количества правильно классифицированных к числу всех предсказаний) и сбор результатов экспериментов при разном наборе параметров метода описан в таблице ниже:

Таблица 1

Число эпох	Скорость обучения	Тестовая точность, %	Тренировочная точность, %
5	0.5	0.85215	0.8539
10	0.5	0.86335	0.865
15	0.5	0.87412	0.8808
5	0.2	0,85768	0.8623
10	0.2	0.86391	0.8655
15	0.2	0.8866	0.8912
5	0.1	0.86448	0.8741
10	0.1	0.8776	0.8801
15	0.1	0.88393	0.8861
5	0.01	0.83001	0.8381
5	0.05	0.85541	0.8666
10	0.05	0.86703	0.8671
15	0.05	0.8795	0.8812

При этом число нейронов на скрытом слое равно 800, точность составляет 0,001, размер батча равен 100.