

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»**

Выполнила:

студентка гр. 381603м4
Вершинина О. С.

Нижний Новгород
2017

Оглавление

Постановка задачи	3
Вывод математических формул для метода обратного распространения ошибки в случае решения задачи классификации	4
Общая схема метода обратного распространения ошибки	7
Программная реализация	8
Результаты экспериментов	9

Постановка задачи

Необходимо изучить и реализовать метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двуслойной полностью связанной сети (один скрытый слой).

В работе рассматривается сеть, ориентированная на решение задачи классификации рукописных цифр из набора данных MNIST.

Для выполнения лабораторной работы требуется решить следующие задачи:

1. Изучить общую схему метода обратного распространения ошибки.
2. Вывести математические формулы для вычисления градиентов функции ошибки по параметрам нейронной сети и формулы коррекции весов.
3. Разработать программную реализацию.
4. Протестировать разработанную программную реализацию.

Вывод математических формул для метода обратного распространения ошибки в случае решения задачи классификации

Введём следующие обозначения:

N – количество входных нейронов;

M – количество выходных нейронов;

K – количество нейронов на скрытом слое;

L – количество обучающих примеров.

В качестве функции ошибки рассматривается кросс-энтропия:

$$E(w) = -\frac{1}{L} \sum_{k=1}^L \sum_{j=1}^M y_j^k \ln(u_j^k), y_j^k = 1 \leftrightarrow x^k \in j - \text{му классу},$$

где $y^k = (y_j^k)_{j=\overline{1,M}} \in Y$ – множество обучающих примеров, а $u^k = (u_j^k)_{j=\overline{1,M}}$ – выход нейронной сети, полученный для входного примера $x^k = (x_i^k)_{i=\overline{1,N}} \in X$.
Учитываем, что $x_0 = 1$ – новый нейрон с весами $w_{0s}^{(1)}$.

Будем рассматривать последовательный (стохастический) режим обучения. В этом режиме корректировка весов выполняется после предъявления каждого примера обучающей выборки.

Возьмём конкретный обучающий пример $x = (x_1, x_2, \dots, x_N)$,
 $y = (y_1, y_2, \dots, y_M)$, $u = (u_1, u_2, \dots, u_M)$.

Тогда имеем $E(w) = -\sum_{j=1}^M y_j \ln(u_j)$. Обозначим $w_{is}^{(1)}$ – веса синаптических связей от входных нейронов к нейронам скрытого слоя, $w_{sj}^{(2)}$ – от нейронов скрытого слоя к выходным нейронам.

Выходной сигнал нейрона скрытого слоя вычисляется как $v_s = \varphi(f_s)$,
где φ – функция активации на скрытом слое,

$$f_s = \sum_{i=0}^N w_{is}^{(1)} x_i, \quad s = \overline{0, K} - \text{взвешенная сумма входных сигналов.}$$

Сигнал выходного нейрона определяется как $u_j = h(g_j)$,

где h – функция активации на последнем слое,

$$g_j = \sum_{s=0}^K w_{sj}^{(2)} v_s, \quad j = \overline{1, M} - \text{взвешенная сумма сигналов со скрытого слоя.}$$

В качестве функции активации на последнем слое рассматривается *softmax*:

$$u_j = \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}.$$

Таким образом,

$$E(w) = -\sum_{j=1}^M y_j \ln\left(\frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}\right) = -\sum_{j=1}^M y_j (g_j - \ln \sum_{m=1}^M e^{g_m}),$$

$$g_j = \sum_{s=0}^K w_{sj}^{(2)} \varphi\left(\sum_{i=0}^N w_{is}^{(1)} x_i\right).$$

Задача обучения нейронной сети сводится к задаче оптимизации функции ошибки по всем синаптическим весам $E(w) \rightarrow \min_w$. Метод обратного распространения ошибки определяет стратегию выбора параметров сети w с использованием градиентных методов оптимизации.

Производная целевой функции по параметрам последнего слоя $w_{sj}^{(2)}$ вычисляется по следующей формуле:

$$\frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial w_{sj}^{(2)}},$$

$$\frac{\partial g_j}{\partial w_{sj}^{(2)}} = v_s,$$

$$\delta_j^{(2)} = \frac{\partial E}{\partial g_j} = -\frac{\partial}{\partial g_j} \left[\sum_{j=1}^M y_j (g_j - \ln \sum_{m=1}^M e^{g_m}) \right] = -(-y_1 \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - \dots - y_{j-1} \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} + y_j \left(1 - \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} \right) - y_{j+1} \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - \dots - y_M \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}) =$$

$$(\sum_{j=1}^M y_j) \cdot \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - y_j = \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - y_j = u_j - y_j.$$

Тут был использован тот факт, что в задачах классификации $\sum_{j=1}^M y_j = 1$.

$$\text{В итоге получаем } \frac{\partial E(w)}{\partial w_{sj}^{(2)}} = (u_j - y_j) \cdot v_s = \delta_j^{(2)} \cdot v_s.$$

Производная целевой функции по параметрам скрытого слоя $w_{is}^{(1)}$ вычисляется по формуле:

$$\frac{\partial E(w)}{\partial w_{is}^{(1)}} = \frac{\partial E}{\partial f_s} \frac{\partial f_s}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i.$$

$$\frac{\partial E}{\partial f_s} = \sum_{j=1}^M \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial v_s} \frac{\partial \varphi}{\partial f_s} = \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial v_s} = \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \delta_j^{(2)} w_{sj}^2$$

$$\text{В итоге имеем } \frac{\partial E(w)}{\partial w_{is}^{(1)}} = \frac{\partial \varphi}{\partial f_s} [\sum_{j=1}^M \delta_j^{(2)} w_{sj}^2] \cdot x_i.$$

Если функция активации на скрытом слое гиперболический тангенс $\varphi(f_s) = th(f_s)$, то

$$\frac{\partial \varphi}{\partial f_s} = (1 - \varphi) * (1 + \varphi) = (1 - v_s) * (1 + v_s).$$

Градиент имеет следующую структуру: $\frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \delta_j^{(2)} \cdot v_s$, $\frac{\partial E(w)}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i$.

