Assignment-2
BITS F312 – Neural Networks and Fuzzy Logic

Name:PITTALA TRINATH SAI SUBHASH REDDY
ID:2017A7PS0228H

1)

CODE:-

```
clc;
clear;
S = load('data5.mat');
X = S.x;
for i = 1:72
X(:,i) = (X(:,i)-mean(X(:,i)))./std(X(:,i));
end
size_data = size(X);%num of training examples = 2148
n0 = size_data(2) - 1; %num of ip features = 72
n1 = 36; % num of neurons in hidden layer 1
n2 = 12; % num of neurons in hidden layer 2
n3 = 2; % num of neurons in the output layer
w1 = rand(n1, n0).*0.1;
b1 = zeros(n1, 1);
w2 = rand(n2, n1).*0.1;
b2 = zeros(n2, 1);
w3 = rand(n3, n2).*0.1;
b3 = zeros(n3, 1);
dw1 = zeros(n1, n0);
db1 = zeros(n1, 1);
dw2 = zeros(n2, n1);
db2 =zeros(n2, 1);
dw3 = zeros(n3, n2);
db3 = zeros(n3, 1);
num_iter = 200;
J = zeros(num_iter, 1);
Y = X(:,73);
alpha = 0.001;
%1504 training examples(70%) and 644 testing examples
m = 1504;
xtrain = zeros(1504, 72);ytrain = zeros(2, 1504);
xtrain = X([1:752 ,1074:1825],1:72);
y1 = Y([1:752 ,1074:1825]);
y1 = y1';
ytrain(1, :) = y1;
for var = 1 : 1504
if ytrain(1, var) == 1
ytrain(2, var) = 0;
else
ytrain(2, var) = 1;
```

```
end
end
xtest = X([753 : 1073 , 1826 : 2148], 1 : 72);
ytest = Y([753 : 1073 , 1826 : 2148]);

% TRAINING
for t = 1 : num_iter
dw1=zeros(n1,n0);db1=zeros(n1,1);dw2=zeros(n2,n1);db2=zeros(n2,1);dw3=zeros(n3,
n2);db3=zeros(n3,1);
j = 0;
for i = 1 : 1504
z1=zeros(n1,1);a1=zeros(n1,1);z2=zeros(n2,1);a2=zeros(n2,1);z3=zeros(n3,1);a3=z
eros(n3,1);
for q = 1 : n1
for r = 1 : n0
z1(q)=z1(q)+w1(q,r)*xtrain(i,r);
end
z1(q)=z1(q)+b1(q);
end
a1=sigmoidFunction(z1);
for p=1:n2
for q=1:n1
z2(p)=z2(p)+w2(p,q)*a1(q);
end
z2(p)=z2(p)+b2(p);
end
a2=sigmoidFunction(z2);
for k=1:n3
for p=1:n2
z3(k)=z3(k)+w3(k,p)*a2(p);
end
z3(k) = z3(k) + b3(k);
end
a3 = sigmoidFunction(z3);
for k = 1 : n3
j = j + ((ytrain(k, i) - a3(k))^2);
end
for k = 1 : n3
db3(k) = db3(k) - ((ytrain(k, i) - a3(k))*z3(k)*(1 -z3(k)));
end
for k = 1 : n3
for p = 1 : n2
dw3(k,p)=dw3(k,p)-((ytrain(k,i)-a3(k))*z3(k)*(1-z3(k))*a2(p));
end
end
for p = 1 : n2
for k = 1 : n3
db2(p)=db2(p)-((ytrain(k,i)-a3(k))*z3(k)*(1-z3(k))*w3(k,p)*z2(p)*(1-z2(p)));
```

```
end
end
for p = 1 : n2
for q = 1 : n1
for k = 1 : n3
dw2(p,q)=dw2(p,q)-((ytrain(k,i)-a3(k))*z3(k)*(1-z3(k))*w3(k,p)*z2(p)*(1-z2(p))*
a1(q));
end
end
end
for q = 1 : n1
for k = 1 : n3
for p = 1 : n2
db1(q)=db1(q)-((ytrain(k,i)-a3(k))*z3(k)*(1-z3(k))*w3(k,p)*z2(p)*(1-z2(p))*w2(p
, q)*z1(q)*(1-z1(q)));
end
end
end
for q = 1 : n1
for r = 1 : n0
for k = 1 : n3
for p = 1 : n2
dw1(q,r)=dw1(q,r)-((ytrain(k,i)-a3(k))*z3(k)*(1-z3(k))*w3(k,p)*z2(p)*(1-z2(p))*
w2(p,q)*z1(q)*(1-z1(q))*xtrain(i,r));
end
end
end
end
end
J(t) = j/(2);
b3 = b3 - alpha*db3;w3 = w3 - alpha*dw3;
b2 = b2 - alpha*db2;w2 = w2 - alpha*dw2;
b1 = b1 - alpha*db1;w1 = w1 - alpha*dw1;
end

  % TESTING
  accuracy = 0;
  for i = 1 : 644
     for q = 1 : n1
         for r = 1 : n0
             z1(q) = z1(q) + w1(q,r)*xtest(i, r);
         end
     z1(q) = z1(q) + b1(q);
     end
     a1 = sigmoidFunction(z1);
     for p = 1 : n2
```

```
        for q = 1 : n1
            z2(p) = z2(p) + w2(p,q)*a1(q);
        end
        z2(p) = z2(p) + b2(p);
    end
    a2 = sigmoidFunction(z2);
    for k = 1 : n3
        for p = 1 : n2
            z3(k) = z3(k) + w3(k,p)*a2(p);
        end
        z3(k) = z3(k) + b3(k);
    end
    a3 = sigmoidFunction(z3);
    [prob, pred] = max(a3);
    if pred == ytest(i)
        accuracy = accuracy + 1;
    end
end
accuracy = (100*accuracy)/644;

function g = sigmoidFunction(z)
%   Compute sigmoidFunction function

 % You need to return the following variables correctly
 g = zeros(size(z));

 % Instructions: z can be a matrix, vector or scalar
 g = 1.0 ./ ( 1.0 + exp(-z)); % For Matlab
 % g = 1.0 ./ ( 1.0 + e.^(-z)); % For Octave, it can use 'exp(1)' or 'e'

end
```

