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
Семеренко А. И.

Нижний Новгород
2018

Оглавление

1. Постановка задачи

В ходе лабораторной работы предполагается решение ***следующих задач***:

- Изучение общей схемы метода обратного распространения ошибки.

- Вывод математических формул для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов.

- Проектирование и разработка программной реализации сети, решающей задачу классификации рукописных символов.

- Тестирование разработанной программной реализации.

- Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Для обучения и тестирования сети предполагается использовать набор данных MNIST.

Метод обратного распространения ошибки разрабатывается, исходя из следующих предположений:

- На входе сети имеется $w \times h$ нейронов, что соответствует разрешению изображения.

- На выходе сети имеется k нейронов, что соответствует количеству классов изображений.

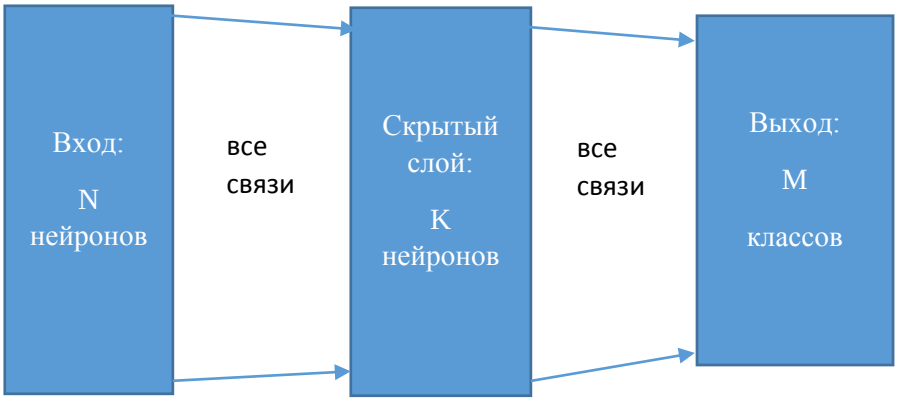
- Скрытый слой содержит s нейронов.

- В качестве функции активации на втором слое используется функция softmax.

- В качестве функции ошибки используется кросс-энтропия.

2. Вывод математических формул метода обратного распространения ошибки для задачи классификации рукописных символов.

Нейронная сеть имеет: N - входных нейронов, K - нейронов на скрытом слое и M - выходных.



В целом метод обратного распространения ошибки сводится к задаче минимизации функции ошибки относительно синоптических весов. В качестве целевой функции будем использовать кросс энтропию.

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

где $y^k = \left(y_j^k\right),\ j = \overline{1,M}, k = \overline{1,L}$ – k-й пример из обучающей выборки, $u^k = \left(u_j^k\right), j = \overline{1,M}, k = \overline{1,L}$– выход нейронной сети на входе $x^k = \left(x_i^k\right),\ i = \overline{1,\mathrm{N}}, k = \overline{1,L}$. Также учтем, что $x_0 = 1$ – новый нейрон.

Для вывода формул и реализации будем использовать предположение, что режим обучения последовательный. Корректировка весов выполняется для каждого элемента обучающей выборки, а также $\sum_{j=1}^{M} y_j = 1$ для задач классификации.

Выведем сначала формулы для одного конкретного элемента выборки. Пусть: $x^* = (x_i^*),\ i = \overline{1,\mathrm{N}}$, $y^* = \left(y_j^*\right),\ j = \overline{1,M}, u^* = \left(u_j^*\right), j = \overline{1,M}$, тогда целевая функция для конкретного примера имеет вид:

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

Обозначим синоптические веса от входного слоя к скрытому - $\mathrm{w}_{\mathrm{is}}^{(1)}$, от скрытого к выходному - $\mathrm{w}_{\mathrm{sj}}^{(2)}$. Также обозначим $h\left(g_j\right) = \frac{e^{g_j}}{\sum_{i=1}^{M} e^{g_i}}$ – функция активации на выходном слое (softmax), $\varphi(f_s) = \tanh(g_j)$ – функция активации на скрытом слое (гиперболический тангенс). В нашем случае все формулы для прямого прохода от входа к выходу выглядят следующим образом:

$$f_s = \sum_{i=0}^{N} w_{is}^{(1)} x_i\,, s = \overline{0,\mathrm{K}}$$

$$v_s = \varphi(f_s), s = \overline{0,\mathrm{K}}$$

$$g_j = \sum_{s=0}^{K} w_{sj}^{(2)} v_s, j = \overline{1, \mathrm{M}}$$

$$u_j = h(g_j), j = \overline{1, \mathrm{M}}$$

Итак, с учетом всех вышеуказанных формул и предположений:

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

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

Поскольку уже говорилось, что задача сводится к минимизации функции ошибки, в частности градиентными методами, то корректировка весов должна выполнятся по следующим формулам:

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

$$\mathrm{w}_{sj}^{(2)(\mathrm{r}+1)} = \mathrm{w}_{\mathrm{sj}}^{(2)(\mathrm{r})} - \eta \frac{\partial \mathrm{E}(\mathrm{w})}{\partial \mathrm{w}_{\mathrm{sj}}^{(2)}},$$

где η – скорость обучения ($0 < \eta < 1$).

Получим градиент для корректировки весов.

Частная производная целевой функции по весам от скрытого к выходному слою.

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

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

$$\delta_j^{(2)} = \frac{\partial E}{\partial g_j} = \left(\sum_{j=1}^{M} y_j\right) \frac{e^{g_j}}{\sum_{i=1}^{M} e^{g_i}} - y_j = \frac{e^{g_j}}{\sum_{i=1}^{M} e^{g_i}} - y_j = u_j - y_j$$

Частная производная целевой функции по весам от входного к скрытому слою.

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

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

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

3. Описание программной реализации

Реализован класс DoubleLayerFCNN.

Конструктор принимает на вход размеры слоев: входного, скрытого и выходного.

Класс имеет 3 публичных метода:

- train(double **data, double *label, int sampleSize, int numberEpochs, double learningRate, double errorCrossEntropy)
- precision(double **data, double *label, int sampleSize)
- cleanNetWeights() – сброс значений весов

где data – набор данных (количество элементов * размер входного слоя), label – разметка (количество элементов), sampleSize – количество элементов выборки, learningRate – скорость обучения, numberEpochs и errorCrossEntropy – количество эпох обучения и требуемая ошибка, по совместительству являются критериями останова.

Общий алгоритм обучения сети:

1. Инициализация весов случайными значениями.
2. Для каждого элемента выборки.
3. Прямой проход: вычисляем значения на скрытом слое, далее вычисляем значения на выходном слое.
4. Обратный проход: вычисляем градиенты и корректируем веса
5. Подсчитываем кросс-энтропию
6. Повторяем пункты 2-5 пока кросс-энтропия не достигнет требуемого значения или не закончится указанное число эпох.

Точность в нашем случае считается как отношение верно определенных изображений к их общему числу.

Для запуска программы необходимо клонировать репозиторий, и запустить один из .bat скриптов в зависимости от версии, установленной на вашем компьютере Visual Studio. Если у вас установлена другая версия, то в корне репозитория из командной строки вызвать соответствующую команду cmake. Открыть собранное решение и запустить проект DL_Lab1_Backpropagation.

4. Результаты экспериментов

Требуемое значение кросс-энтропии для всех экспериментов было указано 0.005. Результаты представлены в таблице ниже.

Число нейронов скрытого слоя	Скорость обучения	Количество эпох	Точность классификации на тестовом наборе	Точность классификации на обучающем наборе
50	0.005	34	0.9731	0.9997
100	0.005	17	0.9789	0.9997
200	0.01	12	0.9803	0.9998
200	0.005	20	**0.9836**	0.9997