Согласно градиентным методам на каждом шаге $r + 1$ обучения сети производится коррекция весов по следующим формулам:

$$w_{is}^{(1)(r+1)} = w_{is}^{(1)(r)} - \eta \frac{\partial E(w)}{\partial w_{is}^{(1)}},$$

$$w_{sj}^{(2)(r+1)} = w_{sj}^{(2)(r)} - \eta \frac{\partial E(w)}{\partial w_{sj}^{(2)}}.$$

где η – скорость обучения.

Общая схема метода обратного распространения ошибки

1. Инициализировать веса w случайными значениями.
2. *for epoch = 1, number_epochs*
3. *for i = 0, N, N = Width_Image * Height_Image*
4. **Прямой проход** нейронной сети в направлении передачи информации от входного сигнала к скрытому и к выходному слою сети.

Подать на вход x_i . Вычислить значения выходных сигналов нейронов скрытого слоя $v_s, s = \overline{0, K}, K$ – количество нейронов на скрытом слое и значение производной функции активации на скрытом слое $\frac{\partial \varphi}{\partial f_s}$. Вычислить выходные сигналы нейронов последнего слоя $u_j, j = \overline{1, M}, M$ – количество классов изображений.

$$x_i \rightarrow v_s, \frac{\partial \varphi}{\partial f_s} \rightarrow u_j$$

Обратный проход нейронной сети в направлении от выходного слоя к входному слою и корректировка синаптических весов. Вычисляются значения градиентов целевой функции.

$$\text{for } j = \overline{1, M}$$

$$\delta_j^{(2)} = u_j - y_j, \frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \delta_j^{(2)} \cdot v_s$$

$$\text{for } s = \overline{0, K}$$

$$\delta_s^{(1)} = - \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \delta_j^{(2)} w_{sj}^2, \frac{\partial E(w)}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i.$$

for по каждой дуге

$$w_{is}^{(1)} = w_{is}^{(1)} - \eta \frac{\partial E(w)}{\partial w_{is}^{(1)}}$$

$$w_{sj}^{(2)} = w_{sj}^{(2)} - \eta \frac{\partial E(w)}{\partial w_{sj}^{(2)}}$$

5. Повторение этапов 3 - 5 до момента выполнения критериев остановки. В качестве критериев остановки используется число итераций метода (количество проходов *number_epochs*), либо достигнутая точность $E(w_{current}) < tolerance_error_cross_entropy$.

Программная реализация

Программная реализация метода обратного распространения ошибки была разработана на языке C++ в соответствии с приведённым выше алгоритмом и выведенными формулами. Запуск программы производился в Microsoft Visual Studio 2010.

В проекте *MethodBackPropagation* имеются следующие файлы:

- *read_mnist.h* – заголовочный файл, содержащий методы для чтения набора данных MNIST.
- *activation_function.h* – заголовочный файл, содержащий методы для вычисления функции активации гиперболический тангенс и её производной, а так же метод для вычисления функции softmax.
- *config.h* – заголовочный файл, содержащий структуру TaskConfig, в которой содержатся следующие поля: пути до директорий с выборками данных MNIST, количество нейронов на скрытом слое, максимальное количество эпох, скорость обучения, требуемая точность.
- *config.cpp* – файл содержащий методы для чтения конфигурационного файла config.txt.
- *neuron_net.h*, *neuron_net.cpp* – класс, представляющий собой нейронную сеть с возможностью обучения методом обратного распространения ошибки.
- *main.cpp* – тестовое приложение для классификации рукописных цифр на базе данных MNIST.

Особенность программной реализации: реализована модель нейрона с дополнительным синапсом $x_0 = 1$. Поэтому фактически количество входных нейронов равно $N = Width_{Image} * Height_{Image} + 1$, а количество скрытых нейронов $K = numberHiddenNeurons + 1$, где *numberHiddenNeurons* – число нейронов скрытого слоя, заданное в config-файле. Входной вектор имеет следующий вид: на первом месте стоит 1, а далее 28*28 пикселей изображения.

Результаты экспериментов

Точность классификации определялась по следующей формуле

$$precision = \frac{true}{true + false},$$

true – количество правильно определённых цифр, *false* – количество неправильно определённых цифр.

При проведении экспериментов в качестве требуемого значения целевой функции использовалось значение 0.005, остальные параметры изменялись. Результаты собраны в таблице 1.

Таблица 1. Результаты экспериментов

Число нейронов скрытого слоя	Скорость обучения	Количество эпох для достижения требуемой точности	Точность классификации на тренировочном наборе	Точность классификации на тестовом наборе
100	0.008	16	0.999850	0.979100
100	0.01	14	0.999867	0.978800
200	0.008	14	0.999700	0.98150
200	0.005	20	0.999750	0.982000