ПРИМЕР СОЗДАНИЯ ОДНОНАПРАВЛЕННОЙ СЕТИ

Цель занятия – продемонстрировать основные этапы реализации нейронно-сетевого подхода для решения конкретной задачи. Можно выделить 4 основных этапа:
1. Подготовка данных для тренировки сети.
2. Создание сети.
3. Обучение сети.
4. Тестирование сети.
5. Моделирование сети. (Использование сети для решения поставленной задачи.)

В качестве примера рассмотрим следующую задачу:

Задан массив, состоящий из нескольких значений функции $y = Ce^{\left(-\frac{(x-A)^2}{S}\right)}$ (S>0) на интервале (0,1). Создать нейронную сеть такую, что при вводе этих значений на входы сети на выходах получались бы значения параметров C, A и S.

1. Подготовка данных для обучения сети

В первую очередь необходимо определиться с размерностью входного массива.
Выберем количество значений функции равным N=21, т.е. в качестве входных векторов массива используем значения функции y в точках x=0.05; …1.0. Для обучения сети необходимо сформировать массив входных векторов для различных наборов параметров C, A и S. Каждый набор этих параметров является вектором-эталоном для соответствующего входного вектора.

Для подготовки входного и эталонного массивов воспользуемся следующим алгоритмом. Выбираем случайным образом значения компонент вектора – эталона C, A и S и вычисляем компоненты соответствующего входного вектора. Повторяем эту процедуру M раз и получаем массив входных векторов в виде матрицы размерностью NxM и массив векторов – эталонов в виде матрицы размерностью в нашем случае 3xM. Полученные массивы мы можем использовать для обучения сети.

Прежде чем приступать к формированию обучающих массивов необходимо определиться с некоторыми свойствами массивов.

1. Диапазон изменения параметров C,A S. Выберем диапазоны изменения параметров C,A,S равными (0.1, 1). Значения, близкие к 0 и сам 0 исключим в связи с тем, что функция не определена при S=0. Второе ограничение связано с тем, что при использовании типичных передаточных функций желательно, чтобы компоненты входных и выходных векторов не выходили за пределы диапазона (-1,1). В дальнейшем мы познакомимся с

методами нормировки, которые позволяют обойти это ограничение.
2. Количество входных и эталонных векторов выберем равным M=100. Этого достаточно для обучения, а процесс обучения не займет много времени.

Тестовые массивы и эталоны подготовим с помощью программы *mas1*:

```
% формирование входных массивов (входной массив P) и (эталоны T)

P=zeros(100,21);
T=zeros(3,100);
x=0:5.e-2:1;
for i=1:100
c=0.9*rand+0.1;
a=0.9*rand+0.1;
s=0.9*rand+0.1;
T(1,i)=c;
T(2,i)=a;
T(3,i)=s;
P(i,:)=c*exp(-((x-a).^2/s));
end;
P=P';
```

С помощью этой программы формируется матрица P из M=100 входных векторов-столбцов, каждый из которых сформирован из 21 точки исходной функции со случайно выбранными значениями параметров C,A,S, и матрица T эталонов из 100 эталонных векторов-столбцов, каждый из которых сформирован из 3 соответствующих эталонных значений. Матрицы P и T будут использованы при обучении сети. Следует отметить, что при каждом новом запуске этой программы будут формироваться массивы с новыми значениями компонентов векторов, как входных, так и эталонных.

2. Создание сети

Вообще, выбор архитектуры сети для решения конкретной задачи основывается на опыте разработчика. Поэтому предложенная ниже архитектура сети является одним вариантом из множества возможных конфигураций.
Для решения поставленной задачи сформируем трехслойную сеть обратного распространения, включающую 21 нейрон во входном слое (по числу компонент входного вектора) с передаточной функцией logsig, 15 нейронов во втором слое с передаточной функцией logsig и 3 нейрона в выходном слое (по числу компонентов выходного вектора) с передаточной функцией purelin. При этом в качестве обучающего алгоритма выбран алгоритм Levenberg-Marquardt (trainlm). Этот алгоритм обеспечивает быстрое обучение, но требует много ресурсов. В случае, если для реализации этого алгоритма не хватит оперативной

