Code:

**sigmoid.m**

function g = sigmoid(z)

%SIGMOID Compute sigmoid functoon

% J = SIGMOID(z) computes the sigmoid of z.

% You need to return the following variables correctly

g = zeros(size(z));

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the sigmoid of each value of z. (z can be a matrix,

% vector or scalar).

% add by Chang Liu, for calculating th sigmoid function of each element

g = 1./(1+exp(-z));

% =============================================================

end

**costFunction.m**

function [J, grad] = costFunction(theta, X, y)

%COSTFUNCTION Compute cost and gradient for logistic regression

% J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the

% parameter for logistic regression and the gradient of the cost

% w.r.t. to the parameters.

% Initialize some useful values

m = length(y); % number of training examples

% You need to return the following variables correctly

J = 0;

grad = zeros(size(theta));

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the cost of a particular choice of theta.

% You should set J to the cost.

% Compute the partial derivatives and set grad to the partial

% derivatives of the cost w.r.t. each parameter in theta

%

% Note: grad should have the same dimensions as theta

%

% Chang Liu, for calculating the cost function

temp1 = -1 \* (y .\* log(sigmoid(X \* theta)));

temp2 = (1 - y) .\* log(1 - sigmoid(X \* theta));

J = sum(temp1 - temp2) / m;

% for calculating the gradient

% Note: multiple xij is calculate the corresponding value, so we

% use the X' to get the correct value

grad = (X' \* (sigmoid(X \* theta) - y)) \* (1/m);

% =============================================================

end

**predict.m**

function p = predict(theta, X)

%PREDICT Predict whether the label is 0 or 1 using learned logistic

%regression parameters theta

% p = PREDICT(theta, X) computes the predictions for X using a

% threshold at 0.5 (i.e., if sigmoid(theta'\*x) >= 0.5, predict 1)

m = size(X, 1); % Number of training examples

% You need to return the following variables correctly

p = zeros(m, 1);

% ====================== YOUR CODE HERE ======================

% Instructions: Complete the following code to make predictions using

% your learned logistic regression parameters.

% You should set p to a vector of 0's and 1's

%

result = sigmoid(X \* theta);

for i=1:size(result)

if result(i) >= 0.5

p(i) = 1;

else

p(i) = 0;

end

end

% ======================================================================

end

**costFunctionReg.m**

function [J, grad] = costFunctionReg(theta, X, y, lambda)

%COSTFUNCTIONREG Compute cost and gradient for logistic regression with regularization

% J = COSTFUNCTIONREG(theta, X, y, lambda) computes the cost of using

% theta as the parameter for regularized logistic regression and the

% gradient of the cost w.r.t. to the parameters.

% Initialize some useful values

m = length(y); % number of training examples

% You need to return the following variables correctly

J = 0;

grad = zeros(size(theta));

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the cost of a particular choice of theta.

% You should set J to the cost.

% Compute the partial derivatives and set grad to the partial

% derivatives of the cost w.r.t. each parameter in theta

% calculate similar the value of J, theta as before

temp1 = -1 \* (y .\* log(sigmoid(X \* theta)));

temp2 = (1 - y) .\* log(1 - sigmoid(X \* theta));

% here we add the correction of the theta

thetaT = theta;

thetaT(1) = 0;

correction = sum(thetaT .^ 2) \* (lambda / (2 \* m));

% new cost function and gradients by adding correction as equation shows

J = sum(temp1 - temp2) / m + correction;

grad = (X' \* (sigmoid(X \* theta) - y)) \* (1/m) + thetaT \* (lambda / m);

% =============================================================

end

**oneVsAll.m**

function [all\_theta] = oneVsAll(X, y, num\_labels, lambda)

%ONEVSALL trains multiple logistic regression classifiers and returns all

%the classifiers in a matrix all\_theta, where the i-th row of all\_theta

%corresponds to the classifier for label i

% [all\_theta] = ONEVSALL(X, y, num\_labels, lambda) trains num\_labels

% logisitc regression classifiers and returns each of these classifiers

% in a matrix all\_theta, where the i-th row of all\_theta corresponds

% to the classifier for label i

% Some useful variables

m = size(X, 1);

n = size(X, 2);

% You need to return the following variables correctly

all\_theta = zeros(num\_labels, n + 1);

% Add ones to the X data matrix

X = [ones(m, 1) X];

% ====================== YOUR CODE HERE ======================

% Instructions: You should complete the following code to train num\_labels

% logistic regression classifiers with regularization

% parameter lambda.

