электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

Алгоритм автоматизированной генерации обучающей выборки для нейросетевого распознавателя рукописных символов.

77-48211/638244
09, сентябрь 2013
УДК: 681.5

Галкин В. А., Чернуха С. Н.

Россия, МГТУ им. Н.Э. Баумана
galkin@bmstu.ru

Введение

При практическом использовании распознавателя рукописных символов важны не только точность и быстродействие распознавания, но и наличие удобного механизма для первоначального обучения распознавателя и его корректировки. Как правило, нейронной сети недостаточно всего одного или двух примеров начертания каждого символа обучающей выборки для проявления высокой точности распознавания, а необходимость задания большего числа обучающих элементов является обременительной для пользователя.

Обучающая выборка — это конечное множество объектов, для которых известна их классовая принадлежность, используемое для обучения (тренировки) нейронной сети. При обучении с учителем элементы обучающей выборки подаются на вход нейронной сети. Выходной вектор подстраивается под некоторый заранее заданный целевой вектор путем изменения весов связей нейронной сети. Как показывает анализ, приведенный в [1,2,3], обучающая выборка не должна содержать противоречий, так как нейронная сеть однозначно сопоставляет выходные значения входным.

Далее рассмотрим зависимость точности распознавания символов от числа элементов обучающей выборки. Для проведения экспериментов будем исследовать способность сети распознавать цифры от 0 до 9.

Автоматизация генерации обучающей выборки

Исследование будем проводить с применением искусственной трехслойной нейронной сети с логарифмической сигмовидной функцией активации нейронов, обучаемой алгоритмом обратного распространения ошибки. Подробно этот алгоритм описан в [4] и реализован Чернухой С.Н в виде программы на КПК. В задаче on - line распознавания известен путь пера в виде последовательности точек, образующих символ. Для упрощения классификатора количество точек уменьшено до восьми, т.е. символ представляется в виде семи отрезков. Такая выборка делает расстояния между точками практически одинаковыми, что позволяет не учитывать при распознавании размер символа. Для инвариантности символа относительно его положения на изображении, преобразуем входную последовательность в вектор синусов и косинусов углов между осями координат и прямой, соединяющей две соседние точки. Таким образом, на входе нейронной сети имеем 14 значений синусов и косинусов углов значимых отрезков, являющихся нормированными (в диапазоне от -1 до 1), поэтому необходимость наличия обособленного входного слоя отсутствует. Число скрытых слоев равно двум. Для каждого символа формируется своя сеть с одним выходом. Значение на выходе интерпретируется как вероятность (вычисленная с точностью до 0,001) того, что символ, поданный на вход сети является тем символом, на котором была обучена данная сеть.

Зададим единственное образцовое начертание символа для каждой цифры и обучим нейронную сеть. Проверим точность распознавания цифры «4» с помощью разработанной Чернухой С.Н. программы.

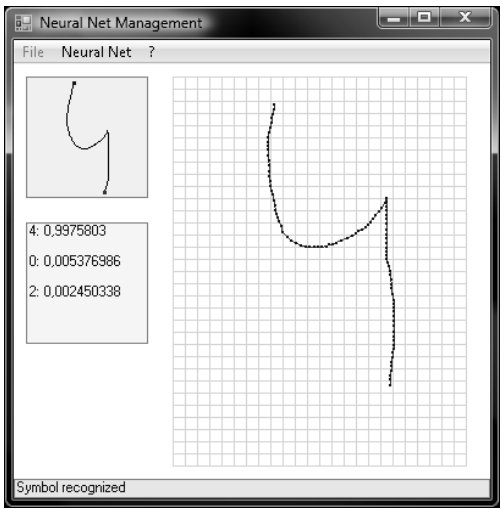

Рис. 1. Распознавание символа «4» из обучающей выборки.

На рис. 1 видно, что программа определяет введенное начертание как символ «4» с выходом 0,9975803, т.е. точность распознавания близка к 1.

