МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ
Институт системной и программной инженерии
и информационных технологий (Институт СПИНТех)

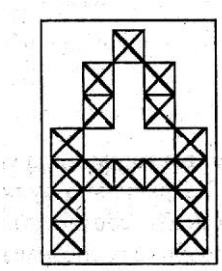
Лабораторный практикум по курсу "Нейронные сети"

Лабораторная работа 4.
Нейронно-сетевое распознавание изображений (символов) [1].

Часто полезно иметь устройство, которое выполняет распознавание образов. В частности, очень эффективны и выгодны машины, которые могут читать символы. Машина, которая читает банковские чеки, может выполнять за то же самое время намного больше проверок, чем человек. Этот вид приложений сохраняет время и деньги, а также устраняет условия, при которых человек выполняет монотонную, периодически повторяющуюся работу. Данная работа иллюстрирует, как распознавание символов может быть выполнено с помощью нейронной сети.

1. Постановка задачи

Требуется создать нейронную сеть для распознавания 26 символов латинского алфавита. В качестве датчика предполагается использовать систему распознавания, которая выполняет оцифровку каждого символа, находящегося в поле зрения. В результате каждый символ будет представлен шаблоном размера 5x7. Например, символ *A* может быть представлен, как это показано на рис. 1, *а* и *б.*



а

0	0	1	0	0
0	1	0	1	0
0	1	0	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

б

Рис.1. Представление символа *A*

Однако система считывания символов обычно работает неидеально и отдельные элементы символов могут оказаться искаженными (Рис. 2).

[1] Лабораторная работа подготовлена совместно с доцентом кафедры биомедицинских систем МИЭТ Ю.П. Маслобоевым.

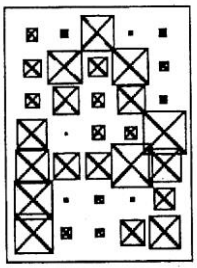


Рис.2. Представление символа *A* при наличии шума

Проектируемая нейронная сеть должна точно распознавать идеальные векторы входа и с максимальной точностью воспроизводить зашумленные векторы. М-функция *prprob* определяет 26 векторов входа, каждый из которых содержит 35 элементов, этот массив называется *алфавитом.* М-функция формирует выходные переменные *alphabet* и *targets*, которые определяют массивы алфавита и целевых векторов. Для того чтобы восстановить шаблон для *i*-й буквы алфавита, надо выполнить следующие операторы:

```
[alphabet, targets] = prprob;
ti = alphabet(:,i);
letter{i} = reshape(ti, 5, 7)';
letter{i}
```

2. Нейронная сеть

На вход сети поступает вектор входа с 35 элементами; вектор выхода содержит 26 элементов, только один из которых равен 1, а остальные - 0. Правильно функционирующая сеть должна ответить вектором со значением 1 для элемента, соответствующего номеру символа в алфавите. Кроме того, сеть должна быть способной распознавать символы в условиях действия шума. Предполагается, что шум - это случайная величина со средним значением 0 и стандартным отклонением, меньшим или равным 0.2.

2.1. Архитектура сети

Для работы нейронной сети требуется 35 входов и 26 нейронов в выходном слое. Для решения задачи выберем двухслойную нейронную сеть с логарифмическими сигмоидными функциями активации в каждом слое. Такая функция активации выбрана потому, что диапазон выходных сигналов для этой функции определен от 0 до 1, и этого достаточно, чтобы сформировать значения выходного вектора. Структурная схема такой нейронной сети показана на рис. 3.

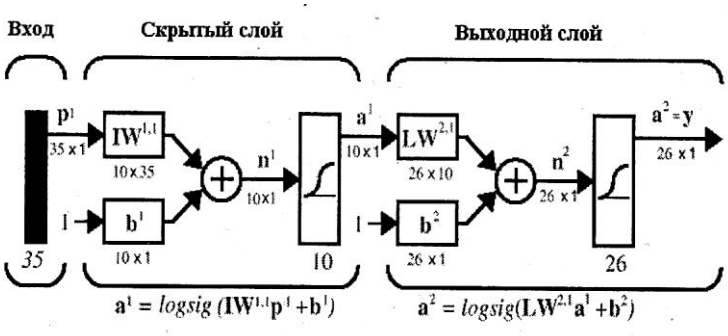


Рис. 3. Архитектура нейронной сети

Скрытый слой имеет 10 нейронов. Такое число нейронов выбрано на основе опыта и разумных предположений. Если при обучении сети возникнут затруднения, то можно увеличить количество нейронов этого уровня. Сеть обучается так, чтобы сформировать единицу в единственном элементе вектора выхода, позиция которого соответствует номеру символа, и заполнить остальную часть вектора нулями. Однако наличие шумов может приводить к тому, что сеть не будет формировать вектора выхода, состоящего точно из единиц и нулей. Поэтому по завершении этапа обучения выходной сигнал обрабатывается М-функцией *compet*, которая присваивает значение 1 единственному элементу вектора выхода, а всем остальным - значение 0.

