Нейронные сети с MATLAB.
3 курс, 6 семестр
доцент кафедры ДУиСА Голубева Л.Л.

Лабораторная работа "Исследование нейронных сетей прямого распространения сигнала и обратного распространения ошибки"

Цель работы: изучение модели нейрона и архитектуры многослойной нейронной сети прямого распространения сигнала, алгоритмов обратного распространения ошибки; создание, обучение и исследование моделей нейронных ff-сетей в системе MATLAB для различных областей применения: аппроксимации произвольных функций с конечным числом точек разрыва, классификации, распознавания символов.

Теоретические сведения

Сеть, не имеющая обратных связей (*feedback*), называется ***сетью с прямой передачей сигнала*** (***feed forward***). Такие сети часто имеют один или более скрытых слоев нейронов с сигмоидальными функциями активации, в то время как выходной слой содержит нейроны с линейными функциями активации.

`Back-propagation` (**`backpropagation`**, **`bp`**) – обратное распространение (обратная передача) ошибки обучения нейронной сети, обратная связь при обучении нейронной сети. Большинство методов обучения основано на вычислении градиента функционала ошибки по настраиваемым параметрам. Все эти методы можно разделить на три класса:

а) **методы нулевого порядка**, в которых для нахождения минимума используется только информация о значениях функционала в заданных точках;

б) **методы первого порядка**, в которых используется градиент функционала ошибки по настраиваемым параметрам, использующий частные производные функционала;

в) **методы второго порядка**, в которых используются вторые производные функционала ошибки.

Градиентные алгоритмы обучения:

- **`GD`** – алгоритм градиентного спуска (**`learngd`**, **`traingd`**);
- **`GDM`** – алгоритм градиентного спуска с возмущением (**`learngdm`**, **`traingdm`**);
- **`GDA`** – алгоритм градиентного спуска с выбором параметра скорости настройки (**`traingda`**);
- **`GDX`** – алгоритм градиентного спуска с возмущением и адаптацией параметра скорости настройки (**`traingda`**);
- **`Rprop`** – пороговый алгоритм обратного распространения ошибки (**`trainp`**).

Алгоритмы **метода сопряженных градиентов**:

- **`CGF`** – алгоритм Флетчера–Ривса (**`traincgf`**);
- **`CGP`** – алгоритм Полака–Ребейры (**`traincgp`**);
- **`CGB`** – алгоритм Биеле–Пауэлла (**`traincgb`**);
- **`SCG`** – алгоритм Молера (**`trainscg`**).

Квазиньютоновские алгоритмы:

- **`BFGS`** – алгоритм Бройдена, Флетчера, Гольдфарба и Шанно (**`trainbfg`**);
- **`OSS`** – одношаговый алгоритм метода секущих плоскостей (алгоритм Баттини) (**`trainoss`**);
- **`LM`** – алгоритм Левенберга–Марквардта (**`trainlm`**);
- **`BR`** – алгоритм Левенберга–Марквардта с регуляризацией по Байесу (**`trainbr`**).

Модель нейрона

Нейрон ff-сети с одним *R*-элементным вектором входа показан на рис.1. Здесь каждый элемент входа p_1, p_2,... p_R умножается на веса $w_{1,1}$, $w_{1,2}$, ... $w_{1,R}$ соответственно и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектор-строки **W** на вектор входа **p**. Нейрон имеет смещение *b*, которое суммируется с взвешенной суммой входов **Wp**. Результирующая сумма *n* равна

$$n=w_{1,1}p_1+w_{1,2}p_2+...+w_{1,R}p_R+b=\mathbf{W}*\mathbf{p}+b.$$

и служит аргументом функции активации *f*.