Accuracy:- 49.8447%

2)

CODE:-

```
data=importdata('data5.mat');
X=data(:,1:end-1);
Y=data(:,end);
s = RandStream('mt19937ar','Seed',1);
trainInput=[]; %Define arrays to segregate training and testing data
trainOutput=[];
testInput=[];
testOutput=[];
for j=1:size(X,1)
```

```
    if rand<0.7
        trainInput=[trainInput;X(j,:)];
        trainOutput=[trainOutput;Y(j,:)];
    else
        testInput=[testInput;X(j,:)];
        testOutput=[testOutput;Y(j,:)];
    end
end
x=trainInput;
y=trainOutput;
xt=testInput;
yt=testOutput;
[l,mu]=kmeans(x,10);
H=zeros(1495,10);
for i=1:size(x,1)
    for j=1:size(mu,1)
        H(i,j)=(norm(x(i,:)-mu(j,:)))^3; %Cubic kernel
    end
end
kk=pinv(H); %Pseudoinverse of matrix
w=kk*y;
for i1=1:size(xt,1)
    for j=1:size(mu,1)
        Ht(i1,j)=(norm(xt(i1,:)-mu(j,:)))^3;
    end
end
yp=Ht*w;
yp(yp>0.5) = 1;
yp(yp<0.5) = 0;
[cm, order]=confusionmat(yt,yp);
IA_1 = cm(1,1)/(cm(1,1) + cm(1,2));
IA_2 = cm(2,2)/(cm(2,1) + cm(2,2));
OA = (cm(1,1) + cm(2,2))/(sum(sum(cm)));
```

Confusion Matrix:

206 103

112 188

Accuracy:

0.6470

3)