2.2. Инициализация сети

Вызовем М-файл *prprob*, который формирует массив векторов входа *alphabet* размера 35x26 с шаблонами символов алфавита и массив целевых векторов *targets*:

```
[alphabet,targets]=prprob;
[R,Q]   =  size(alphabet);
[S2,Q]   =  size(targets);
```

Двухслойная нейронная сеть создается с помощью команды *newff*:

```
S1  =   10;
net = newff(minmax(alphabet),[S1 …
S2],{‘logsig’,’logsig'},'traingdx’);
net.LW{2,1) = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;
```

Указание: Возможна произвольная модификация двухслойной нейронной сети, в частности применение вместо алгоритма обучения, основанного на использовании метода градиентного спуска с адаптивной скоростью обучения, задаваемого функцией `traingdx`, алгоритма обучения Левенберга-Марквардта (функция `trainlm`). Возможны переустановка, например, `net.b{1} = net.b{1}*0.01` и просмотр, например, `net.b{2}` значений весов и смещений В этом случае команда будет выглядеть следующим образом:

```
S1  =   10;
net = newff(minmax(alphabet),[S1 …
S2],{‘logsig’,’logsig'},'trainlm’);
net.LW{2,1) = net.LW{2,1}*0.01;
net.b{1} = net.b{1}*0.01;
net.b{2} = net.b{2}*0.01;
```

3. Обучение нейронной сети

Чтобы создать нейронную сеть, которая может обрабатывать зашумленные векторы входа, следует выполнить обучение нейронной сети как на идеальных, так и на зашумленных векторах. Сначала сеть обучается на идеальных векторах, пока не будет обеспечена минимальная сумма квадратов погрешностей. Затем сеть обучается на 10 наборах идеальных и зашумленных векторов. Две копии свободного от шума алфавита

используются для того, чтобы сохранить способность сети классифицировать идеальные векторы входа. К сожалению, после того, как описанная выше сеть обучилась классифицировать сильно зашумленные векторы, она потеряла способность правильно классифицировать некоторые векторы, свободные от шума. Следовательно, сеть снова надо обучить на идеальных векторах. Это гарантирует, что сеть будет работать правильно, когда на ее вход будет передан идеальный символ. Обучение выполняется с помощью функции *trainbpx*, которая реализует метод обратного распространения ошибки с возмущением и адаптацией параметра скорости настройки.

3.1. Обучение в отсутствие шума

Сеть первоначально обучается в отсутствие шума с максимальным числом циклов обучения 5000, либо до достижения допустимой средней квадратичной погрешности, равной 0.1.

```
P = alphabet;
T = targets;
net.performFcn = 'sse';
net.trainParam.goal = 0.1;
net.trainParam.show =20;
net.trainParam.epochs= 5000;
net.trainParam.mc = 0.95;
[net,tr] = train(net,P,T);
```

3.2. Обучение в присутствии шума

Чтобы спроектировать нейронную сеть, не чувствительную к воздействию шума, обучим ее с применением двух идеальных и двух зашумленных копий векторов алфавита. Целевые векторы состоят из четырех копий векторов. Зашумленные векторы имеют шум со средним значением 0.1 и 0.2. Это обучает нейронную сеть правильно распознавать зашумленные символы и в то же время хорошо распознавать идеальные векторы.

При обучении с шумом максимальное число циклов обучения сократим до 300, а допустимую погрешность увеличим до 0.6:

```
netn = net;
netn.trainParam.goal = 0.6;
netn.trainParam.epochs =300;
T =  [targets  targets targets  targets];
for pass  =  1:10
P  =   [alphabet,   alphabet, …
(alphabet  +  randn(R,Q) *0.1), …
(alphabet  +  randn(R,Q) *0.2) ];
[netn,tr]   =  train (netn, P,T);
end;
```

3.3. Повторное обучение в отсутствие шума

Поскольку нейронная сеть обучалась в присутствии шума, то имеет смысл повторить ее обучение без шума, чтобы гарантировать, что идеальные векторы входа классифицируются

правильно.

```
netn.trainParam.goal = 0.1;
% Предельная среднеквадратичная погрешность
netn.trainParam.epochs = 500;
% Максимальное количество циклов обучения
net.trainParam.show = 5 ;
% Частота вывода результатов на экран
[netn,tr]   =  train(netn,P,T);
```

4. Эффективность функционирования системы

Эффективность нейронной сети будем оценивать следующим образом. Рассмотрим 2 структуры нейронной сети: сеть 1, обученную на идеальных последовательностях, и сеть 2, обученную на зашумленных последовательностях. Проверка функционирования производится на 100 векторах входа при различных уровнях шума.