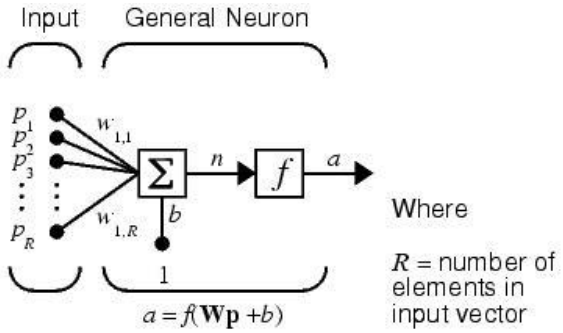
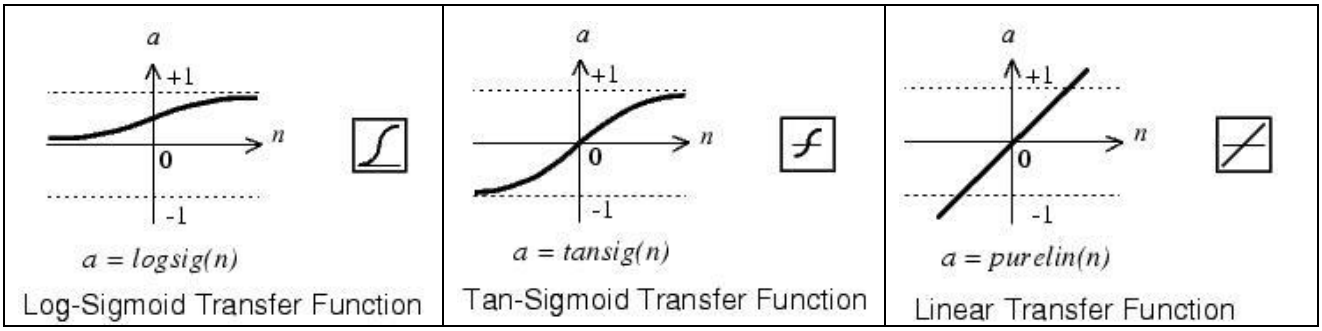


Рис.1. Модель нейрона

Нейроны могут использовать любую ***дифференцируемую*** функцию активации *f*, чтобы сгенерировать свои выходы. В качестве функций активации часто используются сигмоидальные функции **`logsig`**, **`tansig`** или линейные функции **`purelin`**.



Если выходной слой многослойной сети использует сигмоидальные нейроны, то выходы сети ограничены малым диапазоном (от 0 до 1 или от -1 до 1). Если же используются линейные нейроны, то выходы сети могут принимать любые значения.

Виды сетей прямого распространения сигнала

Last used in R2010a NNET 6.0.4.:

- **`newff`** – сеть прямой передачи сигнала и обратного распространения ошибки (*feed-forward backpropagation network*);
- **`newcf`** – сеть каскадной прямой передачи сигнала и обратного распространения ошибки (*cascade-forward backpropagation network*);
- **`newfftd`** – динамическая сеть прямой передачи сигнала и обратного распространения ошибки (*feed-forward input-delay backpropagation network*).

Current used:

- **`feedforwardnet`** – сеть прямой передачи сигнала и обратного распространения ошибки (*feed-forward backpropagation network*);
  - **`fitnet`** – fitting network (аппроксимация кривой);
  - **`patternnet`** – pattern recognition network (распознавание образов);
- **`cascadeforwardnet`** – сеть каскадной прямой передачи сигнала и обратного распространения ошибки (*cascade-forward backpropagation network*);

Способы создания

- **`nftool`** – 2-слойные сети прямой передачи сигнала: **`fitnet`**;
- **`nprtool`** – 2-слойные сети прямой передачи сигнала: **`patternnet`**;
- **`nntool`** – 2-слойные сети прямой передачи сигнала: **`newff`**, **`newcf`**, **`newfftd`**;
- командное окно – N-слойные сети прямой передачи сигнала: **`feedforwardnet`**, **`fitnet`**, **`patternnet`**, **`cascadeforwardnet`**.

Архитектура сети прямого распространения сигнала

Сети с прямой передачей сигнала часто имеют один или более скрытых слоев нейронов с сигмоидальными функциями активации, в то время как выходной слой содержит нейроны с линейными функциями активации.

Для многослойных сетей номер слоя определяется верхним индексом весовых матриц. Соответствующая нотация используется в двухслойной **`tansig`**/**`purelin`** сети, показанной на рис.2.

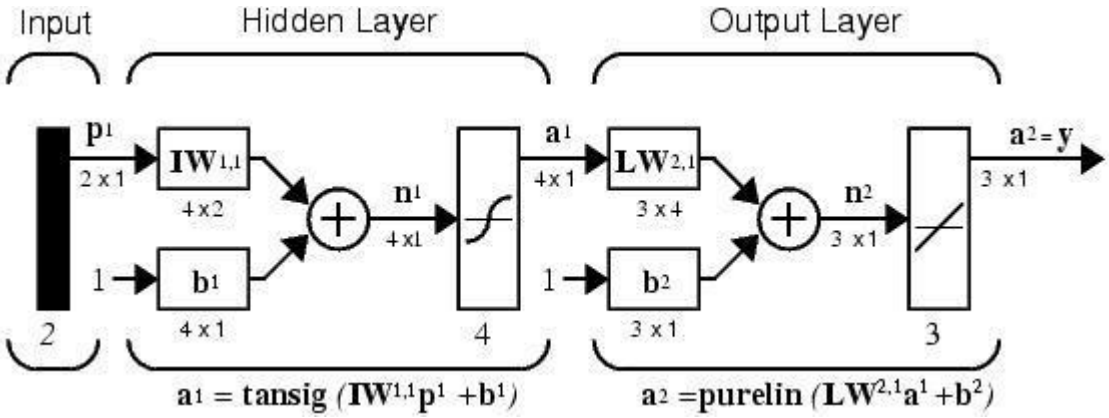


Рис.2. Архитектура сети с прямой передачей сигнала

Сети с такой архитектурой могут воспроизводить весьма сложные нелинейные зависимости между входом и выходом сети.