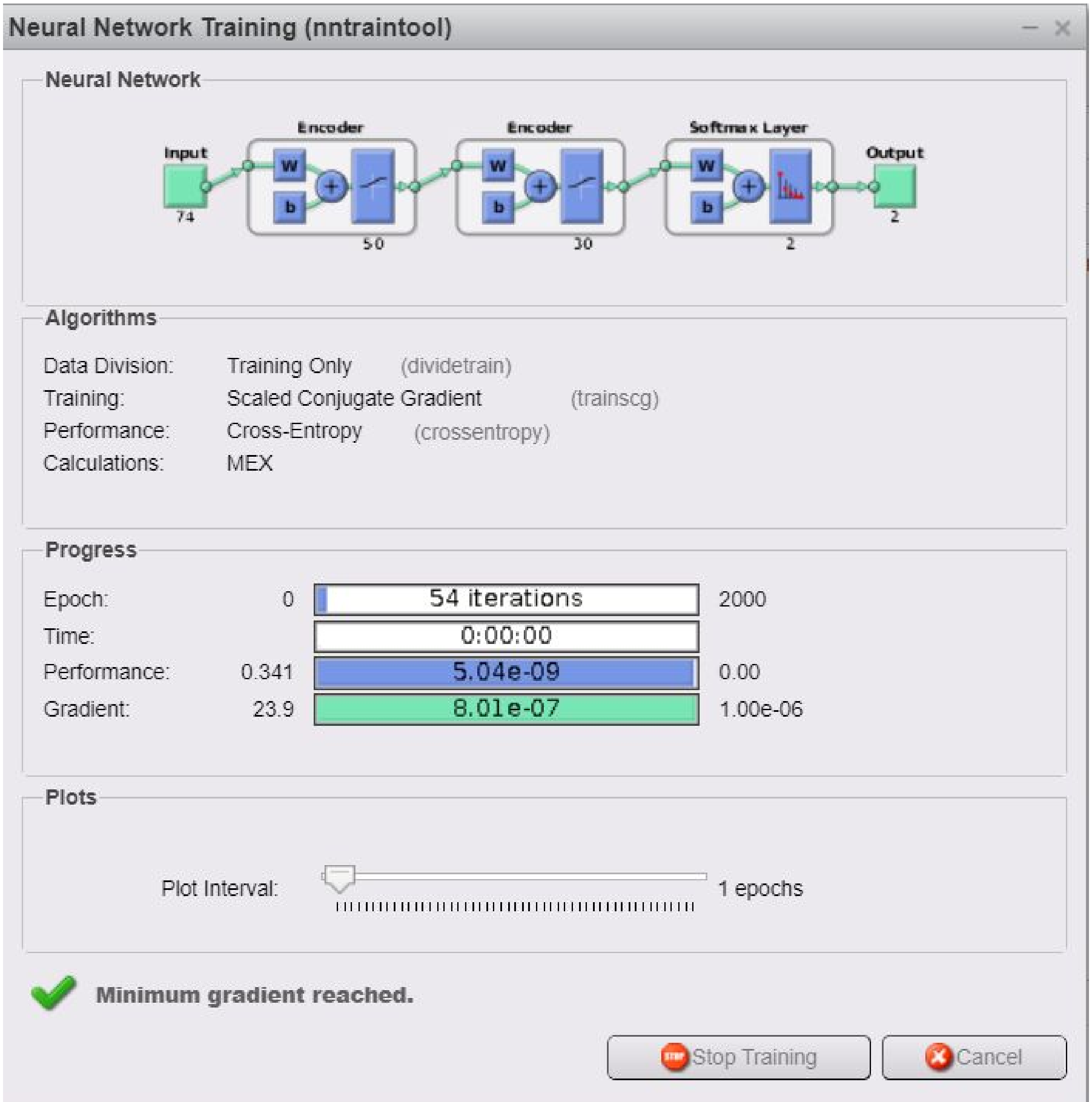
CODE:-

```
clc;
clear;
data=importdata('data5.mat');
```

```matlab
% normalization of the data
data(:,1:72)= (data(:,1:72)-mean(data(:,1:72)));
data(:,1:72)= (data(:,1:72)./std(data(:,1:72)));
% divide data into 70:30
data1=data(randperm(size(data,1)),:);
datatr=(data1(1:1504,:));
datatst=(data1(1505:2148,:));
% ytst:test output
ytst=(datatst(:,73));
% ytr:trainning output
ytr=(datatr(:,73));
% converting to rows
datatr=(data1(1:1504,:))';
datatst=(data1(1505:2148,:))';
%for training data class allocation
for i=1:size(ytr)
if(ytr(i)==1)
ymat(i,2)=1;
ymay(i,1)=0;
     else
         ymat(i,1)=1;
         ymat(i,2)=0;
     end
  end
  ymat=ymat';
  %adding baise values to the feature matrix
  datatr=[ones(1,size(datatr,2));datatr];
  datatst=[ones(1,size(datatst,2));datatst];
  %first encoder input is training data
  %Maximum number of training epochs or iterations, specified as the
  %comma-separated pair consisting of 'MaxEpochs' and a positive
  %integer value.
  autoen1=trainAutoencoder(datatr,50,'MaxEpochs',2000);
  xpr1=predict(autoen1,datatr);
  feat1=encode(autoen1,datatr);
  %second encoder its input is previous ones output
  autoen2=trainAutoencoder(feat1,30,'MaxEpochs',2000);
  feat2=encode(autoen2,feat1);
  %third encoder its input is previous ones output
  autoen3=trainSoftmaxLayer(feat2,ymat,'MaxEpochs',2000);
  stackednet=stack(autoen1,autoen2,autoen3);
  %fine tuning
  stackednet=train(stackednet,datatr,ymat)
  %prediction output is in fractions
  yprmat=stackednet(datatst);
```

```matlab
%class allocation, ypr:final output(predicted)
for i=1:size(yprmat,2)
   if(yprmat(1,i)>yprmat(2,i))
       ypr(i)=0;
   else
       ypr(i)=1;
   end
end
% accuracy calculation
ytst=ytst';
t=0;
for i=1:size(ypr,2)
   if(ypr(i)==ytst(i))
       t=t+1;
   end
end
accuracy=t/644
```