памяти, можно использовать другие алгоритмы (trainbfg, trainrp, trainscg, traincgb, traincgf, traincgp, trainoss, traingdx). По умолчанию используется trainlm. Указанная сеть формируется с помощью процедуры:

```
net=newff(minmax(P),[21,15,3],{'logsig' 'logsig' 'purelin'},'trainlm');
```

Первый аргумент - матрица Mx2 минимальных и максимальных значений компонент входных векторов вычисляется с помощью процедуры minmax.
Результатом выполнения процедуры newff является объект – нейронная сеть net заданной конфигурации. Сеть можно сохранить на диске в виде mat. файла с помощью команды save и загрузить снова с помощью команды load. Более подробную информацию о процедуре можно получить, воспользовавшись командой help.

3. Обучение сети

Следующий шаг – обучение созданной сети. Перед обучением необходимо задать параметры обучения. Задаем функцию оценки функционирования sse.

```
net.performFcn='sse';
```

В этом случае в качестве оценки вычисляется сумма квадратичных отклонений выходов сети от эталонов. Задаем критерий окончания обучения – значение отклонения, при котором обучение будет считаться законченным:

```
net.trainParam.goal=0.01;
```

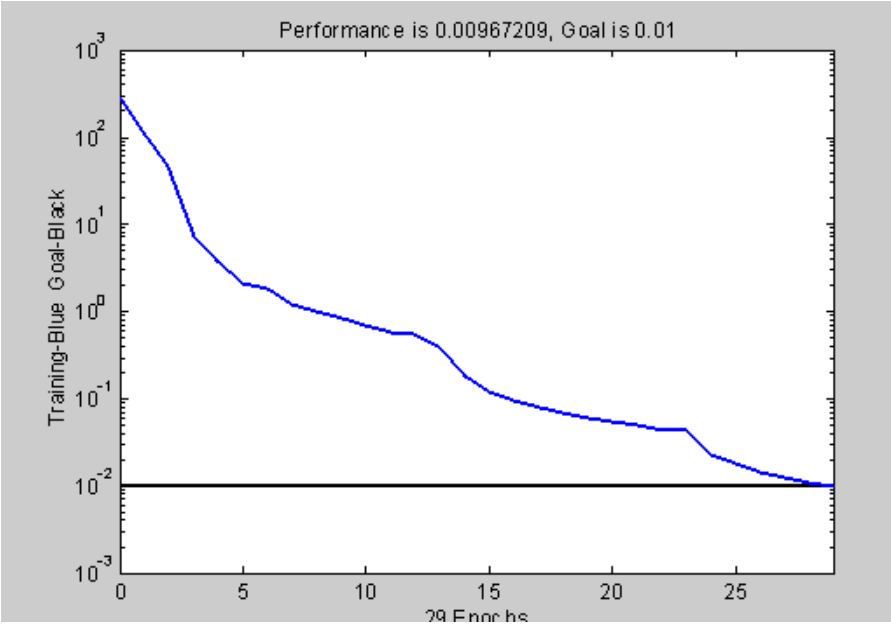
Задаем максимальное количество циклов обучения. После того, как будет выполнено это количество циклов, обучение будет завершено:

```
net.trainParam.epochs=1000;
```

Теперь можно начинать обучение:

```
[net,tr]=train(net,P,T);
```

Процесс обучения иллюстрируется графиком зависимости оценки функционирования от номера цикла обучения.



Таким образом, обучение сети окончено. Теперь эту сеть можно сохранить в файле nn1.mat:

save nn1 net;

