```
function project6
% Alexander Rosales, 4/4/2021
% This code creates a neural network composed of one input layer, one
% hidden layer, and one output layer capable of recognizing the digital
% letters r i c e and the handwritten numbers 0-7.

% Inputs: V0,W0,maxiter,rate,train,target
% V0=random matrix with rows equal to # of hidden cells and columns equal
% to # of input layer cells
% W0= random matrix with rows = to # of outputs and columns = to # of
% hidden layer cells
% maxiter=maximum number of iterations gradient descent is iterated on
% matrices V and W
% rate=step size of gradient descent
% train=training data used to train neural network
% target = what the output of s(W*s(V*train)) should be

% Outputs: Success rate of digital leter recognition, success rate of
% handwriten letter recognition, subplot of rice,rice with each letter
% modified once, rice with each letter modified twice, and a plot for each
% handwritten number 0-7

rng(0,'v5normal'); %makes sure V0 and W0 are the same random numbers
V0=randn(25,25); %each time
W0=randn(2,25); %this creates the randomm matrices
data=train; %creates training data (calls train function)
target=[0 0;0 1;1 0;1 1;0 0;0 1;1 0;1 1;0 0;0 1;1 0;1 1]; %desired output

digitalletterrecog(V0,W0,5000,.1,data,target);
% digitalletterrecog(V0,W0,maxiter,rate,train,target);

V0=randn(784,784); %creates random matrices of appropriate size
W0=randn(3,784);
handwrittennumberrecog(V0,W0,5000,.1)
%V0,W0,maxiter,rate. Train data obtained in function


end

function digitalletterrecog(V0,W0,maxiter,rate,train,target)

[V,W,N]=neural(V0,W0,maxiter,rate,train,target); %calls neural function
%to create weight matrices

le=[1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1]; %codes letter e

counter=0;
k=0;
for i=1:100

    letest=modify(le); %creates a different modified letter e ea. iteration
    output=round(s(W*s(V*letest'))); %computes output with V,W, and soft
    %sigmoid function
    if output==target(4,:) %adds to counter if output is same as target
        counter=counter+1;
    end
    k=k+1;
end
```

```
disp('Digital letter recognition sccess rate (percent):');
disp(100*counter/k); %displays recognition success rate
colormap('gray'); %sets plot color to black and white

for i=1:N
    z=reshape(train(i,:),5,5)'; %creates 12 subplots, one for each letter,
    subplot(3,4,i)%modified letter, and twice modified letter
    imagesc(z);
end
sgtitle('Training Data'); %title for subplots
end

function handwrittennumberrecog(V0,W0,maxiter,rate)
%this function codes the handwritten number recognition portion of the
%project. It calls my neural function to get matrices V and W (trained
%using first 900 data from Samples), runs the remaining 100 data using the
%matrices V & W, then evaluates the recognition success rate and plots an
%image of each number.
% Inputs: V0, W0, maxiter, rate
% Outputs: Success rate, 8 plot pictures of each number 0-7

nsamples = 1000; %extracts test data
for n = 0:7
% open the file corresponding to digit n
fid = fopen(fullfile('digit_database',['data',int2str(n)]));
% read samples digit n
Samples(:,n+1) = fread(fid,28*28*nsamples,'uchar');
end


data=zeros(1000,784); %preallocates data
TSamples=Samples';
targetm=[0 0 0;0 0 1;0 1 1;1 1 1;1 0 1;1 1 0;0 1 0;1 0 0]; %target outputs
target=zeros(1000,3);
a=0;

for i=1:1000

    num=mod(i-1,8)+1; %num=1 corresponds to the number 0, num=8 to 7.

    if mod(i,9)==0 %after all 8 rows (each row is a number) have been
    a=a+1; %iterated through, this increases the value of a by 1
    end %in order to extract the next 784 indices

    %clm1 and clm2 represent the indices for TSamples column.
    %after the first number (a length 784 row vector) from each 8 rows
    %(with each row encoding 125 numbers) have been extracted, clm1 and
    %clm2 shift up by 784 to extract the next handwritten number

    clm1=a*784+1;
    clm2=784*(a+1);

    target(i,:)=targetm(num,:); %creates 1000x3 target matrix
    data(i,:)=TSamples(num,clm1:clm2); %creates 1000x784 matrix
    %with each row encoding one number (row 1 encodes 0, row 2 encodes
    %1, and so on)

end
```

```
traindata=data(1:900,:); %defines training data
testdata=data(901,:); %defines testing data

[V,W,~]=neural(V0,W0,maxiter,rate,traindata,target);

for i=901:1000 %uses calculated V and W matrix to evaluate testdata
    output(i,:)=round(s(W*s(V*data(i,:)')));
end


z=output(901:1000,:)-target(901:1000,:); %every [0 0 0] row represents
%a correctly recognized handwritten number
y=zeros(1,3);
count = 0;
for i=1:100
    if(z(i,:)==y) %adds to count if output matches target
        count  = count + 1;
    end
end

disp('Handwritten letter recognition success rate (percent):')
disp(count); %displays success rate

hold on
colormap('gray');

for i=1:8  %plots each of the 8 numbers
    figure;
    img=reshape(data(i,:),28,28)';
    colormap('gray');
    imagesc(img);
end

end

function [V,W,N] = neural(V0,W0,maxiter,rate,train,target)
%this function codes my multilayer percentron neural network. It begins
%with two random matrices V=hxn W=mxh where m=# of outputs,h=#of hidden
%layer cells, n=# of inputs, then computes the output using s(W*s(V*input)
%and subtracts it from the target value. It then uses gradient descent to
%change the matrices V & W in order to minimize output-target. This runs
%maxiter number of times and finally outputs the matrices V,W, and the
%value N where N is the # of rows of training data.
%inputs:V0,W0,maxiter,rate,train,target
%outputs: V, W, N


V=V0;
W=W0;
[N,~]=size(train); %computes # of rows of training data
for iter=1:maxiter

    j=ceil(rand*N); %picks random # from 1 to N (size of training data)
    p=(train(j,:))'; %picks random training data
    q=s(V*p);
    o=s(W*q); %q and o define output function
    tmp=(o-target(j,:)')*o'*(1-o);
    grad_W=tmp*q'; %computes gradient for W matrix
    grad_V=(W'.*q.*(1-q))*tmp*p'; %computes gradient for V matrix
    V=V-rate*grad_V; %changes W and V matrices according to gradient and
    W=W-rate*grad_W; %rate
end
end


function val=s(x)
% inputs:x (defined by what calls s(x)
% outputs: modified x
val=1./(1+exp(.5-x)); %defines soft sigmoid function
end

function [tdata] = train
%this outputs training data for the digital part of the project
%outputs:tdata (mateix of each letter, modified letter, and twice modified
%letter in order).
lr=[1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0];
li=[0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0];
lc=[1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1];
le=[1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1];
tdata=[lr;li;lc;le;modify(lr);modify(li);modify(lc);modify(le);modify(modify(lr));m
odify(modify(li));modify(modify(lc));modify(modify(le));];
end

function [ltr] = modify(ltr)
%this function modifies a random pixel of the row vector representing a
%letter
%inputs:row vector representing letter
%outputs: that same row vector with a randomly flipped bit
a=ceil(rand*length(ltr));
ltr(a)=~ltr(a);

end
```