Лабораторная работа №1

Задание

- Изучить метод обратного распространения ошибки;
- Вывести математические формулы для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов;
- Спроектировать и разработать программную реализацию;
- Подготовить отчет по проделанной работе.

Запуск решения

1. Открыть решение в VisualStudio 2017
2. Положить базу mnist в одноименную директорию внутри проекта
3. Собрать проект и запустить исполняемый файл

Метод обратного распространения ошибки

Ошибку вычисляем с использованием функции кросс-энтропии:

$$E = -\sum_{j=1}^{N_o} t_j log(y_j) = -\sum_{j=1}^{N_o} t_j log(f(\sum_{s=1}^{N_s} w_{sj} f(\sum_{i=1}^{N_i} w_{is} x_i)))$$

На скрытом слое в качестве функции активации используется гиперболический тангенс:

$$\phi(y_j) = \frac{e^{2y_j} - 1}{e^{2y_j} + 1}$$

На выходном слое в качестве функции активации используется функция softmax:

$$f(y_j) = \frac{e^{y_j}}{\sum_{j=1}^{n} e^{y_j}}$$

Алгоритм

Инициализируем веса значениями от -0.5 до 0.5.

В цикле по количеству эпох:

1. Перемешиваем порядок изображений.
2. В цикле по количеству изображений:
   2.1. В цикле по количеству скрытых нейронов:
      2.1.1.Считаем взвешенную сумму сигнала с первого слоя.
      2.1.2.Применяем функцию активации гиперболический тангенс.
   2.2. В цикле по количеству нейронов выходного слоя:
      2.2.1.Считаем взвешенную сумму сигнала со скрытого слоя.
   2.3. Применяем функцию активации softmax.
   2.4. Считаем градиенты функции ошибки

$$\frac{\partial E}{\partial w_{sj}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{sj}}, \quad \frac{\partial E}{\partial y_j} = u_j - t_j, \quad \frac{\partial y_j}{\partial w_{sj}} = v_s$$

$$\frac{\partial E}{\partial w_{sj}} = (u_j - t_j)v_s = \delta_j v_s$$

$$\frac{\partial E}{\partial w_{is}} = \frac{\partial E}{\partial z_s}\frac{\partial z_s}{\partial w_{is}}$$

$$\frac{\partial E}{\partial z_s} = \sum_{j=1}^{N_o}\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_s}\frac{\partial f}{\partial z_s} = \frac{\partial f}{\partial z_s}\sum_{j=1}^{N_o}\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_s} = \frac{\partial f}{\partial z_s}(\sum_{j=1}^{N_o}\delta_j^2 w_{sj}^2)$$

$$\frac{\partial E}{\partial w_{is}} = \frac{\partial f}{\partial z_s}(\sum_{j=1}^{N_o}\delta_j^2 w_{sj}^2)x_i$$

Так как используется гиперболический тангенс, то:

$$\frac{\partial f}{\partial z_s} = (1 - v_s)(1 + v_s)$$

И тогда:

$$\frac{\partial E}{\partial w_{sj}} = \delta_j v_s, \quad \frac{\partial E}{\partial w_{is}} = \delta_s x_i$$

   2.5. Корректируем веса:

$$w_{i_s}^{n+1} = w_{i_s}^{n} - \eta\frac{\partial E}{\partial w_{is}}$$

$$w_{s_j}^{n+1} = w_{s_j}^{n} - \eta\frac{\partial E}{\partial w_{sj}}$$

Реализация

Программа написана на языке C++ с использованием IDE Visual Studio 2017. Проект состоит из трех основных файлов:

- Main.cpp – задание параметров сети, обучение, проверка и вывод результатов.
- Fcnn.h – объявление класса нейронной сети.
- Fcnn.cpp – реализация класса нейронной сети.

Реализованы следующие методы сети:

- calculateAccuracy (…) – расчет ошибки в натренированной нейронной сети;
- train(…) – обучение сети с помощью метода обратного распространения ошибки;
- calculateOutputs(...) – расчет значений на выходе сети;
- calculateGradient (...) – расчет градиентов для обновления весов;
- correctWeights(...) – обновление весов сетки;
- backPropagation(...) – метод обратного распространения ошибки;
- shuffleSamples(...) – рандомизация порядка изображений;
- crossEntropy(...) – расчет величины кросс-энтропии.

Эксперименты

Число нейронов скрытого слоя	Количество эпох	Скорость обучения	Тренировочная точность	Тестовая точность
100	10	0.01	0.995650	0.975100
150	10	0.01	0.998367	0.979800
200	10	0.01	0.998417	0.979000
100	20	0.01	1.000000	0.979100
150	**20**	**0.01**	**1.000000**	**0.982200**
200	20	0.01	0.999983	0.981900
100	20	0.005	0.999633	0.979100
150	20	0.005	0.999500	0.980000
200	20	0.005	0.999750	0.981200