Назначение

- аппроксимация любых функций с конечным числом разрывов;
- распознавание образов;
- классификация объектов.

Моделирование сетей прямого распространения средствами MATLAB

Сеть прямой передачи сигнала и обратного распространения ошибки (feedforward backpropagation network). Функция **`newff`** предназначена для создания сетей прямой передачи сигнала с заданными функциями обучения и настройки, использующими метод обратного распространения ошибки.

Сеть состоит из N слоев, использует весовую функцию **dotprod**, функцию входа сети **netsum** и специальные функции активации. В сети реализуется прямая передача сигнала. Первый слой имеет веса, идущие от входов. Каждый следующий слой имеет веса, идущие от предыдущего слоя. Все слои имеют смещения. Последний слой является выходным. Веса и смещения каждого слоя инициализируются при помощи функции **initnw**.

net = newff(P,T,[S^1 S^2...$S^{(N-1)}$],{TF^1 TF^2...TF^N}, BTF, BLF, PF, IPF, OPF, DDF)

где:

P – матрица размерности $R \times Q^1$, состоящая из Q^1 образцов R-элементных векторов входа (один столбец – один вектор входа);

T – матрица размерности $S^N \times Q^2$, состоящая из Q^2 образцов S^N-элементных векторов выхода(целей), (один столбец – один вектор цели);

$\mathbf{S}^i$ – размер i-го слоя, i=1…N-1, по умолчанию =[ ]. Размер S^N выходного слоя определяется из количества строк второго входного аргумента T;

$\mathbf{TF}^i$ – функция активации i-го слоя, по умолчанию ='**tansig**' для скрытых слоев и '**purelin**' для выходного слоя;

BTF – обучающая функция, реализующая метод обратного распространения ошибки, по умолчанию ='**trainlm**';

BLF – функция настройки весов/смещений, реализующая метод обратного распространения, по умолчанию = '**learngdm**';

PF – критерий качества обучения, по умолчанию = '**mse**';

IPF – массив ячеек функций обработки векторов входа (Default = {'**fixunknowns**', '**removeconstantrows**', '**mapminmax**'})

OPF – массив ячеек функций обработки выхода. (Default = {'**removeconstantrows**', '**mapminmax**'})

DDF – функция деления обучающего множества (default = '**dividerand**')

и возвращает объект класса **network** с архитектурой N-слойной ***feedforward backpropagation сети***.

Каскадная сеть прямой передачи сигнала и обратного распространения ошибки (**cascade-forward backpropagation network**). Функция **newcf** предназначена для создания ***каскадных*** сетей прямой передачи сигнала с заданными функциями обучения и настройки, использующими метод обратного распространения ошибки. В сети реализуется прямая передача сигнала, но не последовательная, а каскадная. Первый слой имеет веса, идущие от входов. Каждый следующий слой имеет веса, идущие от входов и всех предыдущих слоев.

net = newcf(P,T,[S^1 S^2...$S^{(N-1)}$],{TF^1 TF^2...TF^{N1}}, BTF, BLF, PF, IPF, OPF, DDF)

Входные параметры функции **newcf** совпадают с входными параметрами функции **newff**.

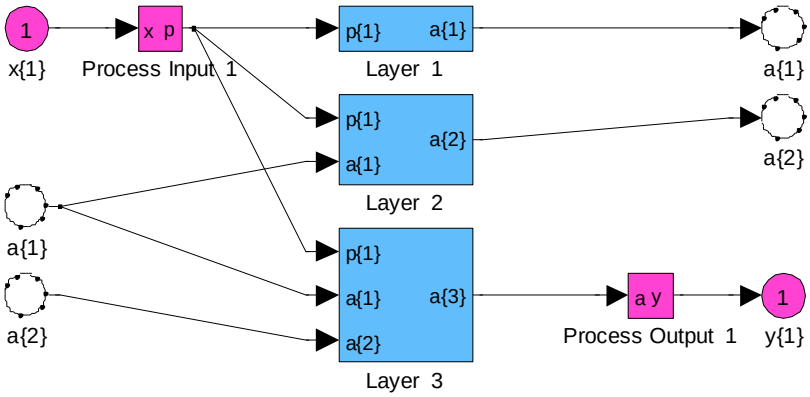


Рис.3. 3-слойная каскадная сеть с прямой передачей сигнала и обратным распространением ошибки

