MAT128

Project 2

Selma Adam

Rick Alarcon

Lucy Chen

Zhoucheng Ye

March 14, 2018

1. (by Rick)

%WEBSITE TO DOWNLOAD: http://academics.davidson.edu/math/chartier/Numerical/matlab.html

% mnist\_all.mat: contains the MNIST database of handwritten digits.

%It has 10 test tables and 10 train tables

%It contains a training set of 60,000 numbers and a test set of 10,000 numbers.

load('mnist\_all.mat');

% to see the what is in our data

who

%ie. each row in train 0 is one handwritten zero

%To visulize the first image in train0, we have:

digit= train0(1,:);

digitImage = reshape(digit,28,28);

image(rot90(flipud(digitImage),-1));

colormap(gray(256)),axis square tight off;

%To visulize the first image in train1, we have:

digit= train1(1,:);

digitImage = reshape(digit,28,28);

image(rot90(flipud(digitImage),-1));

colormap(gray(256)),axis square tight off;

%To visulize the first image in train2, we have:

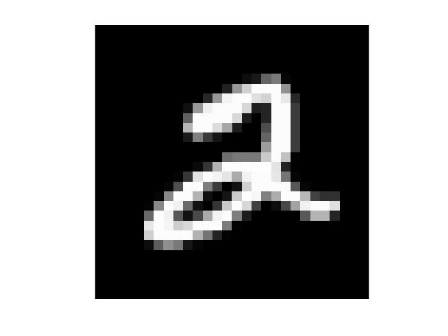
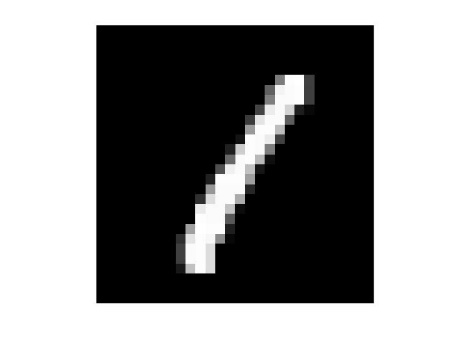
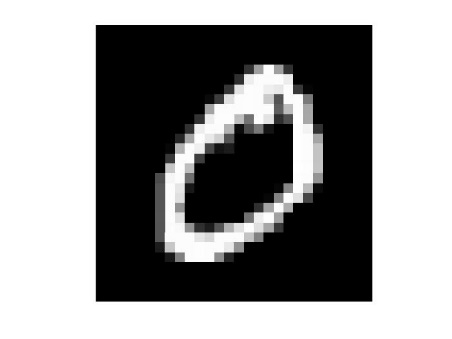
digit= train2(1,:);

digitImage = reshape(digit,28,28);

image(rot90(flipud(digitImage),-1));

colormap(gray(256)),axis square tight off;

Example images:



1. (by Rick)

load('mnist\_all.mat');

%I created two for loops to create the matrix average

% and to iterite each row to find its imagen

for k = 0:9

%this loop created the table of 10 x 784 columns of average numbers

for i = 0:9

filename=sprintf('%s%d','train',i);

myVar = eval(filename);

T(i+1,:) = mean(myVar);

end

%the second loop wil generate the imagens for each average number

% they will be display in one plot using subplot()

subplot(2,5,k+1);

number=T(k+1,:);

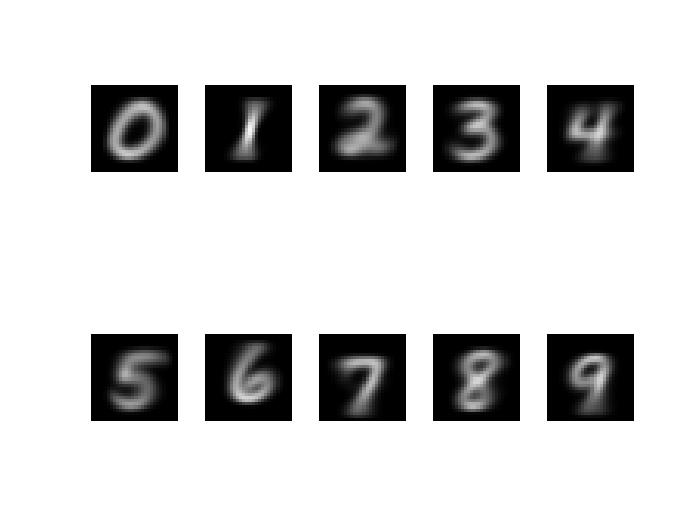
digitImage = reshape(number,28,28);

image(rot90(flipud(digitImage),-1));

colormap(gray(256)),axis square tight off;

hold on;

end

Output image:

1. (by Lucy)

function [net,out] = neuron(X, W)

% X = input data of one number

% W = vector from weight matrix

if(isa(X, 'uint8') || isa(X, 'logical'))

input = double(X);

else

input = X;

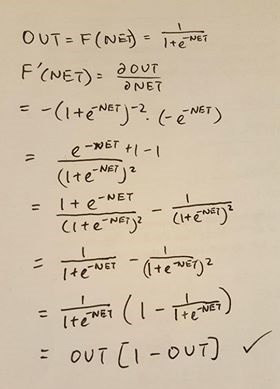
end

net = input\*W';

out = 1./(1.+exp(-net));

end

The derivative derivation is shown below:



For small values of NET, exp(-x) will be approximately 1, so the whole function OUT will be approximately ½, and for large values of NET, exp(-x) approaches 0, so the function will asymptotically approach the value of 1. We can also use OUT2, which gives a similar shape, but when NET values are small, the whole function is approximately ¼, so there is a bigger range between small and large values NET.

1. (by Lucy)

function network = mult\_net(X, W, hid) %data, weights, hid

% X = input data of pixels in row form

% W = weight matrix

% hid = number of hidden layers wanted

% convert the input to be zeros and ones only

if ~exist('hid','var')

hid = size(W{2},1);

end

if(isa(X, 'uint8'))

input = (X > 0);

end

if(isa(X, 'logical'))

input = double(input);

end

numNeuron = size(W{1},1);

net = zeros(1,size(W{2},2));

out = zeros(1,size(W{2},2));

for i = 1:numNeuron

[net(i),out(i)] = neuron(input, W{1}(i,:));

end

for i = 1:hid

for j = 1:numNeuron

net(j) = net(j) \* W{2}(i,j);

end

end

out = 1./(1.+exp(-net));

network = out;

end

The code used to see how to use the functions from part 3 and 4 are shown below:

load('mnist\_all.mat');

% each layer needs 10 neurons, one for each digit

% can input any number of numLayers and numNeurons

numLayers = 3;

numNeurons = 10;

w = cell(1,2);

w{1} = 0.01\*rand(numNeurons,784);

w{2} = rand(numLayers,10);

% checking that neuron works

[tempnet, tempout] = neuron(train9(1,:) > 0, w{1});

%checking that multinetwork works

temp2 = mult\_net(train5(1,:),w);

n = 100;

temp0 = zeros(n,numNeurons);

for i = 1:n

temp0(i,:) = mult\_net(train0(i,:), w);

end

n = length(train9);

temp9 = zeros(n,numNeurons);

for i = 1:n

temp9(i,:) = mult\_net(train9(i,:), w);

end

1. Included in the second code in part 4

w = cell(1,2);

w{1} = 0.01\*rand(numNeurons,784);

w{2} = rand(numLayers,10);

1. (by Zhuocheng)
2. (by Selma)