Теперь введем символ «4» с небольшим наклоном правой составляющей линии относительно левой.

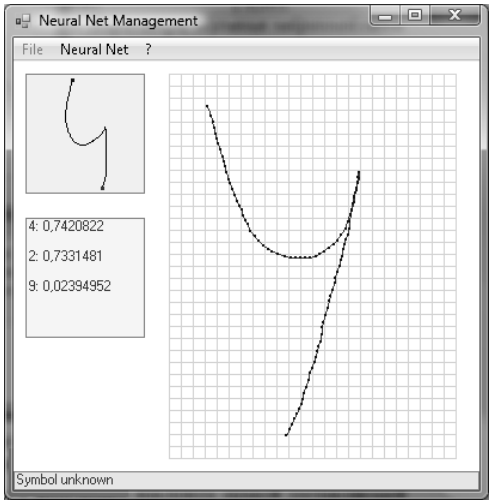

Рис. 2. Распознавание символа «4» с измененным начертанием.

Как видно из рис.2, точность распознавания снизилась до неприемлемого уровня - 0,742 .

Теперь введем символ «9» с небольшим отличием от его начертания в обучающей выборки.

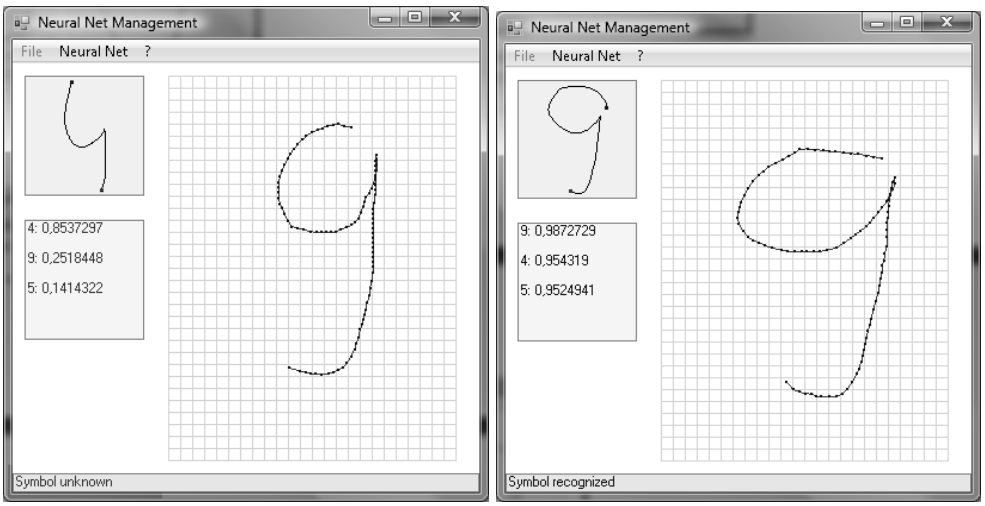

Рис. 3. Распознавание введенного символа «9» в сравнении с эталонным.

Нейронная сеть распознает введенный символ как цифру «4», хоть и с небольшой вероятностью, равной 0,854 (ниже порогового уровня, положенного равным 0,9).

Чтобы понять, почему распознаватель ведет себя подобным образом, необходимо рассмотреть построенные структуры весов нейронной сети для каждого символа, обратившись к файлам базы данных. Для каждого обученного символа формируются свой набор весов нейронной сети, который сохраняется в XML-файле [5].

Рассчитанные по формуле (1) суммы абсолютных величин входных весов первого скрытого слоя каждого из семи отрезков, которыми представляется символ «4» для нейронной сети, и отображены на графическом представлении этого символа (Рис.4).

$$W_n = \sum_{i=1}^{4} \left(\left| HWeight_{ij} \right| + \left| HWeight_{ij+1} \right| \right) = \sum_{i=1}^{4} \sum_{j}^{j+1} \left| HWeight_{ij} \right|, \qquad (1)$$

где $j = 2n - 1$, а $n = [1..7]$ – порядковый номер отрезка.