Динамическая сеть прямой передачи сигнала (***feedforward input time-delay back-propagation network***). Функция **newfftd** предназначена для создания динамических сетей прямой передачи сигнала с заданными функциями обучения и настройки, использующими метод обратного распространения ошибки и ***имеющими линию задержки на входе сети***.

```
net = newfftd(P,T,ID,[S1 S2...S(N-1)],{TF1 TF2...TFNl}, BTF, BLF, PF,
              IPF, OPF, DDF)
```

ID – Input delay vector;

Все остальные входные параметры функции **newfftd** совпадают с входными параметрами функции **newff**.

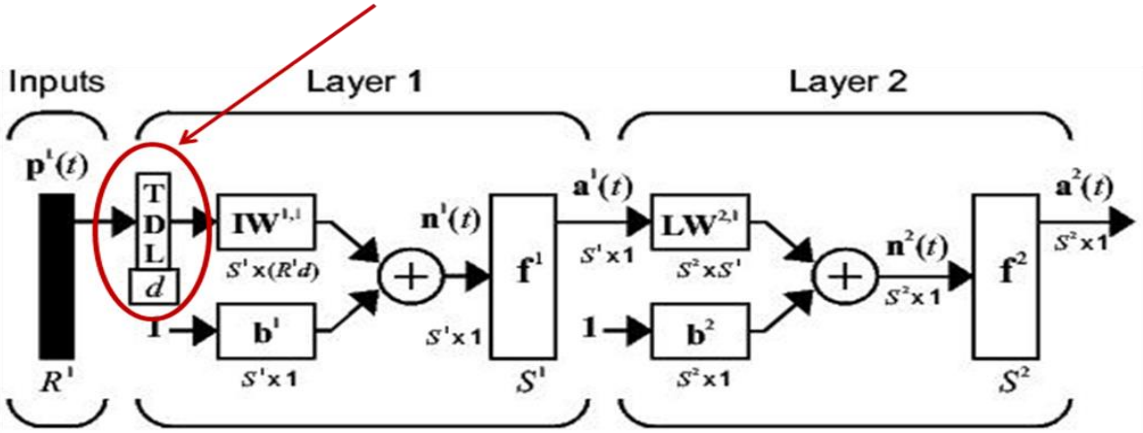


Рис.4. Динамическая сеть прямой передачи сигнала

Примеры

Пример 1. **Аппроксимация функции одной переменной**. Создать ff-нейронную сеть, чтобы обеспечить следующее отображение последовательности входа P в последовательность целей T:

```
P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4];
net = newff(P,T,5);
```

Архитектура нейронной сети: двухслойная (numLayers=2) сеть с одним входом (**numInputs=1**) с прямой передачей сигнала (**layerConnect=[0 0; 1 0]**); первый скрытый слой содержит 5 нейронов с функцией активации **tansig**; второй слой – один нейрон с

функцией активации **purelin**. Выход сети совпадает с выходом слоя 2 (**outputConnect=[0 1]**).

Обучим сеть в течение 50 циклов.

```
net.trainParam.epochs = 50;
net = train(net,P,T);
```

Характеристика точности обучения показана на рис.5; установившаяся среднеквадратичная ошибка составляет приблизительно 0.0064.

Выполним моделирование обученной сети на обучающей последовательности входа P:

```
Y = sim(net,P)
plot(P,T,'b',P,T,'b*',P,Y1,'ok',P,Y,'hr','LineWidth',2)
legend('Source data','..','Before train','After train',-1)
```

Результаты моделирования показаны на рис.6.

