作业三

1. 试编程实现累积BP算法，在西瓜数据集2.0上(用训练数据)训练一个单隐层网络，用验证集计算出均方误差。要自己实现，不能直接调用现成的库函数。

clear

%{

数据定义

转化 色泽 根蒂 敲声 纹理 脐部 触感 好瓜

1 青绿 蜷缩 浊响 清晰 凹陷 硬滑 是-1

2 乌黑 稍蜷 沉闷 稍糊 稍凹 软粘 否-0

3 浅白 硬挺 清脆 模糊 平坦

西瓜数据集2.0

训练集

编号 色泽 根蒂 敲声 纹理 脐部 触感 好瓜

1 1 1 1 1 1 1 1

2 2 1 2 1 1 1 1

3 2 1 1 1 1 1 1

6 1 2 1 1 2 2 1

7 2 2 1 2 2 2 1

10 1 3 3 1 3 2 0

14 3 2 2 2 1 1 0

15 2 2 1 1 2 2 0

16 3 1 1 3 3 1 0

17 1 1 2 2 2 1 0

验证集

4 1 1 2 1 1 1 1

5 3 1 1 1 1 1 1

8 2 2 1 1 2 1 1

9 2 2 2 2 2 1 0

11 3 3 3 3 3 1 0

12 3 1 1 3 3 2 0

13 1 2 1 2 1 1 0

%}

trainingX = [

1,1,1,1,1,1;

2,1,2,1,1,1;

2,1,1,1,1,1;

1,2,1,1,2,2;

2,2,1,2,2,2;

1,3,3,1,3,2;

3,2,2,2,1,1;

2,2,1,1,2,2;

3,1,1,3,3,1;

1,1,2,2,2,1];

trainingY=[1;1;1;1;1;0;0;0;0;0];

testX = [

1,1,2,1,1,1;

3,1,1,1,1,1;

2,2,1,1,2,1;

2,2,2,2,2,1;

3,3,3,3,3,1;

3,1,1,3,3,2;

1,2,1,2,1,1];

testY=[1;1;1;0;0;0;0];

%获取输入参数的样本数与参数数

[row,column]=size(trainingX); %row矩阵的行数，表示总共有多少个训练集。column是矩阵的列数，表示训练集的输入。

OutputLayerNum=1; %输出层神经元

v=rand(column,column+1); %输入层与隐层的权值,v是一个d行d+1列矩阵

w=rand(column+1,OutputLayerNum); %隐层与输出层的权值,w是一个d+1行1列矩阵

gamma=rand(column+1); %隐层阈值,gamma是d+1行1列矩阵

theta=rand(OutputLayerNum); %输出层阈值,theta是1行1列矩阵

output=zeros(row,OutputLayerNum); %输出层输出

b=zeros(column+1); %隐层输出

g=zeros(OutputLayerNum); %均方误差对w,gamma求导的参数

e=zeros(column+1); %均方误差对v,theta求导的参数

LearningRate=0.5; %学习率

IterativeTimes=0; %迭代的次数

AccumulateTimes=0; %同样的均方误差值累积次数

previous\_E=0; %前一次迭代的累计误差

while(1)

IterativeTimes=IterativeTimes+1;

E=0; %当前迭代的均方误差

%计算全部样本输出层输出

for i=1:row %计算隐层的输出

for j=1:column+1

alpha=0;

for k=1:column

alpha=alpha+v(k,j)\*trainingX(i,k);

end

b(i,j)=1/(1+exp(-alpha+gamma(j)));

end %计算输出层输出

for j=1:OutputLayerNum

beta=0;

for k=1:column+1

beta=beta+w(k,j)\*b(i,k);

end

output(i,j)=1/(1+exp(-beta+theta(j)));

end

end

%用来存储累积误差对四个变量的下降方向

delta\_v=zeros(column,column+1);

delta\_w=zeros(column+1,OutputLayerNum);

delta\_gamma=zeros(column+1);

delta\_theta=zeros(OutputLayerNum);

%计算累积误差

for i=1:row

for j=1:OutputLayerNum

E=E+((trainingY(i)-output(i,j))^2)/2;

end %计算w、theta导数参数

for j=1:OutputLayerNum

g(j)=output(i,j)\*(1-output(i,j))\*(trainingY(i)-output(i,j));

end

%计算v、gamma导数参数

for j=1:column+1

teh=0;

for k=1:OutputLayerNum

teh=teh+w(j,k)\*g(k);

end

e(j)=teh\*b(i,j)\*(1-b(i,j));

end

%计算w、theta导数

for j=1:OutputLayerNum

delta\_theta=delta\_theta+(-1)\*LearningRate\*g(j);

for k=1:column+1

delta\_w(k,j)=delta\_w(k,j)+LearningRate\*g(j)\*b(i,k);

end

end

%计算v、gamma导数

for j=1:column+1

gamma(j)= gamma(j)+(-1)\*LearningRate\*e(j);

for k=1:column

delta\_v(k,j)=delta\_v(k,j)+LearningRate\*e(j)\*trainingX(i,k);

end

end

end

%更新参数

v=v+delta\_v;

w=w+delta\_w;

gamma=gamma+delta\_gamma;

theta=theta+delta\_theta;

%迭代终止条件

if(abs(previous\_E-E)<0.0001)

AccumulateTimes=AccumulateTimes+1;

if(AccumulateTimes==50)

break;

end

else

previous\_E=E;

AccumulateTimes=0;

end

end

testoutput=zeros(7,OutputLayerNum);%测试集输出层输出

testb=zeros(column+1);%测试集隐层输出

for i=1:7 %计算测试集隐层输出

for j=1:column+1

alpha=0;

for k=1:column

alpha=alpha+v(k,j)\*testX(i,k);

end

testb(i,j)=1/(1+exp(-alpha+gamma(j)));

end %计算输出层输出

for j=1:OutputLayerNum

beta=0;

for k=1:column+1

beta=beta+w(k,j)\*testb(i,k);

end

testoutput(i,j)=1/(1+exp(-beta+theta(j)));

end

end

testE=0;

for i=1:7

for j=1:OutputLayerNum

testE=testE+((testY(i)-testoutput(i,j))^2)/2;

end

end

testE=testE/7;