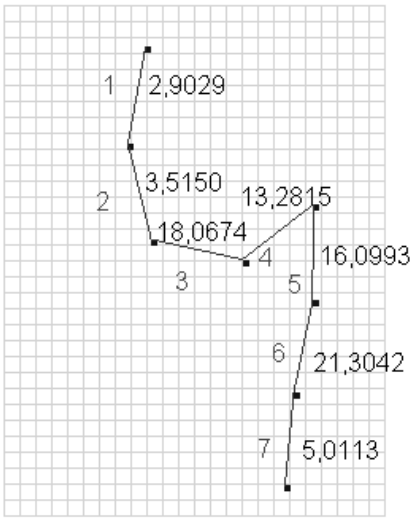

Рис. 4. Внутреннее представление символа «4» с весами отрезков.

Из рисунка 4 видно, что при обучении распознавания символа «4», отрезки, обозначенные номерами 3, 4, 5 и 6, имеют наибольший «удельный вес». Таким образом, если в распознаватель ввести символ, соответствующие отрезки которого точно совпадают с указанными выше, то нейронная сеть с большой вероятностью классифицирует введенный символ как «4». Примером такого начертания является контур символа «9», как показано на рис. 5.

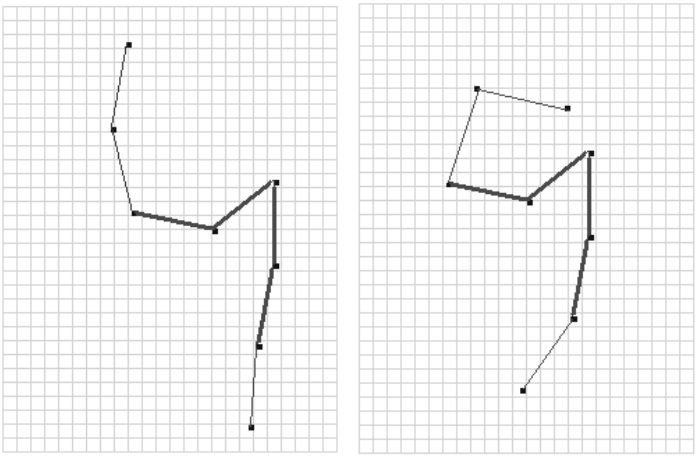

Рис. 5. Сходство начертаний «4» и «9» по значимым отрезкам.

Получается, что при обучении символа «4» нейронная сеть «ориентировалась» именно на эти практически одинаковые отрезки. Это произошло в результате исходного случайного задания весов — первого

этапа обучения сети. Веса распределились случайным образом таким образом, что алгоритм обучения с обратным распространением ошибки подгонял их так, что они стали чрезмерно большими относительно весов других отрезков, но при этом соблюдалась точность распознавания начертания исходного символа обучающей выборки.

Добавим в обучающую выборку, созданную ранее и состоящую из одного элемента на каждый символ, еще по одному элементу. После чего заново обучим сеть.

Вновь проверим точность распознавания, как это было сделано выше.

Введем символ «4» с небольшим отличием от эталонного начертания (как это было сделано ранее) — наклоном правой составляющей линии относительно левой (рис. 6).