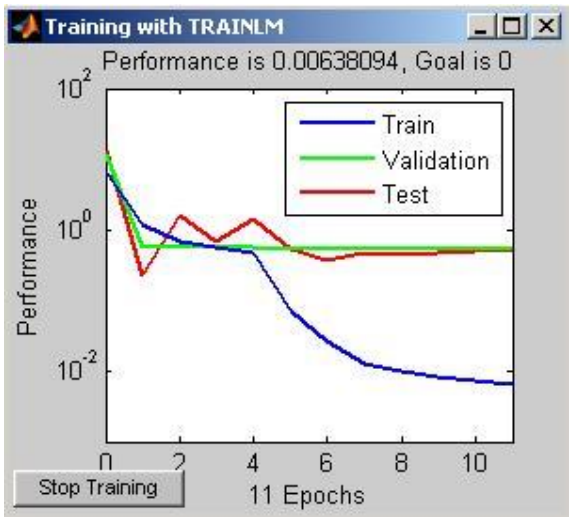


Рис.5. Характеристики проведения обучения

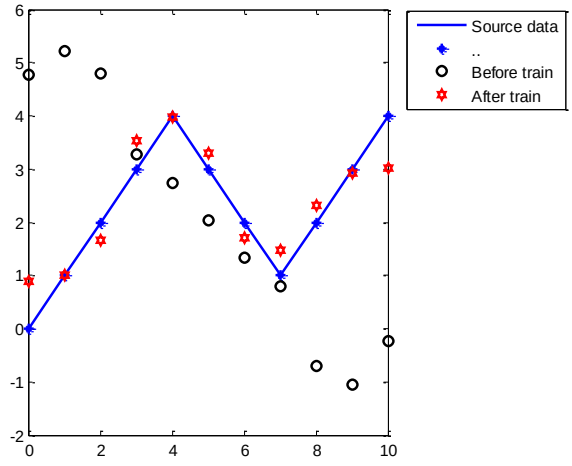


Рис.6. Результаты моделирования

Пример 2. Аппроксимация функции одной переменной с шумом. Создать и обучить нейронную сеть, которая находила бы значение функции по заданному аргументу.

Рассмотрим зашумленную функцию одной переменной y=x×cos(x). Функция задана таблицей координат точек.

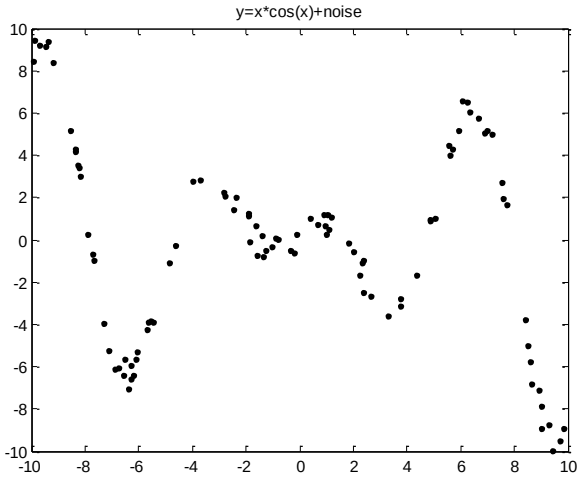


Рис.7. Обучающее множество

Для этой задачи используем однонаправленную сеть с обратным распространением ошибок, сигмоидальной функцией передачи в скрытом слое и линейной функций передачи в выходном слое. Такая структура подходит для аппроксимации функций.

Создать такую нейронную сеть можно тремя способами: с помощью функций командной строки, используя GUI nntool, GUI nftool. Рассмотрим первый из них.

Создадим исходные значения функции. Точки будем задавать неупорядоченно.

```
L=100;
noise=rand(1,L)*1.5-0.75;
P=rand(1,L)*20-10;
[X,i]=sort(P);
Y=P.*cos(P);
T=Y+noise;
plot(P,T,'.k'); title('y=x*cos(x)+noise');
```

Создадим ff-сеть с обратным распространением ошибок, сигмоидальной функцией передачи в скрытом слое и линейной функций передачи в выходном слое. Количество нейронов в скрытом слое зависит от сложности решаемой задачи, в данном случае аппроксимируемой функции. Обучение будем производить, используя алгоритм Левенберга-Маркардта (Levenberg-Marquardt), который реализует функция **trainlm**. Функция ошибки – **mse**.

```
net=newff(P,T,[13]); %,{'tansig','purelin'},'trainlm');
net.trainFcn
ans =
   trainlm
net.performFcn
ans =
   mse
```

Входные и целевые векторы в случайном порядке делятся на 3 набора: 60% векторов используются для обучения (**training subset**), 20% - для проверки достоверности результатов и чтобы избежать переобучения сети (**validation subset**), 20% - для независимого испытания сети (**testing subset**).

```
net.divideFcn
ans =
dividerand
```

```
net.divideParam
ans =
    trainRatio: 0.6000
      valRatio: 0.2000
     testRatio: 0.2000
```

Выполним обучение сети.

```
% train
net.trainParam.epochs = 500;
net.trainParam.show = 25;
net.trainParam.goal = 0;
[net,tr] = train(net,P,T);
```

