# **Лабораторная работа №1**

# **Задание**

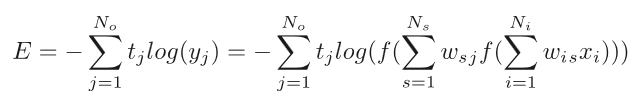
* Изучить метод обратного распространения ошибки;
* Вывести математические формулы для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов;
* Спроектировать и разработать программную реализацию;
* Подготовить отчет по проделанной работе.

# **Запуск решения**

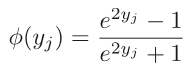
1. Открыть решение в VisualStudio 2017
2. Положить базу mnist в одноименную директорию внутри проекта
3. Собрать проект и запустить исполняемый файл

# **Метод обратного распространения ошибки**

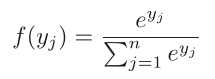
Ошибку вычисляем с использованием функции кросс-энтропии:



На скрытом слое в качестве функции активации используется гиперболический тангенс:



На выходном слое в качестве функции активации используется функция softmax:



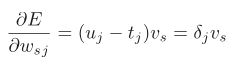
# **Алгоритм**

Инициализируем веса значениями от -0.5 до 0.5.

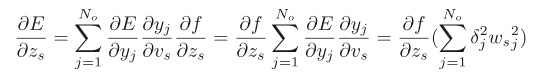
В цикле по количеству эпох:

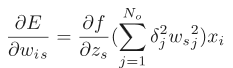
1. Перемешиваем порядок изображений.
2. В цикле по количеству изображений:
   1. В цикле по количеству скрытых нейронов:
      1. Считаем взвешенную сумму сигнала с первого слоя.
      2. Применяем функцию активации гиперболический тангенс.
   2. В цикле по количеству нейронов выходного слоя:
      1. Считаем взвешенную сумму сигнала со скрытого слоя.
   3. Применяем функцию активации softmax.
   4. Считаем градиенты функции ошибки

, , 









Так как используется гиперболический тангенс, то:

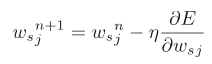


И тогда:

, 

* 1. Корректируем веса:





# **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 |