Приведем фрагмент сценария *appcrl*, который выполняет эти операции:

```
noise_range = 0:.05:.5;
max_test =100;
network1 = [];
network2 = [];
T = targets;

% Выполнить тест
for noiselevel = noise_range;
errors1=0;
errors2=0;

for i=1:max_test
P = alphabet + randn(35,26) *noiselevel;

% Тест для сети 1
A = sim(net,P);
AA = compet(A);
errorsl = errorsl + sum(sum(abs (AA-T)))/2;

% Тест для сети 2
An = sim(netn,P);
AAn = compet(An);
errors2 = errors2 + sum(sum(abs(AAn-T)))/2;
echo off
end

% Средние значения ошибок (100 последовательностей
% из 26 векторов целей)
networkl = [networkl errors1/26/100];
```

```
network2 = [network2 errors2/26/100];
end
```

Тестирование реализуется следующим образом. Шум со средним значением 0 и стандартным отклонением от 0 до 0.5 с шагом 0.05 добавляется к векторам входа. Для каждого уровня шума формируется 100 зашумленных последовательностей для каждого символа и вычисляется выход сети. Выходной сигнал обрабатывается М-функцией *compet* с той целью, чтобы выбрать только один из 26 элементов вектора выхода. После этого оценивается количество ошибочных классификаций и вычисляется процент ошибки.

Соответствующий график погрешности сети от уровня входного шума:

```
plot(noise_range,network1*100,'—',noise_range,network2*100);
```

Сеть 1 обучена на идеальных векторах входа, а сеть 2 - на зашумленных. Обучение сети на зашумленных векторах входа значительно снижает погрешность распознавания реальных векторов входа. Сети имеют очень малые погрешности, если среднеквадратичное значение шума находится в пределах от 0.00 до 0.05. Когда к векторам был добавлен шум со среднеквадратичным значением 0.2, в обеих сетях начали возникать заметные ошибки. При этом погрешности нейронной сети, обученной на зашумленных векторах, на 3-4 % ниже, чем для сети, обученной на идеальных входных последовательностях.

Если необходима более высокая точность распознавания, сеть может быть обучена либо в течение более длительного времени, либо с использованием большего количества нейронов в скрытом слое. Можно также увеличить размер векторов, чтобы пользоваться шаблоном с более мелкой сеткой, например 10x14 точек вместо 5x7.

Проверим работу нейронной сети для распознавания символов. Сформируем зашумленный вектор входа для символа *J* (Рис. 5):

```
noisyJ = alphabet(:,10)+randn(35,1)*0.2;
plotchar(noisyJ);
% Зашумленный символ J
```

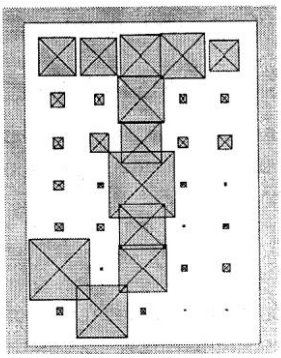


Рис. 4. Представление символа *J* при наличии шума

```
A2 = sim(net,noisyJ);
A2 = compet(A2);
answer = find(compet(A2)) ==1
plotchar(alphabet(:,answer));
```

Нейронная сеть выделила 10 правильных элементов и восстановила символ J без

ошибок (Рис. 6).

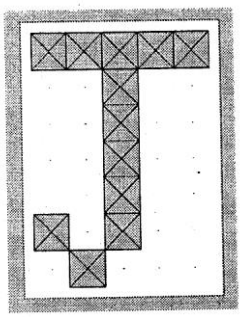


Рис. 5. Представление восстановленного символа *J*

Эта работа демонстрирует, как может быть разработана простая система распознавания изображений. Заметим, что процесс обучения не состоял из единственного обращения к обучающей функции. Сеть была обучена несколько раз при различных векторах входа. Обучение сети на различных наборах зашумленных векторов позволило обучить сеть работать с изображениями, искаженными шумами, что характерно для реальной практики.

Задание.

1. Увеличьте количество нейронов в скрытом слое, а также добавьте еще один слой. Обучите полученную сеть и сравните результат обучения с результатом обучения исходной сети. Сравните погрешности распознавания при различных уровнях шума.
2. Изучите содержание демонстрационного файла <mr_1.m>.
3. Напишите m-процедуру распознавания букв русского алфавита, соответствующих гласным звукам, с помощью обученного 3-х слойного перцептрона.
4. Представьте полный отчет о выполненной работе, поясните отдельные этапы работы и полученные результаты.