Янкевич Анна Артемовна, группа 2.1
Лабораторная работа № 4
Вариант № 9

Задание

1) Провести тестирование нейронной сети с заданным тем же дискретизации и смещением диапазона значений корней полинома 3-й степени
2) Провести переобучение нейронной сети с увеличенным числа нейронов в скрытом слое и проанализировать полученный результат.
3) Написать программу для решения уравнений.
4) Построить график зависимости среднеквадратичной ошибки от количества нейронов в скрытом слое

Код программы

```
clear all;
close all;
%Для 200
mi=-10; ma=10; %диапазон значений
delt=5;%шаг дискретизации
smesh=5;
intrvl=mi+smesh:delt:ma+smesh;
cnt=1;
for x1=intrvl,
   for x2=mi:delt:ma,
       for x3=mi:delt:ma,
           for x4= mi:delt:ma,
               for x5=mi:delt:ma,
                   A(cnt)=-(x1+x2+x3+x4+x5);
                   B(cnt)=x1*x2 + x1*x3 + x1*x4 + x1*x5 +x2*x3+ x2*x4+x2*x5 +
x3*x4 + x3*x5 + x4*x5;
                   C(cnt)=-x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x3*x4 + x1*x3*x5
+ x1*x4*x5 + x2*x3*x4 + x2*x3*x5 + x2*x4*x5 + x3*x4*x5;
                   D(cnt)=x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x4*x5 +x1*x3*x4*x5
+ x2*x3*x4*x5;
                   F(cnt)= x1*x2*x3*x4*x5;
                   X(:,cnt)=[x1; x2; x3; x4; x5];
                   cnt=cnt+1;
               end;
           end;
       end;
   end;
```

```
end;
P=[A;B;C;D;F];
%Для работы с нейронной сети необходимо нормировать массив корней и
%массив коэффициентов при помощи стандартных функций Matlab
Xn=(X-mi)*0.8/(ma-mi)+0.1;
[Pn,Pmin,Pmax]=premnmx(P);
%Создается и обучается нейронную сеть для решения кубического уравнения.
bpn=feedforwardnet([200 5],'trainrp');
view(bpn);
bpn=init(bpn);
bpn.trainParam.epochs=500;
bpn=train(bpn,Pn,Xn);
%Для тестирования нейронной сети формируется случайным образом заполненный
массив корней.
kk=10;
for cnt=1:kk,
   x1=mi+rand*(ma-mi);
   x2=x1+rand*(ma-x1);
   x3=x2+rand*(ma-x2);
   x4=x2+rand*(ma-x3);
   x5=x2+rand*(ma-x4);
   A1(cnt)=-(x1+x2+x3+x4+x5);
   B1(cnt)=x1*x2 + x1*x3 + x1*x4 + x1*x5 +x2*x3+ x2*x4+x2*x5 + x3*x4 + x3*x5 +
x4*x5;
   C1(cnt)=-x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x3*x4 + x1*x3*x5 + x1*x4*x5 +
x2*x3*x4 + x2*x3*x5 + x2*x4*x5 + x3*x4*x5;
   D1(cnt)=x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x4*x5 +x1*x3*x4*x5 + x2*x3*x4*x5;
   F1(cnt)= x1*x2*x3*x4*x5;
   X1(:,cnt)=[x1; x2; x3; x4; x5];
end;
P1=[A1;B1;C1;D1;F1];
P1n=tramnmx(P1,Pmin,Pmax);
X1n=(X1-mi)*0.8/(ma-mi)+0.1;
y=sim(bpn,P1n);
mse(X1n-y)
%Для 500
mi=-10; ma=10; %диапазон значений
delt=5;%шаг дискретизации
smesh=5;
intrvl=mi+smesh:delt:ma+smesh;
cnt=1;
for x1=intrvl,
   for x2=mi:delt:ma,
       for x3=mi:delt:ma,
           for x4= mi:delt:ma,
               for x5=mi:delt:ma,
                   A(cnt)=-(x1+x2+x3+x4+x5);
                   B(cnt)=x1*x2 + x1*x3 + x1*x4 + x1*x5 +x2*x3+ x2*x4+x2*x5 +