%

% Hint: theta(:) will return a column vector.

%

% Hint: You can use y == c to obtain a vector of 1's and 0's that tell use

% whether the ground truth is true/false for this class.

%

% Note: For this assignment, we recommend using fmincg to optimize the cost

% function. It is okay to use a for-loop (for c = 1:num\_labels) to

% loop over the different classes.

%

% fmincg works similarly to fminunc, but is more efficient when we

% are dealing with large number of parameters.

%

% Example Code for fmincg:

%

% % Set Initial theta

% initial\_theta = zeros(n + 1, 1);

%

% % Set options for fminunc

% options = optimset('GradObj', 'on', 'MaxIter', 50);

%

% % Run fmincg to obtain the optimal theta

% % This function will return theta and the cost

% [theta] = ...

% fmincg (@(t)(lrCostFunction(t, X, (y == c), lambda)), ...

% initial\_theta, options);

%

for k = 1:num\_labels

init\_theta = zeros(n + 1, 1);

options = optimset('GradObj', 'on', 'MaxIter', 50);

[theta] = fmincg (@(t)(lrCostFunction(t, X, (y == k), lambda)), init\_theta, options);

all\_theta(k,:) = theta';

end;

% =====================================================================

end

**lrCostFunction.m**

function [J, grad] = lrCostFunction(theta, X, y, lambda)

%LRCOSTFUNCTION Compute cost and gradient for logistic regression with

%regularization

% J = LRCOSTFUNCTION(theta, X, y, lambda) computes the cost of using

% theta as the parameter for regularized logistic regression and the

% gradient of the cost w.r.t. to the parameters.

% Initialize some useful values

m = length(y); % number of training examples

% You need to return the following variables correctly

J = 0;

grad = zeros(size(theta));

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the cost of a particular choice of theta.

% You should set J to the cost.

% Compute the partial derivatives and set grad to the partial

% derivatives of the cost w.r.t. each parameter in theta

%

% Hint: The computation of the cost function and gradients can be

% efficiently vectorized. For example, consider the computation

%

% sigmoid(X \* theta)

%

% Each row of the resulting matrix will contain the value of the

% prediction for that example. You can make use of this to vectorize

% the cost function and gradient computations.

%

% Hint: When computing the gradient of the regularized cost function,

% there're many possible vectorized solutions, but one solution

% looks like:

% grad = (unregularized gradient for logistic regression)

% temp = theta;

% temp(1) = 0; % because we don't add anything for j = 0

% grad = grad + YOUR\_CODE\_HERE (using the temp variable)

%

sig = sigmoid(X \* theta);

cost = -y .\* log(sig) - (1 - y) .\* log(1 - sig);

thetaNoZero = [ [ 0 ]; theta([2:length(theta)]) ];

J = (1 / m) \* sum(cost) + (lambda / (2 \* m)) \* sum(thetaNoZero .^ 2);

grad = (1 / m) .\* (X' \* (sig - y)) + (lambda / m) \* thetaNoZero;

% =============================================================

grad = grad(:);

end

**predictOneVsAll.m**

function p = predictOneVsAll(all\_theta, X)

%PREDICT Predict the label for a trained one-vs-all classifier. The labels

%are in the range 1..K, where K = size(all\_theta, 1).

% p = PREDICTONEVSALL(all\_theta, X) will return a vector of predictions

% for each example in the matrix X. Note that X contains the examples in

% rows. all\_theta is a matrix where the i-th row is a trained logistic

% regression theta vector for the i-th class. You should set p to a vector

% of values from 1..K (e.g., p = [1; 3; 1; 2] predicts classes 1, 3, 1, 2

% for 4 examples)

m = size(X, 1);

num\_labels = size(all\_theta, 1);

% You need to return the following variables correctly

p = zeros(size(X, 1), 1);

% Add ones to the X data matrix

X = [ones(m, 1) X];

% ====================== YOUR CODE HERE ======================

% Instructions: Complete the following code to make predictions using

% your learned logistic regression parameters (one-vs-all).

% You should set p to a vector of predictions (from 1 to

% num\_labels).

