Применение пакета NNT MATLAB для построения нейронных классификаторов

Пакет Neural Networks Toolbox (Нейронные сети) содержит средства для проектирования, моделирования, обучения и использования множества известных парадигм аппарата искусственных нейронных сетей (ИНС): от базовых моделей персептрона до самых современных ассоциативных и самоорганизующихся сетей. Пакет может быть использован для решения множества разнообразных задач, таких как обработка сигналов, нелинейное управление, финансовое моделирование и т. п.

Для каждого типа архитектуры и обучающего алгоритма ИНС имеются функции инициализации, обучения, адаптации, создания, моделирования, демонстрации, а также примеры применения.

Общие сведения о MATLAB

Среда MATLAB – это набор инструментов и приспособлений, с которыми работает пользователь или программист MATLAB. Она включает в себя средства для управления переменными в рабочем пространстве MATLAB, вводом и выводом данных, а также создания, контроля и отладки M-файлов и приложений MATLAB.

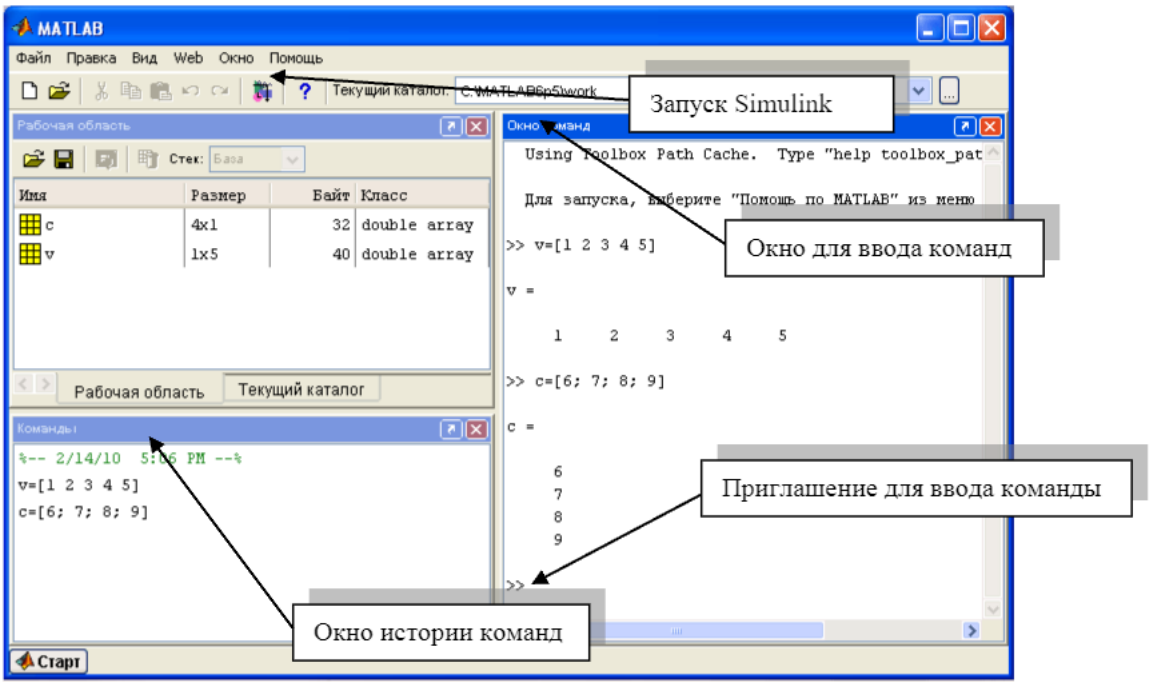


Рис.1 Окно MATLAB

Пакет Neural Network Toolbox

Первый шаг при работе с нейронными сетями – это создание модели сети. Для разбора процедур адаптации и обучения будем руководствоваться моделями однослойной линейной сети и многослойной сети прямой передачи. Их создание заключается в следующем...

Создание сети многослойной сети

Для создания сетей с прямой передачей сигнала в ППП NNT предназначена функция **ncwff**. Она имеет 4 входных аргумента и 1 выходной аргумент – объект класса network.

Сеть прямой передачи FF – NEWFF

net = newff(PR, [S1 S2...SNI], {TF1 TF2...TFNI}, btf, blf, pf)

Функция **newff** предназначена для создания многослойных нейронных сетей прямой передачи сигнала с заданными функциями обучения и настройки, которые используют

Входные аргументы:

PR – массив размера Rx2 минимальных и максимальных значений для R векторов входа;

Si – количество нейронов в слое i;

TFi – функция активации слоя i, по умолчанию tansig;

btf – обучающая функция, реализующая метод обратного распространения, по умолчанию trainlm;

blf – функция настройки, реализующая метод обратного распространения, по умолчанию learngdm;

pf – критерий качества обучения, по умолчанию mse.

