Пермский национальный исследовательский

Политехнический университет

Кафедра «Автоматика и телемеханика»

*Лабораторная работа*

*Реализация нейронных сетей для решения задачи распознавания образов.*

Выполнил: студенты гр. АТ3-16-1м

Соромотин К.А.

Томилин Т.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Пермь, 2017

***Цель работы***: Сформировать способность построения нейросетевой интеллектуальной системы для решения задач распознавания.

***Требования***:

1. Возможность добавления, удаления, изменения элементов обучающей выборки.
2. Возможность рисования образа мышью.
3. Распознавание хотя бы 8 фигур.
4. Программный код, хранящийся в системе контроля версий.
5. Код единообразно отформатирован.
6. Должны присутствовать комментарии.[1]

# **Ход работы**

***Алгоритм работы программы***:

|  |
| --- |
|  |
| ***Рис.1*** *Преобразование изображения* |

1. Входные изображения преобразуем в двумерный массив N\*M, в цикле пробегаемся по массиву и черно-белим её (рис.1).
2. Создаем нейронную сеть. Задаем количество нейронов на входе (ширина картинки\*на длину картинки), выходов (кол-во классов), скрытых слоев и нейронов на них.

3. Создаем обучающую выборку для каждого класса и обучаем сеть.

4. Тестируем программу подавая различные изображения на вход.

В качестве инструмента для разработки была выбрана среда Visual Studio и язык программирования C#.

Реализованы следующие классы:

* Layer – слой нейросети
* Neural –нейронная сеть

Класс Layer обладает следующими методами:

1. Инициализация весов
2. Задание кол-ва входов и выходов

Класс Neural обладает следующими методами:

1. Задание числа слоев нейросети.
2. Обучение нейросети (Алгоритм обратного распространения ошибки)
3. Запись в файл результатов обучения
4. Задание кол-ва слоев в сети

В качестве активационной функции использовали сигмоид.

|  |
| --- |
|  |
| ***Рис.2*** *Выражение задающая сигмоид* |

где α – параметр наклона сигмоидальной функции активации. Изменяя этот параметр, можно построить функции с различной крутизной.

|  |
| --- |
|  |
| ***Рис.3*** *График сигмоидальной функции* |

Монотонно возрастающая всюду дифференцируемая S-образная нелинейная функция с насыщением. Сигмоид позволяет усиливать слабые сигналы и не насыщаться от сильных сигналов.[3]

***Алгоритм обратного распространения ошибки***

Обучение сети включает в себя три стадии: подача на входы сети обучающих данных, обратное распространение ошибки и корректировка весов. Алгоритм обучения выглядит следующим образом:

* **Инициализация весов случайными небольшими значениями.** (0)
* До тех пор, пока условие прекращения работы алгоритма неверно, выполняются шаги 2-9. (1)
* Для каждой пары {данные, целевое значение} выполняются шаги 3-8. (2)
* **Распространение данных от входов к выходам**
* Каждый входной нейрон отправляет полученный сигнал всем нейронам в следующем слое (скрытом). (3)
* Каждый скрытый нейрон суммирует взвешенные входящие сигналы и применяет активационную функцию, вычисляя выходной сигнал (4)
* Каждый выходной нейрон суммирует взвешенные входящие сигналы и применяет активационную функцию, вычисляя выходной сигнал. (5)
* **Обратное распространение ошибки**
* Каждый выходной нейрон получает целевое значение – то выходное значение, которое является правильным для данного входного сигнала, и вычисляет ошибку, так же вычисляет величину, на которую изменится вес связи. Помимо этого, вычисляет величину корректировки смещения и посылает ошибку нейронам в предыдущем слое. (6)
* Каждый скрытый нейрон суммирует входящие ошибки (от нейронов в последующем слое) и вычисляет величину ошибки, умножая полученное значение на производную активационной функции: так же вычисляет величину, на которую изменится вес связи, помимо этого, вычисляет величину корректировки смещения. (7)
* **Изменение весов**
* Каждый выходной нейрон изменяет веса своих связей с элементом смещения и скрытыми нейронами. Каждый скрытый нейрон изменяет веса своих связей с элементом смещения и выходными нейронами. (8)
* **Проверка условия прекращения алгоритма**
* Условием прекращения работы алгоритма может быть, как достижение суммарной квадратичной ошибкой результата на выходе сети, предустановленного заранее минимума в ходе процесса обучения, так и выполнения определенного количества итераций алгоритма. В основе алгоритма лежит метод под названием градиентный спуск. В зависимости от знака, градиент функции (в данном случае значение функции — это ошибка, а параметры — это веса связей в сети) дает направление, в котором значения функции возрастают (или убывают) наиболее стремительно. (9) [2]