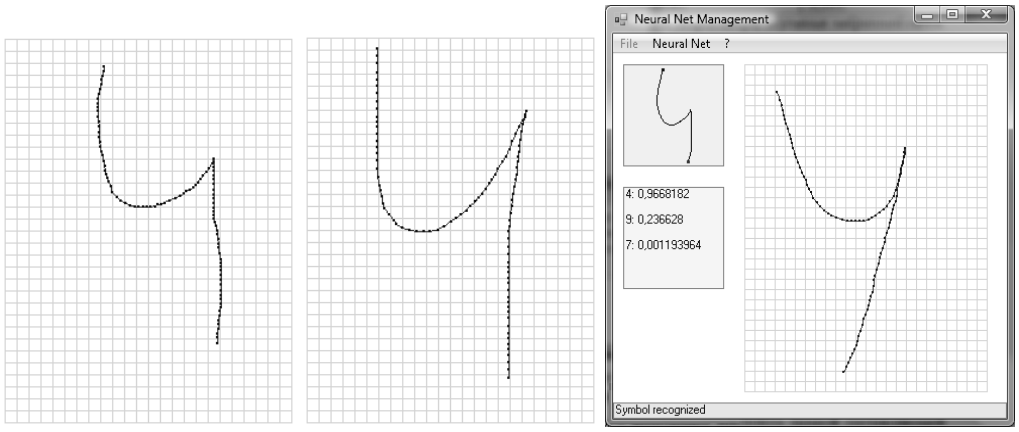

Рис. 6. Обучающая выборка и распознанный символ «4».

Точность распознавания немного снижена до 0,967, но остается на допустимом уровне.

Теперь введем символ «9» с небольшим отличием от его начертания в обучающей выборке, как это было проделано ранее.

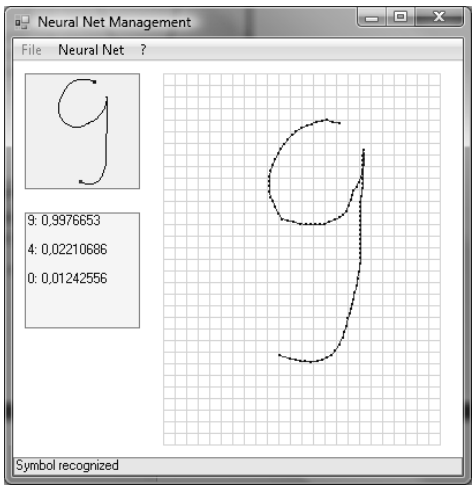

Рис. 7. Распознавание введенного символа «9».

Нейронная сеть уверенно и верно распознает введенный символ как цифру «9».

Вновь рассмотрим построенные структуры весов нейронной сети для символа «4», обратившись к файлам базы данных. Рассчитаем суммы абсолютных величин входных весов первого скрытого слоя каждого из семи отрезков по формуле (1), которыми представляется символ «4» для нейронной сети, и отобразим их на графическом представлении этого символа.

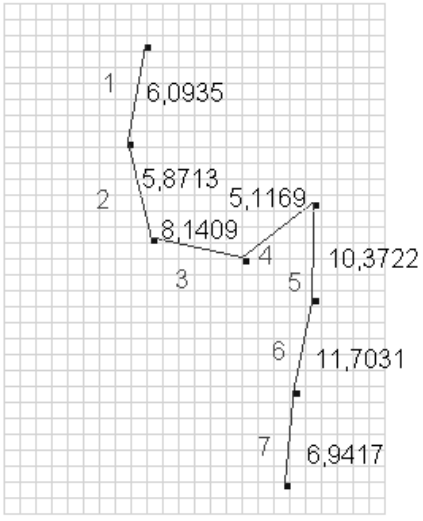

Рис. 8. Внутреннее представление символа «4» с весами отрезков.

Сравнивая рисунки 4 и 8, мы видим, что отрезки под номерами 3, 4, 5 и 6 более не имеет высокого «удельного веса». Абсолютные величины весов стали распределяться более равномерно. Теперь алгоритм обучения с обратным распространением ошибки не может обеспечить требуемый уровень погрешности распознавания, «ориентируясь» только лишь на отрезки 3–6 и поставив им большой вес, т.к. заданные два элемента обучающей выборки немного отличаются. Отныне при подстройке весов учитываются погрешности распознавания обоих элементов обучающей выборки. Процесс обучения происходит более равномерно.

При увеличении числа элементов обучающей выборки, адекватность поведения нейронной сети возрастает. Зависимость точности распознавания от числа элементов обучающей выборки представлена ниже.