Выходные аргументы:

net – объект класса network object многослойной нейронной сети.

Первый входной аргумент – это массив размера Rx2, содержащий допустимые границы значений (минимальное и максимальное) для каждого из R элементов вектора входа;

второй – массив для задания количества нейронов каждого слоя;

третий – массив ячеек, содержащий имена функций активации для каждого слоя; четвертый - имя функции обучения.

Например, следующий оператор создает сеть с прямой передачей сигнала:

net = newff([-l 2; 0 5], [3,1], {'tansig','purelin'}, 'traingd');

Эта сеть использует 1 вектор входа с двумя элементами, имеющими допустимые границы значений [-1 2] и [0 5]; имеет 2 слоя с тремя нейронами в первом слое и одним нейроном во втором слое; используемые функции активации: tansig-в первом слое, purelin - во втором слое; используемая функция обучения - traingd.

М-функция **newff** не только создает архитектуру сети, но и инициализирует ее веса и смещения, подготавливая нейронную сеть к обучению. Однако существуют ситуации, когда требуется специальная процедура инициализации сети.

Также, приступая к процедуре обучения и адаптации, необходимо обмолвиться и о функциях оценки качества обучения.

Функции оценки качества обучения

Процесс обучения нейронных сетей связан с такой настройкой ее весов и смещений чтобы минимизировать некоторый функционал, зависящий от ошибок сети, т.е. разности между желаемым и реальным сигналами на ее выходе. В качестве таких функционалов в ППП NNT используются: сумма квадратов ошибок

$$SSE = \sum_{i=1}^{N} e_i^2;$$

средняя квадратичная ошибка

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i^2;$$

комбинированная ошибка

$$MSEREG = \frac{pp}{N}\sum_{i=1}^{N} e_i^2 + \frac{1-pp}{n}\sum_{j=1}^{n} x_j^2;$$

средняя абсолютная ошибка

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |e|.$$

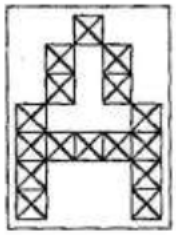
В дальнейшем будет более подробно рассмотрено применение и настройка этих функций.

Реализация многослойной искусственной нейронной сети

Распознавание образов – одна из основных задач, возлагаемая на ИНС. Рассмотрим на примере распознавания символов латинского алфавита реализацию и обучение ИНС.

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

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

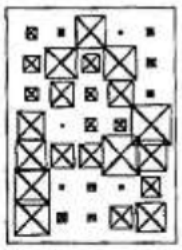


Рис. 2

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 формируем выходные переменные **alphabet** и **targets**, которые определяют массивы алфавита и целевых векторов. Массив targets определяется как **eye(26)**. 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}
```

Пример.

Определим шаблон для символа A, который является первым элементом алфавита:

```
[alphabet, targets] = prprob;
i = 2;
ti = alphabet(:, i);
letter{i} = reshape(ti, 5, 7)';
letter{i}
ans =
     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
```

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.

Архитектура сети

Для работы нейронной сети требуется 35 входов и 26 нейронов в выходном слос. Для решения задачи выберем двухслойную нейронную сеть с логарифмическими сигмоидальными 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.

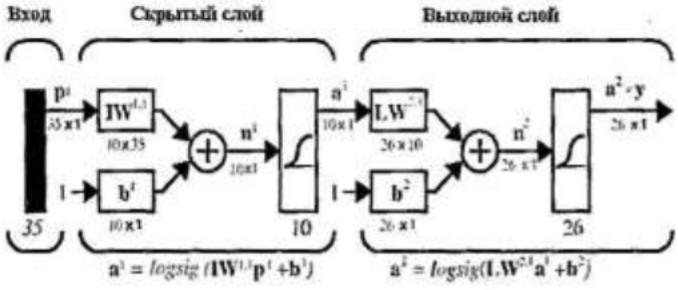


Рис. 3

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.

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;
```