Структура **tr** содержит информацию о процессе обучения сети. Поля **tr.trainInd**, **tr.valInd**, **tr.testInd** содержат индексы векторов, попавших в обучающее, контрольное и тестовое подмножества соответственно.

```
>> tr
epoch: [0 1 2 3 4 5 6 7 8 9 10 11 12]
     perf: [1x13 double]
    vperf: [1x13 double]
    tperf: [1x13 double]
       mu: [1x13 double]
 gradient: [1x13 double]
 trainInd: [1x60 double]
   valInd: [6 15 16 18 19 26 29 36 47 53 58 61 62 70 82 85 87 89 93 96]
  testInd: [5 8 9 12 14 23 24 31 34 35 38 39 44 49 54 56 63 77 79 86]
```

Отобразим графически результаты моделирования :

```
x=-10:.1:10;
y=sim(net,x)
plot(x,x.*cos(x),'-',X,T(i),'.',x,y, '-','LineWidth',2);%,'.')
legend('y=x*cos(x)','with noise','output');%,-1)
```

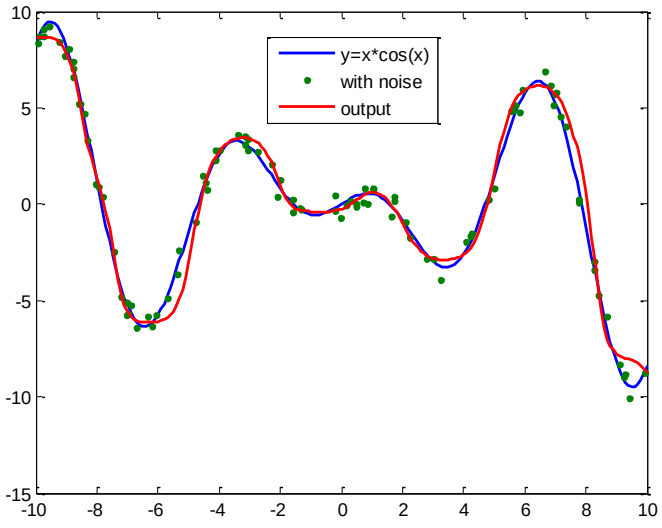


Рис.8. Результат работы обученной сети

Полезным инструментом оценки результата обучения нейронной сети может быть регрессионный анализ сети, т.е. построение функций регрессий результатов **y** (которые дает обученная нейронная сеть) от целевых значений **T** (которые были изначальным условием задачи).

```
y=sim(net,P);
[m,b,r]=postreg(y,T)
m =
    0.9907
b =
    0.0511
r =
    0.9890
```

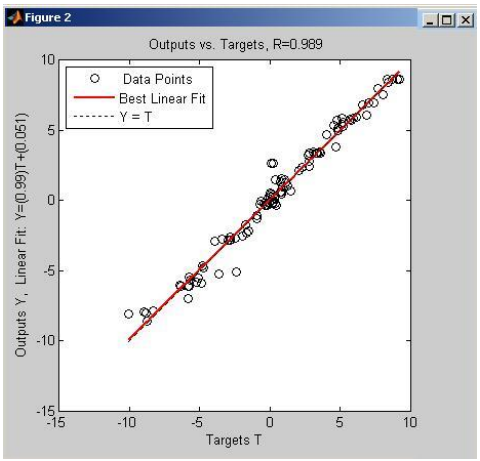


Рис.9. Параметры регрессии сети

Выход сети **y** и вектор целей **T** подаются на вход функции **postreg**, а она возвращает параметры регрессии: первые два – **m** и **b** – определяют наклон и смещение линии регрессии в координатах цель-выход, третий параметр – **r** – коэффициент корреляции. Если выходы точно равны целям, то наклон должен быть равен 1, а смещение 0. В данном примере эти параметры достаточно точно отражают линейную регрессию. Коэффициент корреляции близок к 1, это указывает на существенную корреляцию между выходами и целями, т.е. малые изменения цели будут адекватно отражаться в выходах нейронной сети, что является характеристикой ее высокого качества.

Практические задания

Задание 1. Метод обратного распространения ошибок. Изучите Раздел 6, §6.2, п.6.2.1 Метод обратного распространения ошибок курса лекций Воронцова К.В. «Математические методы обучения по прецедентам» (файл Voron-ML-1-lections.pdf) [1].

Задание 2. Функции к применению. Изучите работу функций **newff**, **newcf**, **newfftd**, **postreg**, **mapminmax**, **mapstd**, **processpca**, **fixunknowns**, **removeconstantrows**.

Изучите работу функций создания N-слойных сетей прямой передачи сигнала: **feedforwardnet**, **fitnet**, **patternnet**, **cascadeforwardnet**.