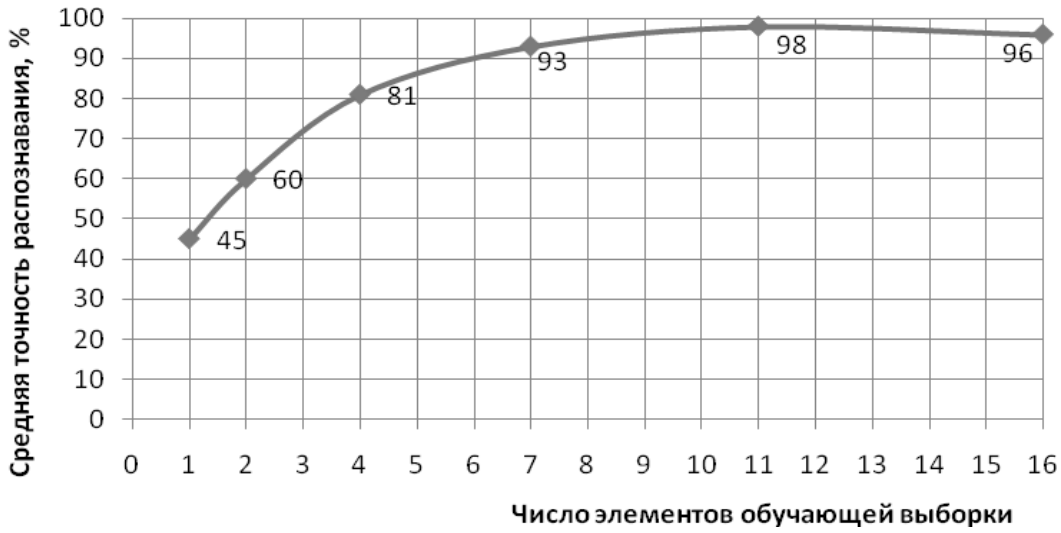

Рис. 9. Зависимость точности распознавания от числа элементов обучающей выборки.

Однако, надо понимать, что обучение и распознавание каждого символа происходит в рамках одного конечного множества вариантов написания каждого символа, причем такого, что отклонение начертания всех символов одного класса не превышает 15% (величины углов значимых отрезков с осями координат) от начертания единственного эталонного символа данного класса.

Из рис.9 видна необходимость задания достаточно большого числа элементов обучающей выборки для обеспечения высокой точности распознавания. Так, для уверенного распознавания цифр от 0 до 9 желательно вводить по 7 – 10 вариантов начертания каждого (в рамках одного класса, ибо мы не рассматриваем различные существующие формы написания цифр). Такая необходимость, как было показано выше, вызвана спецификой процесса обучения нейронной сети. Чтобы алгоритм обратного распространения не «акцентировался» на конкретном отрезке символа, нужно задавать несколько слегка различных вариантов начертания каждого символа. При этом веса будут подстраиваться равномерно и иметь одинаковый «удельный вес».

Человек – учитель нейронной сети, задавая по несколько вариантов начертания каждого символа, фактически накладывает «шумы» на форму эталонного символа. Именно это помогает алгоритму обратного распространения правильным образом подстраивать веса.

Получается, что человеку достаточно ввести всего один-два элемента в обучающую выборку, а остальное необходимое количество можно получить, накладывая на них «шум» — ту самую погрешность до 15% величины углов значимых отрезков.

Построим алгоритм, принимающий на вход массив направляющих синусов и косинусов значимых отрезков эталонного символа, введенного человеком, а так же значения минимального и максимального отклонения позиций крайних точек значимых отрезков символа в процентах в дробном виде. Выходным параметром алгоритма будет сгенерированный массив направляющих синусов и косинусов значимых отрезков с наложенным «шумом».

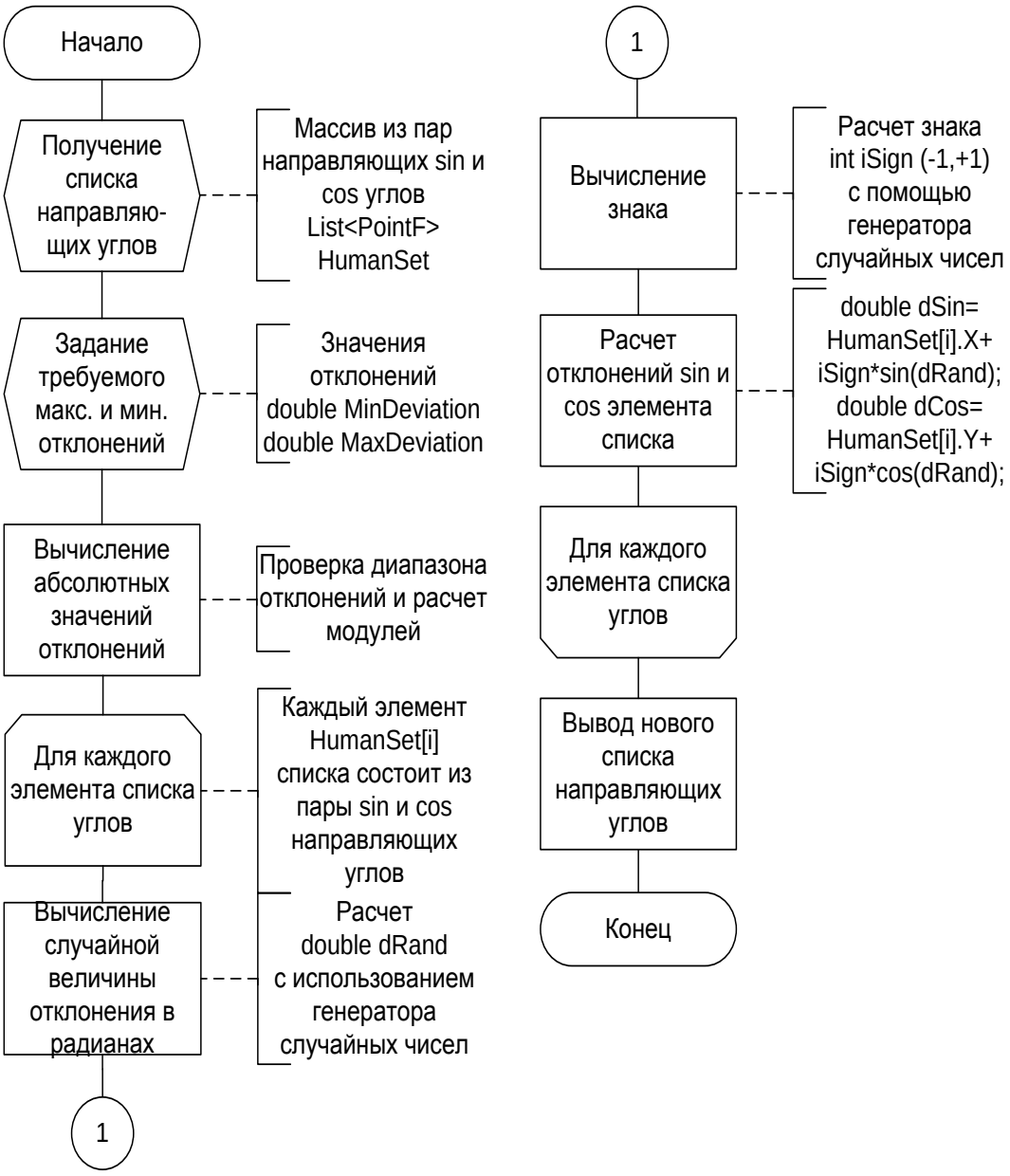

Рис. 10. Блок - схема алгоритма генерации «шума».