%

% Hint: This code can be done all vectorized using the max function.

% In particular, the max function can also return the index of the

% max element, for more information see 'help max'. If your examples

% are in rows, then, you can use max(A, [], 2) to obtain the max

% for each row.

%

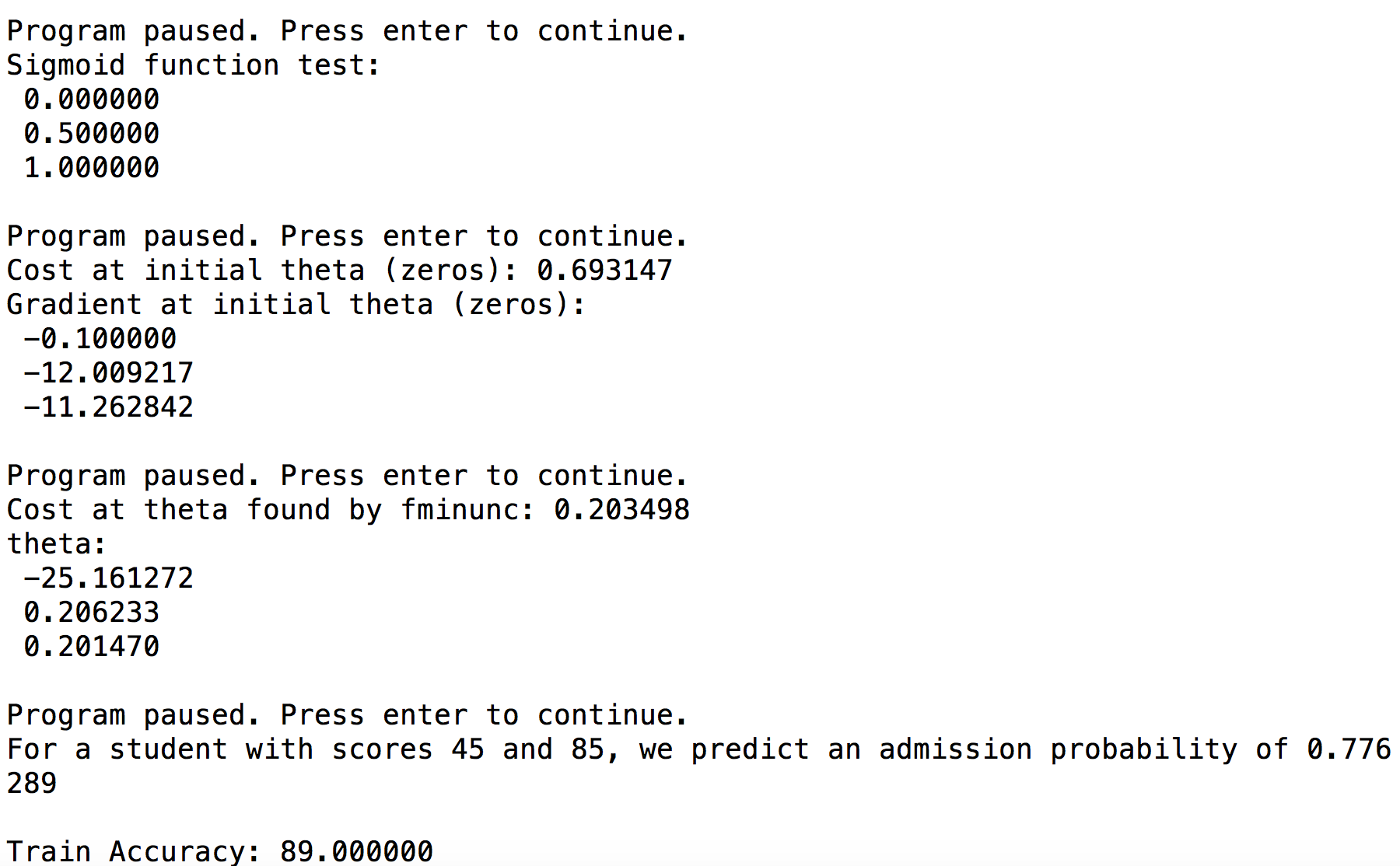
[maxVal, index] = max(X \* all\_theta', [], 2);

p = index;

% ======================================================================

end

**lda.m**

Screenshot:

