Intelligent Systems

20-EECE-5136

Homework 4

Due Tuesday, November 21st, 2017 A. D.

By Tomas Seymour

M05873855

**Problem 1**

System Description:  
  
Use the neural network that was created in Homework no. 3 part 1, but instead of the weights that were generated in part 1 use the weights that were learned as in part 2.

It is the same neural network system that takes the first layer of 748 activation input (the greyscale pixels of the handwritten digitalized integer), then outputs to 10 neurons representing which class that handwritten digit should belong to.

Use backpropagation techniques to adjust only the weights between the hidden layer and the output layer of neurons. The input to hidden weights will remain as they are.

Make use of the same .txt files as before.

**Problem 2**

System Description:

Repeat the process for problem 1, but use full backpropagation which trains both matrices of weights instead of just the hidden to output weights.

Train also the weights between the input and hidden neurons.

This is the incomplete code for problem 1.

%this is the code for problem 1 developed in matlatb, it is incomplete and

%does not have the required weights or learning capabilities that are

%satisfactory.

filename = 'MNISTnumImages5000.txt';

a = importdata(filename); % loading pixels. Each row of the matrix represents one number

A = 1./(1 + exp(-a)); % put values in sigmoid function

filename = 'MNISTnumLabels5000.txt';

L = importdata(filename); % loading labels. Each row of the matrix is the number label for each row of matrix A.

% weights for this problem are missing of course because homework 3 part 2

% was never completed. Here they are randomized.

weights1 = randi([-2.0 2.0],200,784); % randomized weights between first and second layers

weights2 = randi([-2.0 2.0],10,200); % randomized weights between second and final layers

weights1 = zeros(200,784);

weights2 = zeros(10,200);

Y1 = [];

Y2 = [];

y1 = [];

y2 = [];

activation1 = [];

activation2 = [];

%testset = A(vector(1:4000),:);

dW2 = 0;

dW1 = 0;

%FORWARD PASS FOR ONE EPOCH

delta3 = [];

delta2 = [];

DELTAij1 = zeros(200,784);

DELTAij2 = zeros(10,200);

%50 epochs

for j = 1:50

Y1 = [];

Y2 = [];

y1 = [];

y2 = [];

activation1 = [];

activation2 = [];

%testset = A(vector(1:4000),:);

dW2 = 0;

dW1 = 0;

delta3 = [];

delta2 = [];

DELTAij1 = zeros(200,784);

DELTAij2 = zeros(10,200);

for i = 1:4000

%forward pass

activation1 = A(i,:).'; %transposed for easy multiplication with weights

y1(i,:) = (weights1\*activation1).';

Y1(i,:) = 1./(1 + exp(-(weights1\*activation1).')); % holds activations for the next layer of 200 neurons (784 -> 200), for 100 different inputs of 784

activation2 = Y1(i,:).';

y2(i,:) = (weights2\*activation2).';

Y2(i,:) = 1./(1 + exp(-(weights2\*activation2).')); % should hold the final layer of neurons which bright up which neuron most represents the given input data of 784 pixels

% the ideal output of the network would be something like all low

% activations except for the one neuron which classifies the

% handwritten number

%backward pass

%next, determine costs, error, and delta

desiredoutput = zeros(10,1); %begin with vector of zero

desiredoutput(L(i)+1) = 1; %must add 1 for MATLAB vector syntax.

delta3 = (Y2(i,:).' - desiredoutput).';

delta2 = (weights2.')\*(delta3.') .\* (Y1(i,:).' .\* (1-(Y1(i,:).')));

delta2 = delta2.';

DELTAij1 = DELTAij1 + delta2.' \* A(i,:);

DELTAij2 = DELTAij2 + delta3.' \* Y1(i,:);

end

%update weights (the first matrix of weights is not updated as part of this

%homework assignment)

%how exactly the weights are to be updated is unclear

Dij1 = DELTAij1./4000;

Dij2 = DELTAij2./4000;

%weights1 = weights1 - Dij1;

weights2 = weights2 - Dij2;

end

%training is complete, test on one new input vector

for i = 4001:4001

activation1 = A(i,:).'; %transposed for easy multiplication with weights

y1(i,:) = (weights1\*activation1).';