Реализуем алгоритм автоматизированной генерации обучающей выборки на языке программирования Microsoft C# [6].

```
public List<PointF> GenerateTrainingSet(List<PointF> HumanSet, double MinDeviation, double MaxDeviation)
{
MinDeviation = Math.Abs(MinDeviation);
MaxDeviation = Math.Abs(MaxDeviation);
//Значения отклонений должны быть в пределах [-1..1] (не более 100%)
if (MinDeviation > 1 || MaxDeviation > 1) throw new ArgumentException();
//Массив результирующего набора значений направляющих синусов и косинусов
```

```
List<PointF> AutoSet = new List<PointF>();
Random Generator = new Random();
foreach (PointF HumanPoint in HumanSet)
{
double dRand = Generator.NextDouble();
//Равномерное псевдослучайное распределение от MinDeviation до //MaxDeviation
dRand = MinDeviation + dRand / Math.Pow(MaxDeviation - MinDeviation, -1);
//Знак числа
int iSign = Generator.Next(0, 1)> 0 ? 1 : -1;
 //Величина отклонения в радианах
 double Deviation = 2 * Math.PI * dRand;

PointF AutoPoint = new PointF();
//sin
AutoPoint.X = HumanPoint.X + iSign * (float)Math.Sin(Deviation);
//cos
AutoPoint.Y = HumanPoint.Y + iSign * (float)Math.Cos(Deviation);
AutoSet.Add(AutoPoint);
}
 return AutoSet;
}
```

Очевидно, что задавать значения отклонений более 0,25 бессмысленно, т.к. это может привести к генерации слабо похожих на эталонную форм. В дальнейшем будут использованы показатели минимального и максимального отклонений, равные соответственно 0,05 и 0,15.

Для наглядности представим искусственно полученную обучающую выборку для символа «4» в графическом виде.

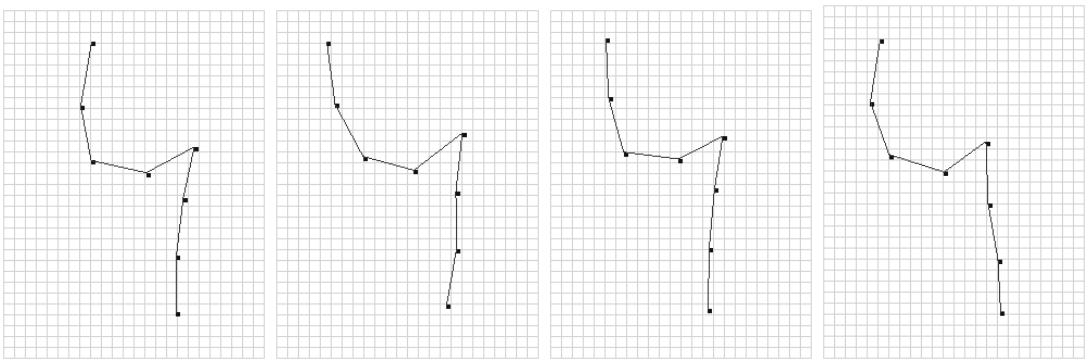

Рис. 11. Эталонный символ «4» слева и сгенерированные начертания.

Как видно, образы, сгенерированные ЭВМ сильно похожи на ручной ввод человека.

Зависимость точности распознавания от числа элементов обучающей выборки представлена ниже.