x3*x4 + x3*x5 + x4*x5;
```

```
                    C(cnt)=-x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x3*x4 + x1*x3*x5
+ x1*x4*x5 + x2*x3*x4 + x2*x3*x5 + x2*x4*x5 + x3*x4*x5;
                    D(cnt)=x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x4*x5 +x1*x3*x4*x5
+ x2*x3*x4*x5;
                    F(cnt)= x1*x2*x3*x4*x5;
                    X(:,cnt)=[x1; x2; x3; x4; x5];
                    cnt=cnt+1;
                end;
            end;
        end;
    end;
end;
P=[A;B;C;D;F];
%Для работы с нейронной сети необходимо нормировать массив корней и
%массив коэффициентов при помощи стандартных функций Matlab
Xn=(X-mi)*0.8/(ma-mi)+0.1;
[Pn,Pmin,Pmax]=premnmx(P);
%Создается и обучается нейронную сеть для решения кубического уравнения.
bpn=feedforwardnet([500 5],'trainrp');
view(bpn);
bpn=init(bpn);
bpn.trainParam.epochs=500;
bpn=train(bpn,Pn,Xn);
%Для тестирования нейронной сети формируется случайным образом заполненный
массив корней.
kk=10;
for cnt=1:kk,
   x1=mi+rand*(ma-mi);
   x2=x1+rand*(ma-x1);
   x3=x2+rand*(ma-x2);
   x4=x2+rand*(ma-x3);
   x5=x2+rand*(ma-x4);
   A1(cnt)=-(x1+x2+x3+x4+x5);
   B1(cnt)=x1*x2 + x1*x3 + x1*x4 + x1*x5 +x2*x3+ x2*x4+x2*x5 + x3*x4 + x3*x5 +
x4*x5;
   C1(cnt)=-x1*x2*x3 + x1*x2*x4 + x1*x2*x5 + x1*x3*x4 + x1*x3*x5 + x1*x4*x5 +
x2*x3*x4 + x2*x3*x5 + x2*x4*x5 + x3*x4*x5;
   D1(cnt)=x1*x2*x3*x4 + x1*x2*x3*x5 + x1*x2*x4*x5 +x1*x3*x4*x5 + x2*x3*x4*x5;
   F1(cnt)= x1*x2*x3*x4*x5;
   X1(:,cnt)=[x1; x2; x3; x4; x5];
end;
P1=[A1;B1;C1;D1;F1];
P1n=tramnmx(P1,Pmin,Pmax);
X1n=(X1-mi)*0.8/(ma-mi)+0.1;
y=sim(bpn,P1n);
mse(X1n-y)
%3 Написать программу для решения уравнений
%3.1 Решение уравнения
p=[-1;3;2;1;0;1]; %Коэффициенты исходного уравнения
```

```
solution = roots(p); % Находим корни исходного уравнения численным методом
Peq=[3;-2;1;0;1]; %Коэффициенты уравнения, приведенного к каноническому виду
Pnorm=tramnmx(Peq,Pmin,Pmax); %Проведем нормировку вектора коэффициентов
res=sim(bpn,Pnorm); %Получение решения с помощью нейронной сети
disp('Решение уравнения');
disp(solution);
disp('СКО решенного уравнения с помощью нейронной сети и численного метода
roots()');
disp(mse(res-solution));
%4 Построить график зависимости среднеквадратичной ошибки от количества
%нейронов в скрытом слое
%4.1 Находим зависимость СКО от количества нейронов и строим график
sko=zeros(2,20);
for n=1:20
   hidden_neurons=50*n;
   bpn=feedforwardnet([hidden_neurons 3],'trainrp');
   bpn=init(bpn);
   bpn.trainParam.epochs=300;
   bpn=train(bpn,Pn,Xn);
   y=sim(bpn,P1n);
   sko(1,n)=hidden_neurons;
   sko(2,n)=mse(X1n-y);
end
figure('Name','СКО от количества нейронов в скрытом слое');
plot(sko(1,:),sko(2,:));
```

Результаты выполнения

```
СКО решенного уравнения с помощью нейронной сети и численного метода roots()
   2.0399 - 0.0032i
```

СКО от количества нейронов в скрытом слое