%% Machine Learning - Logistic Regression

%% Initialization

clear ; close all; clc

%% Load Data

% The first two columns contains the exam scores and the third column

% contains the label.

data = load('ex2data1.txt');

X = data(:, [1, 2]); %each training example includes scores of 2 exams

y = data(:, 3); % includes whether student gets admit or not.

%% ==================== Part 1: Plotting ====================

plotData(X, y);

hold on;

xlabel('Exam 1 score')

ylabel('Exam 2 score')

legend('Admitted', 'Not admitted')

hold off;

%% ==================== Part 2: SIGMOID Function ====================

function g = sigmoid(z)

g = zeros(size(z));

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

end

%% ============ Part 3: Compute Cost and Gradient ============

[m, n] = size(X);

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

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

[cost, grad] = costFunction(initial\_theta, X, y);

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

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

J = 0;

grad = zeros(size(theta));

Htheta=sigmoid(theta'\*X');

J=(1/m)\*(-1\*(y'\*log(Htheta'))-1\*(1-y')\*log(1-Htheta'));

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

end

fprintf('Cost at initial theta (zeros): %f\n', cost);

fprintf('Expected cost (approx): 0.693\n');

fprintf('Gradient at initial theta (zeros): \n');

fprintf(' %f \n', grad);

fprintf('Expected gradients (approx):\n -0.1000\n -12.0092\n -11.2628\n');

test\_theta = [-24; 0.2; 0.2];

[cost, grad] = costFunction(test\_theta, X, y);

fprintf('\nCost at test theta: %f\n', cost);

fprintf('Expected cost (approx): 0.218\n');

fprintf('Gradient at test theta: \n');

fprintf(' %f \n', grad);

fprintf('Expected gradients (approx):\n 0.043\n 2.566\n 2.647\n');

fprintf('\nProgram paused. Press enter to continue.\n');

pause;

%% ============= Part 4: Optimizing using fminunc =============

% Set options for fminunc

options = optimset('GradObj', 'on', 'MaxIter', 400);

% This function will return theta and the cost

[theta, cost] = ...

fminunc(@(t)(costFunction(t, X, y)), initial\_theta, options);

% Print theta to screen

fprintf('Cost at theta found by fminunc: %f\n', cost);

fprintf('Expected cost (approx): 0.203\n');

fprintf('theta: \n');

fprintf(' %f \n', theta);

fprintf('Expected theta (approx):\n');

fprintf(' -25.161\n 0.206\n 0.201\n');

% Plot Boundary

plotDecisionBoundary(theta, X, y);

function plotDecisionBoundary(theta, X, y)

% Plot Data

plotData(X(:,2:3), y);

hold on

if size(X, 2) <= 3

% Only need 2 points to define a line, so choose two endpoints

plot\_x = [min(X(:,2))-2, max(X(:,2))+2];

% Calculate the decision boundary line

plot\_y = (-1./theta(3)).\*(theta(2).\*plot\_x + theta(1));

% Plot, and adjust axes for better viewing

plot(plot\_x, plot\_y)

% Legend, specific for the exercise

legend('Admitted', 'Not admitted', 'Decision Boundary')

axis([30, 100, 30, 100])

else

% Here is the grid range

u = linspace(-1, 1.5, 50);

v = linspace(-1, 1.5, 50);

z = zeros(length(u), length(v));

% Evaluate z = theta\*x over the grid

for i = 1:length(u)

for j = 1:length(v)

z(i,j) = mapFeature(u(i), v(j))\*theta;

end

end

z = z'; % important to transpose z before calling contour

% Plot z = 0

% Notice you need to specify the range [0, 0]

contour(u, v, z, [0, 0], 'LineWidth', 2)

end

hold off

end

% Put some labels

hold on;

xlabel('Exam 1 score')

ylabel('Exam 2 score')

legend('Admitted', 'Not admitted')

hold off;

fprintf('\nProgram paused. Press enter to continue.\n');

pause;

%% ============== Part 5: Predict and Accuracies ==============

prob = sigmoid([1 45 85] \* theta);

fprintf(['For a student with scores 45 and 85, we predict an admission ' ...

'probability of %f\n'], prob);

fprintf('Expected value: 0.775 +/- 0.002\n\n');

% Compute accuracy on our training set

p = predict(theta, X);

function p = predict(theta, X)

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

p = zeros(m, 1);

p = round(sigmoid(X\*theta));

end

fprintf('Train Accuracy: %f\n', mean(double(p == y)) \* 100);

fprintf('Expected accuracy (approx): 89.0\n');

fprintf('\n');