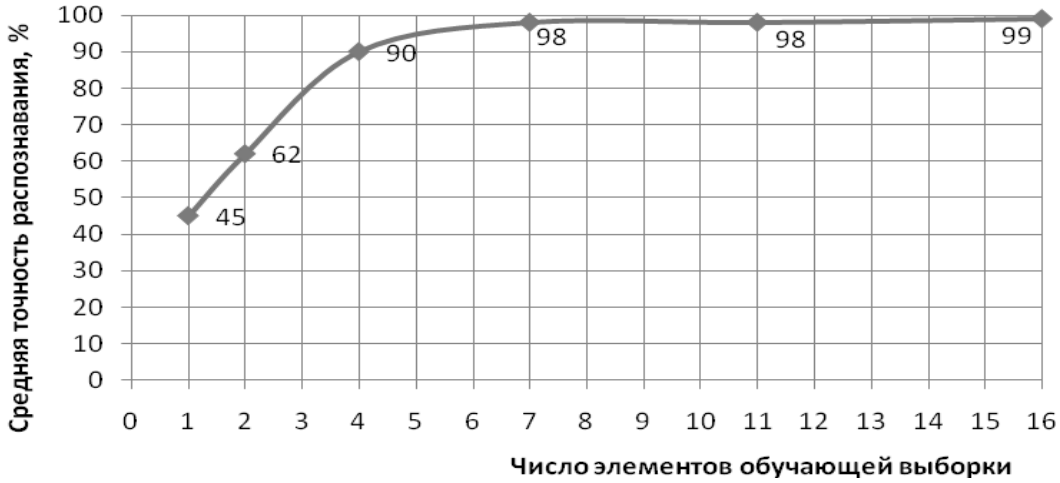

Рис. 12. Зависимость точности распознавания от числа элементов сгенерированной обучающей выборки.

Сравнивая график на рис. 12 с подобным на рис. 9, можно видеть, что нейронная сеть на искусственно сгенерированной выборке достигает приемлемого уровня погрешности распознавания, имея меньшее число элементов обучающей выборки нежели при их задании человеком–учителем. Это объясняется тем, что человеку свойственно повторяться при начертании символов. Т.е. как минимум два из десяти вариантов начертания получаются

очень схожими, практически неразличимыми, что по сути является одним элементом обучающей выборки. Алгоритм генерации же лишен такого недостатка (вероятность его проявления крайне мала), ибо основывается на генераторе псевдослучайных чисел ЭВМ.

Заключение.

В данной статье была показана необходимость задания большого числа элементов обучающей выборки при построении распознавателя рукописных символов на основе нейронной сети, обучаемой алгоритмом обратного распространения ошибки. При настройке такого распознавателя под свой почерк, пользователю сложно вводить по 7–10 вариантов для каждого из символов, которые в дальнейшем планируется распознавать. Поэтому был предложен алгоритм, позволяющий на основе одного эталонного начертания символа, введенного пользователем, сгенерировать все необходимые элементы обучающей выборки для данного символа. По результатам исследования можно сделать вывод, что предложенный алгоритм полностью справляется с поставленной задачей, обеспечивая генерацию обучающей выборки на основе единственного эталонного символа, введенного человеком, что избавляет человека-учителя от необходимости затрачивать большое количество времени на ввод обучающей выборки вручную. Причем выборка, полученная с помощью разработанного алгоритма, обеспечивает лучшие результаты распознавания при меньшем объеме обучающей выборки, что положительно сказывается на времени обучения распознавателя, т.к. позволяет сократить число элементов обучающей выборки.

Литература.

1. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. Пер. с англ. – М.: Мир, 1992. – 240 с.

2. Пупков К. А., Коньков В. Г. Интеллектуальные системы (Исследование и создание). - М. : Изд-во МГТУ им. Н. Э. Баумана, 2003. - 345 с. - Библиогр.: с. 336-345. - ISBN 5-7038-2038-3.
3. Девятков В. В. Системы искусственного интеллекта : учеб. пособие для вузов / - М. : Изд-во МГТУ им. Н. Э. Баумана, 2001. - 350 с. - (Информатика в техническом университете). - Библиогр.: с. 346. - ISBN 5-7038-1727-7.
4. Rumelhart D. E., Hinton G. E., Williams R. J. 1986. Learning internal reprentations by error propagation. In Parallel distributed processing, vol. 1, pp. 318-62. Cambridge, MA: MIT Press.
5. Кэгл К. XML. Пер. с англ.- М.: Лори, 2006. – 656 с.
6. Галисеев Г.В. Программирование на языке C# 2005.- М.: Диалектика, 2006. – 368 с.