Задание 3. Изучите работу ff-сетей на демонстрационных моделях пакета MATLAB:

- *Help\Demos\Toolboxes\Neural Network\ Backpropagation*:
  - Generalization (Function Approximation): **`nnd11gn.m`**
- *Help\Demos\Toolboxes\Neural Network\ Application Examples*:
  - Character Recognition: **`appcr1.m`** (**`newff`**)
  - Crab Classification: **`crabclassify.m`** (**`newff`**).

Задание 4. Аппроксимация функции одной переменной. Для заданного преподавателем варианта (таблица):

1) Сгенерировать обучающее множество, воспользовавшись следующим кодом:

```
i=5;          % the Difficulty Index of the Function for Approximation
P = -2:(.4/i):2;         % Training Subset: Inputs
T = 1 + sin(i*pi*P/4);   % Training Subset: Targets
P1 = -2:(.04/i):2;       % Testing Subset
```

2) Создать двухслойную ff-сеть при помощи функции **`newff`**. Предварительно задать произвольное число нейронов **`S1`** скрытого слоя, например:

```
S1=5; % Number of Hidden Neurons S1
```

Обучающее множество должно использоваться только для обучения, т.е. не использоваться для контроля и тестирования.

3) Установить параметры обучения:

```
net.trainParam.goal=0.001;
net.trainParam.show=NaN;
net.trainParam.epochs=1;
```

4) В цикле выполнять пошагово обучение на обучающем множестве **`{P,T}`** и тестирование на тестовом множестве **`P1`** сети, визуализируя каждый шаг процесса (пример на рис.10).

```
[net,tr]=train(net,P,T);    % Training
Y1=sim(net,P1);             % Testing
```

5) Критерием выхода из цикла должно служить достижение заданной цели или количество пройденных эпох. Для этого исследуйте структуру **`tr`** выходного параметра функции **`train`**.

6) Выполнить регрессионный анализ выходных и целевых данных (функция **`postreg`**);

7) Если обучение выполнено неудачно, изменить количество нейронов **`S1`** скрытого слоя и повторить весь алгоритм заново, пока обучение не достигнет заданной цели.

8) **Рассмотрите способ решения данной задачи при помощи функций feedforwardnet, fitnet.**