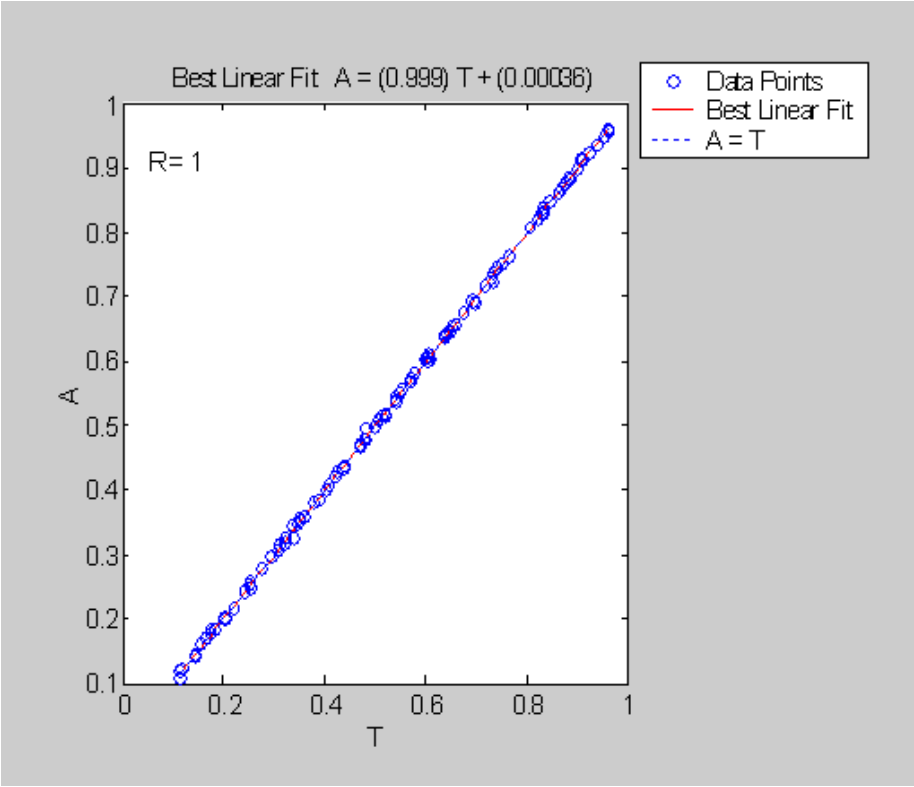
4. Тестирование сети

Перед тем, как воспользоваться нейронной сетью, необходимо исследовать степень достоверности результатов вычислений сети на тестовом массиве входных векторов. В качестве тестового массива необходимо использовать массив, компоненты которого отличаются от компонентов массива, использованного для обучения. В нашем случае для получения тестового массива достаточно воспользоваться еще раз программой mas1.

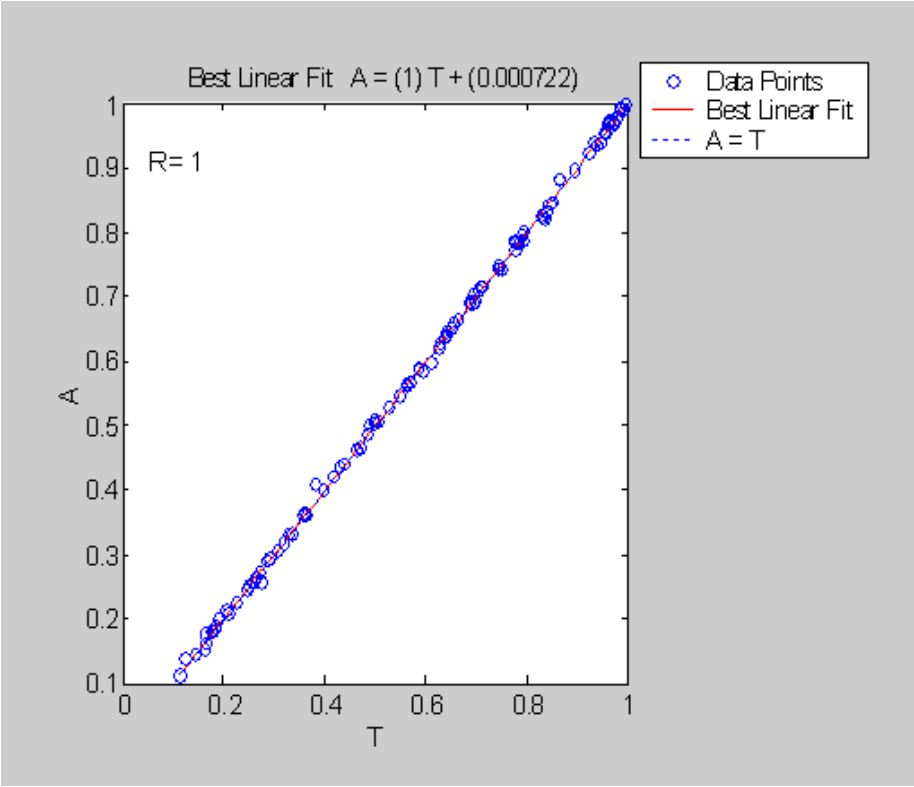
Для оценки достоверности результатов работы сети можно воспользоваться результатами регрессионного анализа, полученными при сравнении эталонных значений со значениями, полученными на выходе сети когда на вход поданы входные векторы тестового массива. В среде MATLAB для этого можно воспользоваться функцией postreg. Следующий набор команд иллюстрирует описанную процедуру:

>> mas1 - создание тестового массива P
>> y=sim(net,P); - обработка тестового массива
>> [m,b,r]=postreg(y(1,:),T(1,:)); - регрессионный анализ результатов обработки.

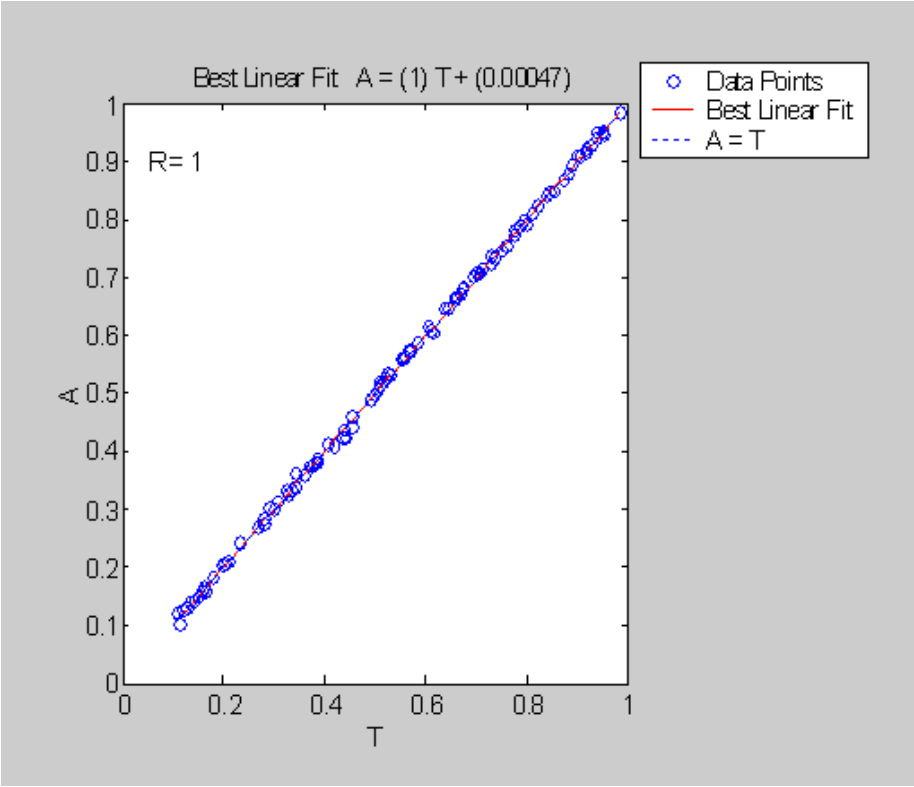
Сравнение компонентов C эталонных векторов с соответствующими компонентами выходных векторов сети. Видно, что все точки легли на прямую, что говорит о правильной работе сети на тестовом массиве.




```
>> [m,b,r]=postreg(y(2,:),T(2,:));
```




```
>> [m,b,r]=postreg(y(3,:),T(3,:));
```



Как видно из рисунков, наша сеть отлично решает поставленную задачу для всех трех выходных параметров. Сохраним обученную сеть net на диске в файл nn1.mat

save nn1 net

5. Моделирование сети. (Использование сети для решения поставленной задачи)

Для того, чтобы применить обученную сеть для обработки данных, необходимо воспользоваться функцией sim:
Y=sim(net,p);
где p – набор входных векторов, Y – результат анализа в виде набора выходных векторов. Например, пусть C=0.2, A=0.8, S=0.7, тогда
p=0.2*exp(-((x-0.8).^2/0.7));

Подставив этот входной вектор в качестве аргумента функции sim:

Y=sim(net,p)

Получим:
Y =
0.2048 (C)
0.8150 (A)
0.7048 (S)
>>
Что весьма близко к правильному результату (0.2; 0.8; 07).