

Нижегородский государственный университет им. Н. И. Лобачевского
Институт информационных технологий, математики и механики

Направление подготовки Прикладная математика и информатика

Магистерская программа Вычислительные методы и суперкомпьютерные
технологии

Образовательный курс «Методы глубокого обучения для решения задач
компьютерного зрения»

Отчёт

по лабораторной работе № 1

**«Реализация метода обратного распространения ошибки для
двуслойной полностью связанной нейронной сети»**

Выполнил:

студент гр. 381603м4

Розанов А.

Нижний Новгород
2017

Содержание

- Постановка задачи3
- Метод обратного распространения ошибки4
 - Обозначения.....4
 - Нейронная сеть4
 - Описание метода4
- Программная реализация7
 - Класс Neuron.....7
 - Класс Layer7
 - Класс HiddenLayer7
 - Класс OutputLayer.....8
 - Класс Network.....8

Постановка задачи

Основной целью данной лабораторной работы является изучение метода обратного распространения ошибки для обучения глубоких нейронных сетей на примере двуслойной полностью связанной сети, ориентированной на решение задачи классификации рукописных цифр из набора данных MNIST.

В ходе выполнения данной лабораторной работы предполагается выполнение следующих подзадач:

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. Разработка программной реализации.
4. Тестирование разработанной программной реализации.

Метод обратного распространения ошибки

Обозначения

$x_i^{(n)}$ – вход на i -й нейрон (n) -го слоя.

$b_i^{(n)}$ – сдвиг на i -м нейроне (n) -го слоя.

$w_{ij}^{(n)}$ – вес от i -го нейрона $(n-1)$ -го слоя к j -му нейрону (n) -го слоя.

N_n – количество нейронов на (n) -м слое.

$z_j^{(n)} = b_j^{(n)} + \sum_{i=1}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)} = \sum_{i=0}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)}$ – значение линейной комбинации вектора весов и вектора входных значений j -го нейрона (n) -го слоя.

$y_i^{(n)} = f^{(n)}(z_i^{(n)})$ – выход i -го нейрона (n) -го слоя, где $f^{(n)}$ – функция активации.

t_i – ожидаемый выход i -го нейрона последнего слоя.

Нейронная сеть

Будем рассматривать двухслойную полносвязную нейронную сеть с функцией активации softmax на последнем слое и с функцией ошибки кросс-энтропия.

При функции активации softmax получаем следующие выходы:

$$y_i^{(n)} = \frac{e^{z_i^{(n)}}}{\sum_{j=1}^n e^{z_j^{(n)}}}$$

Функция ошибки кросс-энтропия имеет следующий вид:

$$C = - \sum_{j=1}^{N_2} t_j \log(y_j^{(n)})$$

Описание метода

В ходе метода обратного распространения ошибки выполняется уточнение весов нейронной сети с целью уменьшения ошибки. В начальный момент веса инициализируются случайным образом. Далее для каждого примера обучающей выборки выполняется следующая процедура:

1. Прямой проход нейронной сети, на котором вычисляются выходы каждого слоя.
2. Вычисление значений целевой функции и градиента этой функции.
3. Обратный проход нейронной сети, при котором корректируются веса.
4. Повторение этапов 1-3 до момента выполнения критериев остановки (количество проходов или достигнутая точность).

Корректировка весов происходит по следующей формуле:

$$w(k+1) = w(k) + \Delta w,$$

где $\Delta w = \eta p(w)$, $0 < \eta < 1$ – скорость обучения, $p(w)$ – направление в многомерном пространстве параметров нейронной сети. Направление движения совпадает с направлением антиградиента:

$$p(w) = -\nabla C(w)$$

В соответствии с выбранной нейронной сети, целевая функция записывается следующим образом:

$$C = - \sum_{i=1}^{N_2} t_i \log(y_i^{(2)}) = - \sum_{i=1}^{N_2} t_i \log \left(f^{(2)} \left(\sum_{j=0}^{N_1} w_{ji}^{(2)} f^{(1)} \left(\sum_{s=0}^{N_0} w_{sj}^{(1)} x_s^{(0)} \right) \right) \right)$$

Найдем производную целевой функции по параметрам (n)-го слоя:

$$\frac{\partial C}{\partial w_{ji}^{(n)}} = \frac{\partial C}{\partial z_i^{(n)}} \frac{\partial z_i^{(n)}}{\partial w_{ji}^{(n)}}$$

$$\frac{\partial z_i^{(n)}}{\partial w_{ji}^{(n)}} = \sum_{j=0}^{N_{n-1}} \frac{\partial w_{ji}^{(n)} x_j^{(n)}}{\partial w_{ji}^{(n)}} = x_j^{(n)}$$

Рассмотрим производные по последнему слою:

$$\frac{\partial C}{\partial z_i^{(2)}} = \sum_{j=1}^{N_2} \frac{\partial C}{\partial y_j^{(2)}} \frac{\partial y_j^{(2)}}{\partial z_i^{(2)}}$$

$$\frac{\partial C}{\partial y_j^{(2)}} = - \frac{\partial \left(\sum_{k=1}^{N_2} t_k \log y_k^{(2)} \right)}{\partial y_j^{(2)}} = - \frac{\partial \left(t_j \log y_j^{(2)} \right)}{\partial y_j^{(2)}} = - \frac{t_j}{y_j^{(2)}}$$

$$\frac{\partial y_j^{(2)}}{\partial z_i^{(2)}} = \begin{cases} y_i^{(2)} (1 - y_i^{(2)}), & \text{если } i = j \\ -y_i^{(2)} y_j^{(2)}, & \text{если } i \neq j \end{cases}$$