Структура нейронной сети представлена на рис. 4.

```
gensim(net)
```

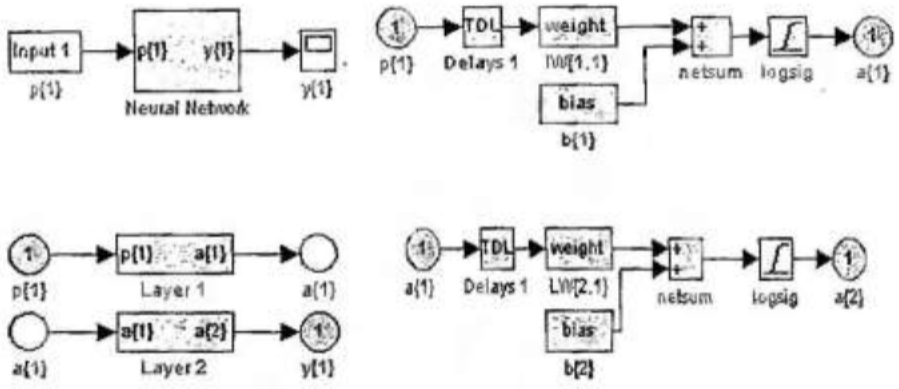


Рис. 4

Обучение

Чтобы создать нейронную сеть, которая может обрабатывать зашумленные векторы входа, следует выполнить обучение 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 сожа-

лению, 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 (backpropagation) с возмущением и адаптацией параметра скорости настройки.

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 (рис. 5):

```
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); % Рис.9.23
```

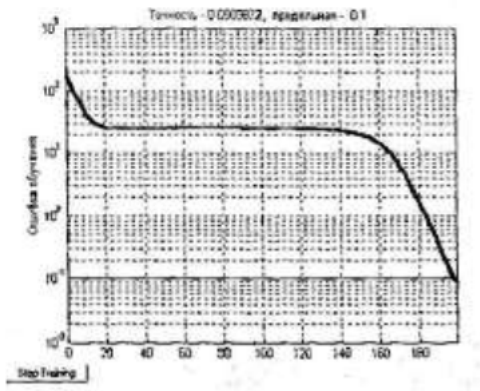


Рис. 5

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 нейрон 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 (рис. 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 % Рис.9.24
```

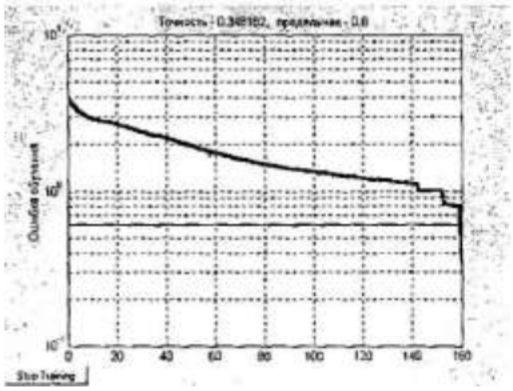


Рис. 6

Повторное обучение в отсутствие шума

Поскольку нейронная сеть обучалась в присутствии шума, то имеет смысл повторить се 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);
```

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 apper1, 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);
    errors1 = errors1 + 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
  network1 = [network1 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

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 показан на рис. 7.

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

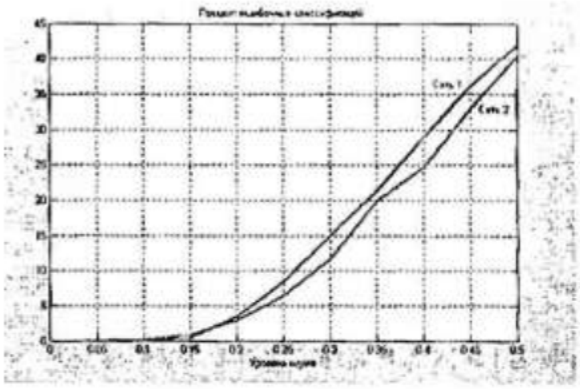


Рис. 7

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. 8):

```
A2 = sim(net,noisyJ);
A2 = compet(A2);
answer = find(compet(A2) == 1)
answer =    10
plotchar(alphabet(:,answer)); % Распознанный символ J
```

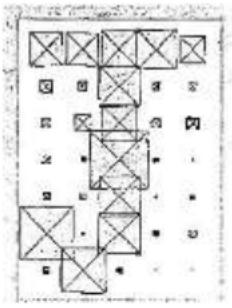


Рис. 8

Нейронная сеть выделила 10 правильных элементов и восстановила символ **J** без ошибок (рис. 9).

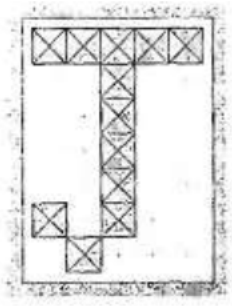


Рис. 9

Эта задача 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.