**Программная реализация**:

|  |
| --- |
| //X, Y - обучающая пара. kLern - скорость обучаемости  // Заполняем веса случайными числами  public void GenerateWeights()  {  Random rnd = new Random();  for (int i = 0; i < cX; i++)  {  for (int j = 0; j < cY; j++)  {  Weights[i, j] = rnd.NextDouble() - 0.5;  }  }  }  do{  // Вычисляем выход сети // Вычисляем значение активационной функции(сигмоида)  GetOUT(X);  // Заполняем дельта последнего слоя  for (int j = 0; j < Layers[countLayers - 1].countY; j++)  {  O = NETOUT[countLayers][j]; //Возвращает значение j-го слоя НС  DELTA[countLayers - 1][j] = (Y[j] - O) \* O \* (1 - O);// коэффициент изменения весов  }  // Перебираем все слои начиная споследнего  // изменяя веса и вычисляя дельта для скрытого слоя  for (int k = countLayers - 1; k >= 0; k--)  {  // Изменяем веса выходного слоя  for (int j = 0; j < Layers[k].countY; j++)  {  for (int i = 0; i < Layers[k].countX; i++)  {  Layers[k][i, j] += kLern \* DELTA[k][j] \* NETOUT[k][i];  }  }  if (k > 0)  {  // Вычисляем дельта слоя к-1  for (int j = 0; j < Layers[k - 1].countY; j++)  {  s = 0;  for (int i = 0; i < Layers[k].countY; i++)  {  s += Layers[k][j, i] \* DELTA[k][i];  }  DELTA[k - 1][j] = NETOUT[k][j] \* (1 - NETOUT[k][j]) \* s;  }  }  }  Error = CalcError(X, Y);// Возвращает ошибку  }while(Error<0.001) |
| ***Листинг 1.*** *Алгоритм обратного распространения ошибки* |

# Пример работы программы

|  |
| --- |
| https://pp.userapi.com/c830209/v830209763/1a505/NZcKSVvDSNM.jpg  ***Рис.4*** *Окно рисования и распознавания рисунка* |
| https://pp.userapi.com/c841036/v841036367/499f9/4jJLLX7BZ0U.jpg  ***Рис.5*** *Распознанный треугольник* |
| https://pp.userapi.com/c841036/v841036367/49a01/jh-XTlL-kqU.jpg  ***Рис.6*** *Распознанная окружность* |
| D:\Загрузки\Скриншоты\Скриншот 19-12-2017 011051.jpg  ***Рис.7*** *Распознанный прямоугольник* |

**Заключение**

В результате выполнения лабораторной работы была разработана нейронная сеть и её визуальное представление. Для обучения нашей нейронной сети был выбран алгоритм обратного распространения ошибки.

# **Литература**

1. Лабораторный практикум по интеллектуальным системам и технологиям: учеб. пособие. / Д.А. Истомин, В.Ю. Столбов. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2017. – 36 с.
2. Хабрахабр «Алгоритм обучения многослойной нейронной сети методом обратного распространения ошибки». [URL:](URL:%20%20http://www.cta.ru/cms/f/389393.pdf)  <https://habrahabr.ru/post/198268/>
3. AIportal «Функции активации в нейронных сетях». <http://www.aiportal.ru/articles/neural-networks/activation-function.html>