Лабораторная работа №1 Реализация метода обратного распространения ошибки для двухслойной полностью связанной нейронной сети

Малова Анастасия Алексеевна

November 2019

1 Цель работы

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).

2 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.

В качестве тестовой задачи была взята задача классификации одноканальных изображений на примере набора данных MNIST.

3 Вывод формул

Математическая модель нейрона имеет следующий вид:
$u_k = \sum_{j=n}^{n} w_{k,i} x_j$
$y_k = \phi(u_k)$, где - ϕ функция активации, w - вес, x - вход.

Рассмотрим двухслойную нейронную сетю (с одним скрытым слоем). Начальная инициализация нейронов - случайна.

Функции активации

На скрытом слое используем функцию ReLU:

$\phi^{(1)}(u) = max(0, u)$

На выходном слое используем функцию Softmax:

$\phi^{(2)}(u_j) = \frac{e^{u_j}}{\sum_{i=0}^{u} i}$

Производные:

$\frac{\partial \phi^{(2)}(u_j)}{\partial u_j} = \phi^{(2)}(u_j)(1 - (\phi^{(2)}(u_j)))$
$\frac{\partial \phi^{(2)}(u_j)}{\partial u_i} = -\phi^{(2)}(u_j)\phi^{(2)}(u_i)$

Функция ошибки

В качестве функции ошибки была рассмотрена кросс-энтропия:

$E(w) = \sum_{j=1}^{M} y_j \ln u_j$
$u_j = \phi^{(2)}(\sum_{m=0}^{K} w_{j,m}^{(2)} v_m)$
$v_m = \phi^{(1)}(\sum_{i=0}^{N} w_{s,i}^{(1)} x_i)$
где
N - кол-во входных нейронов
M - кол-во выходных нейронов
K - кол-во нейронов на скрытом слое
L - кол-во обучающих примеров

Найдем производные функции ошибки по весам.

1. По выходному слою:

$\frac{\partial E(w)}{\partial w_{j,m}^{(2)}} = (y_j - u_i) v_m = \delta_j^{(2)} v_m$

2. По скрытому слою:

$\frac{\partial E(w)}{\partial w_{s,i}^{(1)}} = (y_j - u_i) w^{(2)_{j,m}} x_i * \frac{\partial \phi^{(1)}(\sum_{i=0}^{N} w_{s,i}^{(1)} x_i)}{\partial \sum_{i=0}^{N} w_{s,i}^{(1)} x_i)} = \delta_s^{(1)} x_j (\phi^{(1)})'$

4 Програмная реализация

Процессе обучения многослойной полносвязной нейронной сети - многократное предъявляение предопределенного множества обучающих примеров.
Эпоха - полный цикл предъявления полного набора примеров.
Для обучения нейронной сети использовался пакетный режим метода обратного распространения ошибки. Внутри эпохи выборка делилась на пакеты и производилась корректировка весов для каждого пакета методом обратного распростарнения ошибки.

Метод обратного распространения ошибки:

1. Прямой проход нейронной сети 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 весов.
4. Повторение шагов 1 – 3 до момента выполнения критериев остановки.

Обучение на тестовой выборке

Сеть обучается заданное число эпох. Внутри эпохи набор тестовых данных делится на пакеты.

Алгоритм:
Для каждой эпохи:

1. Перетосовываем выборку
2. Делим на пакеты
3. Для каждого пакета:
   Метод обратного распространения ошибки для корректировки весов

Программная реализация была написана на языке Python 3 при помощи библиотек:
tensorflow - основа для Keras
keras - фреймворк для сравнения реализаций,так же отуда были взяты данные MNIST
numpy - для работы с векторами и матрицами
datetime - для замера времени работы

Для запуска обучения и тестирования нейронной сети на базе MNIST необходимо выполнить в командной строке следующее:

python MyLab.py hiddenSize batchSize epochs lr

hiddenSize - int, число нейронов скрытого слоя
batchSize - int, размер пакета
epochs - int, кол-во эпох
lr - float, скорость обучения

5 Тестирование програмной реализации

hiddenSize	batchSize	epochs	lr	Точность	Время	Точность Keras	Время Keras
30	128	5	0.1	0.9397	3.5812	0.9470	3.0628
30	128	5	0.01	0.878	3.5468	0.9045	2.9732
30	128	10	0.1	0.9542	7.1627	0.9564	5.7077
30	128	10	0.01	0.9014	7.0677	0.9125	6.02
256	128	5	0.1	0.9707	16.8158	0.9605	5.714
256	128	5	0.01	0.9357	16.9877	0.9094	5.9987
256	128	10	0.1	0.9759	35.4503	0.9718	11.1614
256	128	10	0.01	0.9509	34.4536	0.9267	11.7348
256	128	20	0.1	0.9775	67.8124	0.9769	21.8207
256	256	5	0.1	0.9634	13.8468	0.9466	4.7379
256	256	5	0.01	0.9213	14.0427	0.8967	5.0428
256	256	10	0.1	0.9702	29.2512	0.9588	9.2725
256	256	20	0.1	0.9749	56.6560	0.9714	17.8178