Accuracy: 1



4)

CODE:-

```
clc;
clear;
data=importdata('data5.mat');
  % normalization of the data
  data(:,1:72)= (data(:,1:72)-mean(data(:,1:72)));
  data(:,1:72)= (data(:,1:72)./std(data(:,1:72)));
  % divide data into 70:30
  data1=data(randperm(size(data,1)),:);
  datatr=data1(1:1504,:);
```

```
datatst=data1(1505:2148,:);
% ytst:test output
ytst=datatst(:,73);
% ytr:training output
ytr=datatr(:,73);
datatr=[ones(size(datatr,1),1),datatr(:,1:72)];
datatst=[ones(size(datatst,1),1),datatst(:,1:72)];
%training with tanh activation function
n=input('Enter the no. of hidden neurons: ');
win=randn(size(datatr,2),n);
temp=datatr*win;
h=tanh(temp);
wout=pinv(h)*ytr;
%testing with tanh activation function
temp1=datatst*win;
h1=tanh(temp1);
ypr=h1*wout;
%class allocation, ypr:final output(predicted)
for i=1:size(ypr,1)
   if(ypr(i)>0.5)
       ypr(i)=1;
   else
       ypr(i)=0;
   end
end
% accuracy calculation
t1=0;
for i=1:size(ypr,1)
   if(ypr(i)==ytst(i))
       t1=t1+1;
   end
end
% gives confusion matrix
[cm,a]=confusionmat(ytst,ypr);
cm
accuracy = t1/644
```

Enter the no. of hidden neurons:12

Confusion Matrix:

292 21

276 55

Accuracy : 0.5388

5)

CODE:-

```
clc;
clear;
  data=importdata('data5.mat');
  % normalization of the data
  data(:,1:72)= (data(:,1:72)-mean(data(:,1:72)));
  data(:,1:72)= (data(:,1:72)./std(data(:,1:72)));
  % divide data into 70:30
  data1=data(randperm(size(data,1)),:);
  datatr=data1(1:1504,:);
  datatst=data1(1505:2148,:);
  % ytst:test output
  ytst=datatst(:,73);
  % ytr:trainning output
  ytr=datatr(:,73);
  datatr=[ones(size(datatr,1),1),datatr(:,1:72)];
  datatst=[ones(size(datatst,1),1),datatst(:,1:72)];
  %training with tanh activation function
  n=input('Enter the no. of hidden neurons: ');
  win=randn(size(datatr,2),n);
  temp=datatr*win;
  h=tanh(temp);
  wout=pinv(h)*ytr;
  %testing with tanh activation function
  temp1=datatst*win;
  h1=tanh(temp1);
  ypr=h1*wout;
  %class allocation, ypr:final output(predicted)
  for i=1:size(ypr,1)
     if(ypr(i)>0.5)
         ypr(i)=1;
     else
         ypr(i)=0;
     end
  end
```

```
% accuracy calculation
t1=0;
for i=1:size(ypr,1)
   if(ypr(i)==ytst(i))
       t1=t1+1;
   end
end
% gives confusion matrix
[cm,a]=confusionmat(ytst,ypr);
cm
accuracy=t1/644
```

Enter the no. of hidden neurons:12

Confusion Matrix:

317	**17**
245	**65**

Accuracy: 0.5932