作业三

关文聪 2016060601008

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。

解答：首先对每个属性通过赋不同值进行划分，先用训练集数据训练一个单隐层神经网络，通过设定合适的学习率、迭代次数等，训练得到网络的阈值、连接权重等参数，得到的网络再输入测试集数据计算均方误差。

Matlab实现代码：

clear

clear all

%对每个属性进行赋值：

%色泽：青绿-1 乌黑-2 浅白-3

%根蒂：蜷缩-1 稍蜷-2 硬挺-3

%敲声：浊响-1 沉闷-2 清脆-3

%纹理：清晰-1 稍糊-2 模糊-3

%脐部：凹陷-1 稍凹-2 平坦-3

%触感：硬滑-1 软粘-2

%好瓜：是-1 否-0

X=[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;

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];

Y=[1;1;1;1;1;0;0;0;0;0;1;1;1;0;0;0;0];

trainingX=X(1:10,:); %取前10个数据为训练集

trainingY=Y(1:10,:); %取前10个数据为训练集

testX=X(11:17,:); %取后7个数据为测试集

testY=Y(11:17,:); %取后7个数据为测试集

[row,column]=size(trainingX); %row是矩阵的行数，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; %输出层神经元数

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

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

gamma=rand(column+1); %隐层阈值,gamma是column+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=0.1; %学习率，在0-1之间

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))); %代入sigmoid函数

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))); %代入sigmoid函数

end

end

%用来存储累积误差对四个变量的下降方向，即delta项

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; %均方误差E

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));%µ¼Êý

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)); %导数

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;

%设置迭代终止条件：前后两次误差之差绝对值小于0.01%，且累计500次

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

AccumulateTimes=AccumulateTimes+1;

if(AccumulateTimes==500) %误差位于设定范围内累计500次

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

%计算测试集隐层的输出

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))); %代入sigmoid函数

end

%计算测试集输出层输出

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))); %代入sigmoid函数

end

end

%计算测试集均方误差

testE=0;

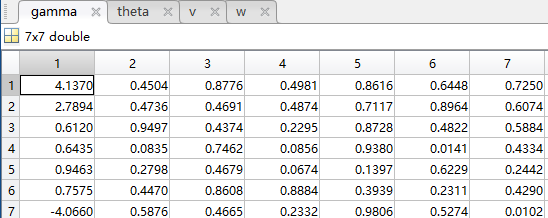
for i=1:7

for j=1:OutputLayerNum

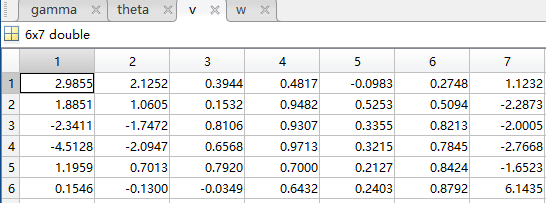
testE=testE+((testY(i)-testoutput(i,j))^2)/2;

end

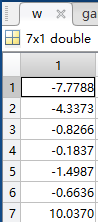
end

运行代码，训练得到隐层阈值gamma为

输出层阈值theta=1.1740 

输入层与隐层的权重v为

隐层与输出层的权重w为



应用于测试集上，计算得到测试集上的均方误差E为1.4472（大约在1.45左右） 