%% ================== Generate and Plot Training Set ================== %%

clear all; close all; clc;

n = 2; % number of feature dimensions

N = 100; % number of iid samples

% parallel distributions

mu(:,1) = [2;0]; Sigma(:,:,2) = [2 0.5;0.5 30];

mu(:,2) = [-2;0]; Sigma(:,:,1) = [2 0.5;0.5 30];

%mu(:,1) = [3;0]; Sigma(:,:,1) = [5 0.1;0.1 .5];

%mu(:,2) = [0;0]; Sigma(:,:,2) = [.5 0.1;0.1 5];

% Class priors for class 0 and 1 respectively

p = [0.9,0.1];

% Generating true class labels

label = (rand(1,N) >= p(1))';

Nc = [length(find(label==0)),length(find(label==1))];

% Draw samples from each class pdf

x = zeros(N,n);

for L = 0:1

x(label==L,:) = mvnrnd(mu(:,L+1),Sigma(:,:,L+1),Nc(L+1));

end

%Plot samples with true class labels

figure(1);

plot(x(label==0,1),x(label==0,2),'o',x(label==1,1),x(label==1,2),'+');

legend('Class 0','Class 1'); title('Training Data and True Class Labels');

xlabel('x\_1'); ylabel('x\_2'); hold on;

%% ======================== Logistic Regression ======================= %%

% Initialize fitting parameters

x = [ones(N, 1) x];

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

label=double(label);

% Compute gradient descent to get theta values

[theta, cost] = gradient\_descent(x,N,label,initial\_theta,1,1000);

[theta2, cost2] = fminsearch(@(t)(cost\_func(t, x, label, N)), initial\_theta);

% Choose points to draw boundary line

plot\_x1 = [min(x(:,2))-2, max(x(:,2))+2];

plot\_x2(1,:) = (-1./theta(3)).\*(theta(2).\*plot\_x1 + theta(1));

plot\_x2(2,:) = (-1./theta2(3)).\*(theta2(2).\*plot\_x1 + theta2(1)); % fminsearch

% Plot decision boundary

plot(plot\_x1, plot\_x2(1,:), plot\_x1, plot\_x2(2,:));

axis([plot\_x1(1), plot\_x1(2), min(x(:,3))-2, max(x(:,3))+2]);

legend('Class 0', 'Class 1', ' Classifier (from scratch)', 'Classifier (fminsearch)');

% Plot cost function

figure(2); plot(cost);

title('Calculated Cost');

xlabel('Iteration number'); ylabel('Cost');

%% ====================== Generate Test Data Set ====================== %%

N\_test = 10000;

% Generating true class labels

label\_test = (rand(1,N\_test) >= p(1))';

Nc\_test = [length(find(label\_test==0)),length(find(label\_test==1))];

% Draw samples from each class pdf

x\_test = zeros(N\_test,n);

for L = 0:1

x\_test(label\_test==L,:) = mvnrnd(mu(:,L+1),Sigma(:,:,L+1),Nc\_test(L+1));

end

%% ========================= Test Classifier ========================== %%

% Coefficients for decision boundary line equation

coeff(1,:) = polyfit([plot\_x1(1), plot\_x1(2)], [plot\_x2(1,1), plot\_x2(1,2)], 1);

coeff(2,:) = polyfit([plot\_x1(1), plot\_x1(2)], [plot\_x2(2,1), plot\_x2(2,2)], 1); %fminsearch

% Decide based on which side of the line each point is on

for i = 1:2

if coeff(i,1) >= 0

decision(:,i) = (coeff(i,1).\*x\_test(:,1) + coeff(i,2)) < x\_test(:,2);

else

decision(:,i) = (coeff(i,1).\*x\_test(:,1) + coeff(i,2)) > x\_test(:,2);

end

end

error1 = plot\_test\_data(decision(:,1), label\_test, Nc\_test, p, 3, x\_test, plot\_x1, plot\_x2(1,:));

title('Test Data Classification (from scratch)');

fprintf('Total error (classifier from scratch): %.2f%%\n',error1);

error2 = plot\_test\_data(decision(:,2), label\_test, Nc\_test, p, 4, x\_test, plot\_x1, plot\_x2(2,:));

title('Test Data Classification (using fminsearch)');

fprintf('Total error (classifier using fminsearch): %.2f%%\n',error2);

%% ============================ Functions ============================= %%

function [theta, cost] = gradient\_descent(x, N, label, theta, alpha, num\_iters)

cost = zeros(num\_iters, 1);

for i = 1:num\_iters % while norm(cost\_gradient) > threshold

h = 1 ./ (1 + exp(-x\*theta)); % Sigmoid function

cost(i) = (-1/N)\*((sum(label' \* log(h)))+(sum((1-label)' \* log(1-h))));

cost\_gradient = (1/N)\*(x' \* (h - label));

theta = theta - (alpha.\*cost\_gradient); % Update theta

end

end

function cost = cost\_func(theta, x, label,N)

h = 1 ./ (1 + exp(-x\*theta)); % Sigmoid function

cost = (-1/N)\*((sum(label' \* log(h)))+(sum((1-label)' \* log(1-h))));

end

function error = plot\_test\_data(decision, label, Nc, p, fig, x, plot\_x1, plot\_x2)

ind00 = find(decision==0 & label==0); % true negative

ind10 = find(decision==1 & label==0); p10 = length(ind10)/Nc(1); % false positive

ind01 = find(decision==0 & label==1); p01 = length(ind01)/Nc(2); % false negative

ind11 = find(decision==1 & label==1); % true positive

error = (p10\*p(1) + p01\*p(2))\*100;

% Plot decisions and decision boundary

figure(fig);

plot(x(ind00,1),x(ind00,2),'og'); hold on,

plot(x(ind10,1),x(ind10,2),'or'); hold on,

plot(x(ind01,1),x(ind01,2),'+r'); hold on,

plot(x(ind11,1),x(ind11,2),'+g'); hold on,

plot(plot\_x1, plot\_x2);

axis([plot\_x1(1), plot\_x1(2), min(x(:,2))-2, max(x(:,2))+2])

legend('Class 0 Correct Decisions','Class 0 Wrong Decisions','Class 1 Wrong Decisions','Class 1 Correct Decisions','Classifier');

end