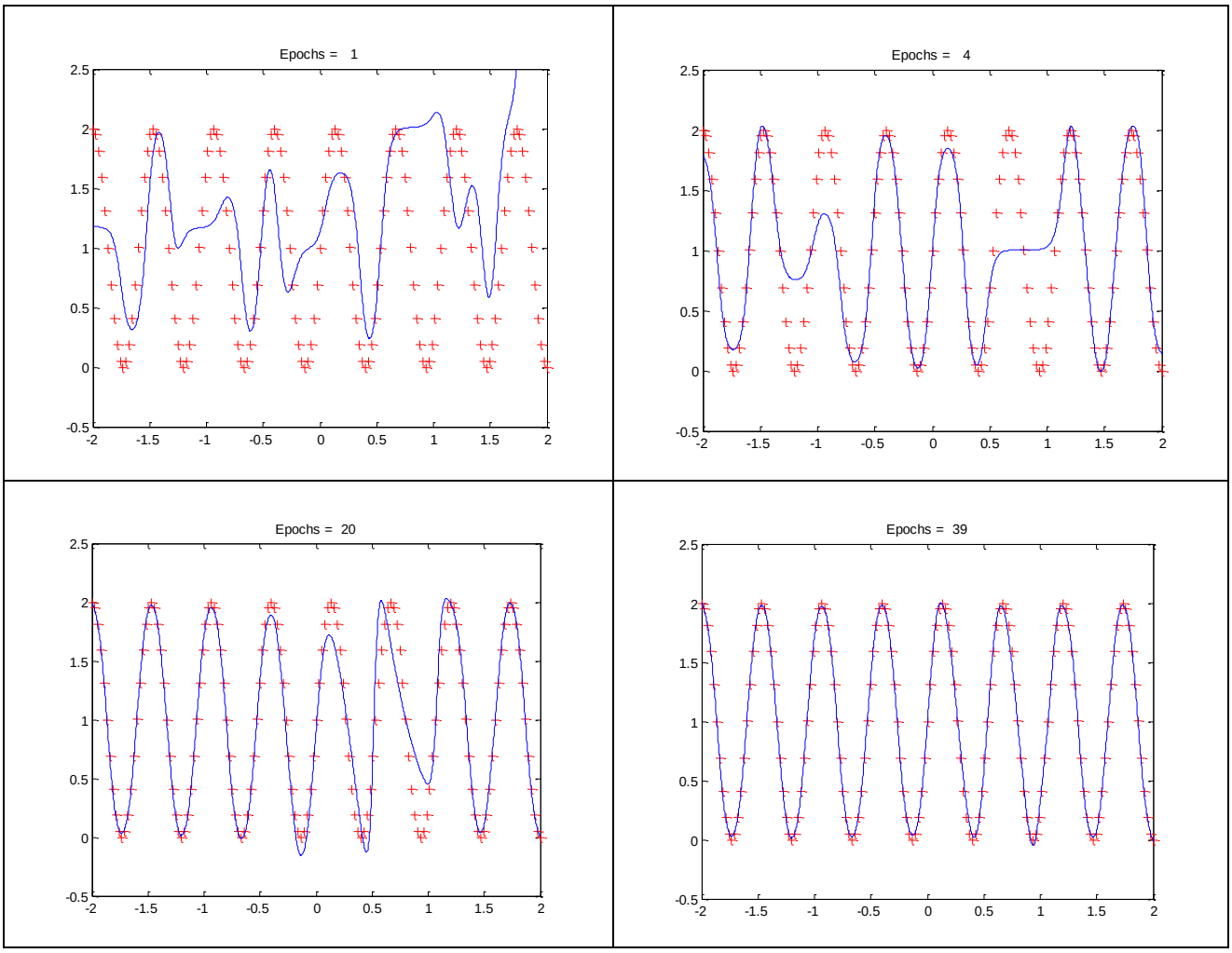


Рис.10. Пошаговый процесс аппроксимации функции

Варианты заданий:

№ варианта	Значение индекса i сложности аппроксимируемой функции	
	Задание 1	Задание 2
1	1.0	4
2	1.2	5
3	1.4	6
4	1.6	7
5	1.8	8
6	1.9	9
7	1.1	10
8	1.3	11
9	1.5	12
10	1.7	15

Задание 5. Оснастите приложение, реализующее Задание 3, графическим интерфейсом пользователя, например, как на рис.11. Значение индекса `i` сложности аппроксимируемой

функции и количество нейронов **`S1`** скрытого слоя должны задаваться при помощи графических компонентов **`Slider`**.

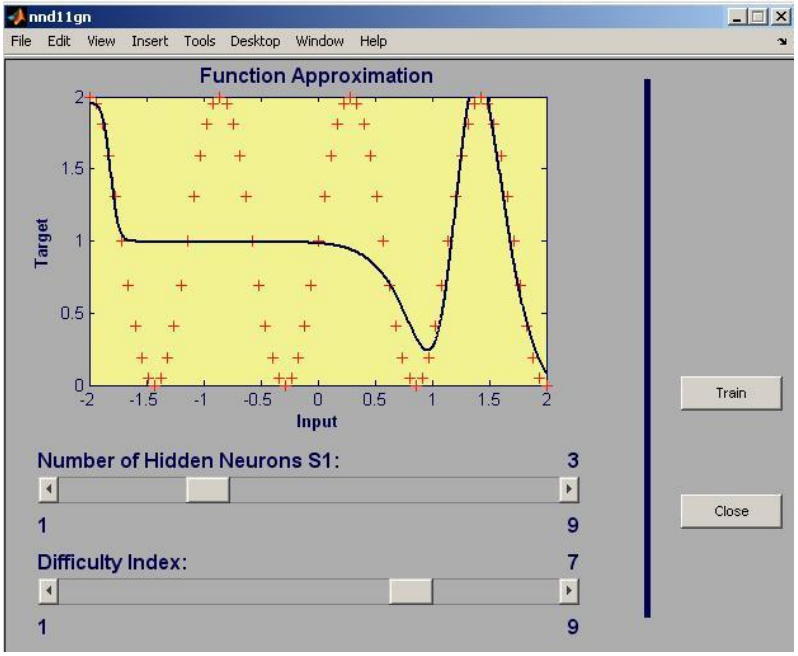


Рис.11. Графический интерфейс пользователя

Литература

1. Воронцов К.В. Курс лекций «Математические методы обучения по прецедентам». (Voron-ML-1-lections.pdf, Раздел 6, §6.2, п.6.2.1 *Метод обратного распространения ошибок*).
2. H.Demuth, M.Beale. Neural Network Toolbox. For use with MATLAB. 840 p. (MathWorks_nnet.pdf, part 5: *Backpropagation*)
3. M.Beale, M. Hagan, H.Demuth. Neural Network Toolbox™. User's Guide (MathWorks_nnet_ug.pdf, part 2: *Multilayer Networks and Backpropagation Training*)
4. Медведев В. С. Нейронные сети. MATLAB 6. Учебно-справочное издание / В. С. Медведев, В. Г. Потемкин. – М.: Диалог МИФИ, 2002 (*разд. 3.2.Методы обучения*).