$$\frac{\partial C}{\partial y_j^{(2)}} \frac{\partial y_j^{(2)}}{\partial z_i^{(2)}} = \begin{cases} - \frac{t_j}{y_j^{(2)}} y_i^{(2)} (1 - y_i^{(2)}) = -t_i (1 - y_i^{(2)}), & \text{если } i = j \\ \left(- \frac{t_j}{y_j^{(2)}} \right) (-y_i^{(2)} y_j^{(2)}) = t_j y_i^{(2)}, & \text{если } i \neq j \end{cases}$$

$$\frac{\partial C}{\partial z_i^{(2)}} = \sum_{j=1, j \neq i}^{N_2} t_j y_i^{(2)} - t_i (1 - y_i^{(2)}) = -t_i + t_i y_i^{(2)} + y_i^{(2)} \sum_{j=1, j \neq i}^{N_2} t_j = -t_i + y_i^{(2)} \left(t_i + \sum_{j=1, j \neq i}^{N_2} t_j \right)$$

Так как $t_i + \sum_{j=1, j \neq i}^{N_2} t_j = 1$ (сумма значений выходного вектора), то $\frac{\partial C}{\partial z_i^{(2)}} = y_i^{(2)} - t_i$

$$\frac{\partial C}{\partial w_{ji}^{(2)}} = x_j^{(2)} (y_i^{(2)} - t_i)$$

Рассмотрим производные по скрытому слою:

$$\frac{\partial C}{\partial w_{ji}^{(1)}} = \sum_{k=1}^{N_1} \frac{\partial C}{\partial y_k^{(1)}} \frac{\partial y_k^{(1)}}{\partial z_i^{(1)}} x_j^{(1)}$$

$$\frac{\partial \mathcal{C}}{\partial y_k^{(1)}} = \sum_{s=1}^{N_2} \frac{\partial \mathcal{C}}{\partial y_s^{(2)}} \frac{\partial y_s^{(2)}}{\partial z_s^{(2)}} \frac{\partial z_s^{(2)}}{\partial y_k^{(1)}} = \sum_{s=1}^{N_2} \frac{\partial \mathcal{C}}{\partial y_s^{(2)}} \frac{\partial y_s^{(2)}}{\partial z_s^{(2)}} w_{ks}^{(2)}$$

$$\frac{\partial \mathcal{C}}{\partial y_s^{(2)}} \frac{\partial y_s^{(2)}}{\partial z_s^{(2)}} = y_s^{(2)} - t_s$$

$$\frac{\partial \mathcal{C}}{\partial w_{ji}^{(1)}} = x_j^{(1)} f' \left(z_i^{(1)} \right) \sum_{k=1}^{N_1} \sum_{s=1}^{N_2} \left(y_s^{(2)} - t_s \right) w_{ks}^{(2)}$$

Программная реализация

Класс Neuron

Программное представление искусственного нейрона.

Поля:

1. `int numPrevNeurons` – количество нейронов на предыдущем слое.
2. `vector<double> weights` – веса на связях от нейронов предыдущего слоя к текущему.
3. `vector<double> inputs` – входы от нейронов предыдущего слоя.
4. `double bias` – смещение для текущего нейрона.

Методы:

1. `double getSum()` – сумматор.
2. `void updateBias(double gradient, double learnRate)` – обновление смещения.
3. `void updateWeights(double gradient, double learnRate)` – обновление весов.

Класс Layer

Абстрактный класс для представления слоя в нейронной сети.

Поля:

1. `int numNeurons` – количество нейронов на текущем слое.
2. `int numPrevNeurons` – количество нейронов на предыдущем слое.
3. `vector<Neuron> neurons` – вектор нейронов принадлежащих слою.
4. `vector<double> outputs` – выход слоя. Состоит из выходов каждого нейрона.

Методы:

1. `virtual double activateNeuron(vector<double> &sums, double sum)` – виртуальный метод активации нейронов в слое.
2. `vector<double> computeOutputs()` – вычисление выходов каждого нейрона.
3. `void updateWeights(vector<double> &gradient, double learnRate)` – обновление весов у каждого нейрона.
4. `void updateBiases(vector<double> &gradient, double learnRate)` – обновление смещений у каждого нейрона.
5. `vector<double> computeWeightedSumErrors(vector<double> &gradient)` – рассчитывает взвешенную сумму ошибок для предыдущего слоя.

Класс HiddenLayer

Наследник класса `Layer`, предназначен для представления скрытого слоя. Реализует методы:

1. `vector<double> computeGradient(vector<double> &layerOutputs, vector<double> &nextLayerWeightedSumErrors)`
2. `virtual double activateNeuron(vector<double> &sums, double sum)`

Класс OutputLayer

Наследник класса Layer, предназначен для представления последнего выходного слоя. Реализует методы:

1. `virtual double activateNeuron(vector<double> &sums, double sum)`
2. `vector<double> computeGradient(vector<double> &outputs, vector<double> &labels)`

Класс Network

Представление искусственной нейронной сети.

Поля:

1. `HiddenLayer *hiddenLayer` – скрытый слой.
2. `OutputLayer *outputLayer` – выходной слой.

Методы:

1. `vector<double> test(vector<double> &inputs)` – тестирование сети. Возвращает предсказанное значение.
2. `double train(vector<vector<double>> trainInputs, vector<vector<double>> trainLabels, int maxEpoches, double minError, double learnRate)` – обучение сети.
3. `double MeanCrossEntropyError(vector<vector<double>> trainInputs, vector<vector<double>> trainLabels)` – вычисление средней ошибки.
4. `vector<double> forwardPropagation()` – прямой проход.
5. `void backwardPropagation(vector<double> &hiddenOutputs, vector<double> &outputs, vector<double> &expectedValues, double learnRate)` – обратный проход.