**EX2.m – Regularized Logistic Regression**

clear ; close all; clc

data = load('ex2data2.txt');

X = data(:, [1, 2]); y = data(:, 3);

%% =========== Part 1: Regularized Logistic Regression ============

X = mapFeature(X(:,1), X(:,2));

initial\_theta = zeros(size(X, 2), 1);

lambda = 1;

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

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

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

pause;

%% ============= Part 2: Regularization and Accuracies =============

initial\_theta = zeros(size(X, 2), 1);

lambda = 1;

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

[theta, J, exit\_flag] = ...

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

plotDecisionBoundary(theta, X, y);

hold on;

title(sprintf('lambda = %g', lambda))

xlabel('Microchip Test 1')

ylabel('Microchip Test 2')

legend('y = 1', 'y = 0', 'Decision boundary')

hold off;

p = predict(theta, X);

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

function out = **mapFeature**(X1, X2)

% MAPFEATURE Feature mapping function to polynomial features

% MAPFEATURE(X1, X2) maps the two input features

% to quadratic features used in the regularization exercise.

% Returns a new feature array with more features, comprising of

% X1, X2, X1.^2, X2.^2, X1\*X2, X1\*X2.^2, etc..

degree = 6;

out = ones(size(X1(:,1)));

for i = 1:degree

for j = 0:i

out(:, end+1) = (X1.^(i-j)).\*(X2.^j);

end

end

end

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

%COSTFUNCTIONREG Compute cost and gradient for logistic regression with regularization

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

J = 0;

grad = zeros(size(theta));

h\_theta = sigmoid(X\*theta);

J = -(1./m)\*sum(y.\*log(h\_theta) + (1-y).\*log(1-h\_theta)) + lambda \* (theta'\*theta) / (2\*m);

grad(1) = (1./m)\*sum(X(:,1).\*(h\_theta-y));

for i=2:size(theta,1)

grad(i) = (1./m)\*sum(X(:,i).\*(h\_theta-y)) + (lambda/m)\*theta(i);

end

end

function **plotData**(X, y)

%PLOTDATA Plots the data points X and y into a new figure

% PLOTDATA(x,y) plots the data points with + for the positive examples

% and o for the negative examples. X is assumed to be a Mx2 matrix.

figure; hold on;

pos = find(y==1); neg = find(y == 0);

plot(X(pos, 1), X(pos, 2), 'k+','LineWidth', 2, 'MarkerSize', 7);

plot(X(neg, 1), X(neg, 2), 'ko', 'MarkerFaceColor', 'y', 'MarkerSize', 7);

hold off;

end

function **plotDecisionBoundary**(theta, X, y)

%PLOTDECISIONBOUNDARY Plots the data points X and y into a new figure with

%the decision boundary defined by theta

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

hold on

if size(X, 2) <= 3

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

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

plot(plot\_x, plot\_y)

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

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

else

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

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

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

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

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

end

hold off

end