Y1(i,:) = 1./(1 + exp(-(weights1\*activation1).')); % holds activations for the next layer of 200 neurons (784 -> 200), for 100 different inputs of 784

activation2 = Y1(i,:).';

y2(i,:) = (weights2\*activation2).';

Y2(i,:) = 1./(1 + exp(-(weights2\*activation2).'));

end

This is the incomplete code for problem 2  
%this is the code for problem 2 developed in matlatb, it is incomplete and

%does not have the required weights or learning capabilities that are

%satisfactory. Will output all 1's after training for any input vector. The

%only difference in this code compared to the first is that one line of code is uncommented.

filename = 'MNISTnumImages5000.txt';

a = importdata(filename); % loading pixels. Each row of the matrix represents one number

A = 1./(1 + exp(-a)); % put values in sigmoid function

filename = 'MNISTnumLabels5000.txt';

L = importdata(filename); % loading labels. Each row of the matrix is the number label for each row of matrix A.

% weights for this problem are missing of course because homework 3 part 2

% was never completed. Here they are randomized.

weights1 = randi([-2.0 2.0],200,784); % randomized weights between first and second layers

weights2 = randi([-2.0 2.0],10,200); % randomized weights between second and final layers

weights1 = zeros(200,784);

weights2 = zeros(10,200);

Y1 = [];

Y2 = [];

y1 = [];

y2 = [];

activation1 = [];

activation2 = [];

%testset = A(vector(1:4000),:);

dW2 = 0;

dW1 = 0;

%FORWARD PASS FOR ONE EPOCH

delta3 = [];

delta2 = [];

DELTAij1 = zeros(200,784);

DELTAij2 = zeros(10,200);

%50 epochs

for j = 1:50

Y1 = [];

Y2 = [];

y1 = [];

y2 = [];

activation1 = [];

activation2 = [];

%testset = A(vector(1:4000),:);

dW2 = 0;

dW1 = 0;

delta3 = [];

delta2 = [];

DELTAij1 = zeros(200,784);

DELTAij2 = zeros(10,200);

for i = 1:4000

%forward pass

activation1 = A(i,:).'; %transposed for easy multiplication with weights

y1(i,:) = (weights1\*activation1).';

Y1(i,:) = 1./(1 + exp(-(weights1\*activation1).')); % holds activations for the next layer of 200 neurons (784 -> 200), for 100 different inputs of 784

activation2 = Y1(i,:).';

y2(i,:) = (weights2\*activation2).';

Y2(i,:) = 1./(1 + exp(-(weights2\*activation2).')); % should hold the final layer of neurons which bright up which neuron most represents the given input data of 784 pixels

% the ideal output of the network would be something like all low

% activations except for the one neuron which classifies the

% handwritten number

%backward pass

%next, determine costs, error, and delta

desiredoutput = zeros(10,1); %begin with vector of zero

desiredoutput(L(i)+1) = 1; %must add 1 for MATLAB vector syntax.

delta3 = (Y2(i,:).' - desiredoutput).';

delta2 = (weights2.')\*(delta3.') .\* (Y1(i,:).' .\* (1-(Y1(i,:).')));

delta2 = delta2.';

DELTAij1 = DELTAij1 + delta2.' \* A(i,:);

DELTAij2 = DELTAij2 + delta3.' \* Y1(i,:);

end

%update weights (full backpropagation, we update both weight matrices)

%how exactly the weights are to be updated is unclear

Dij1 = DELTAij1./4000;

Dij2 = DELTAij2./4000;

weights1 = weights1 - Dij1;

weights2 = weights2 - Dij2;

end

%training is complete, test on one new input vector

for i = 4001:4001

activation1 = A(i,:).'; %transposed for easy multiplication with weights

y1(i,:) = (weights1\*activation1).';

Y1(i,:) = 1./(1 + exp(-(weights1\*activation1).')); % holds activations for the next layer of 200 neurons (784 -> 200), for 100 different inputs of 784

activation2 = Y1(i,:).';

y2(i,:) = (weights2\*activation2).';

Y2(i,:) = 1./(1 + exp(-(weights2\*activation2).'));

end