Barry Wu
Machine Intelligence
Homework 4

Problem 1

Below is the new costFunctionLogisticRegression.m where the gradient and the cost is not calculated using a loop.

```
function [J, grad] = costFunctionLogisticRegression(theta, X, y, lambda)
% costFunctionLogisticRegression Compute cost and gradient for logistic regression with regularization
%    [J, grad] = costFunctionLogisticRegression(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.

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

% pre-allocate space for gradient
%grad = zeros(size(theta));
%dont need a vector of 0s

% Logistic Regression Cost Function
J = (1/n)*sum(-y.*(log(sigmoid(X*theta))) -(1-y).*log(1-(sigmoid(X*theta)))) + (lambda/(2*n))*sum(theta(2:end).^2);

theta_0 = theta;
theta_0(1) = 0;
grad = (1/n)*X'*(sigmoid(X*theta)-y) + (lambda/n)*theta_0;
end
```

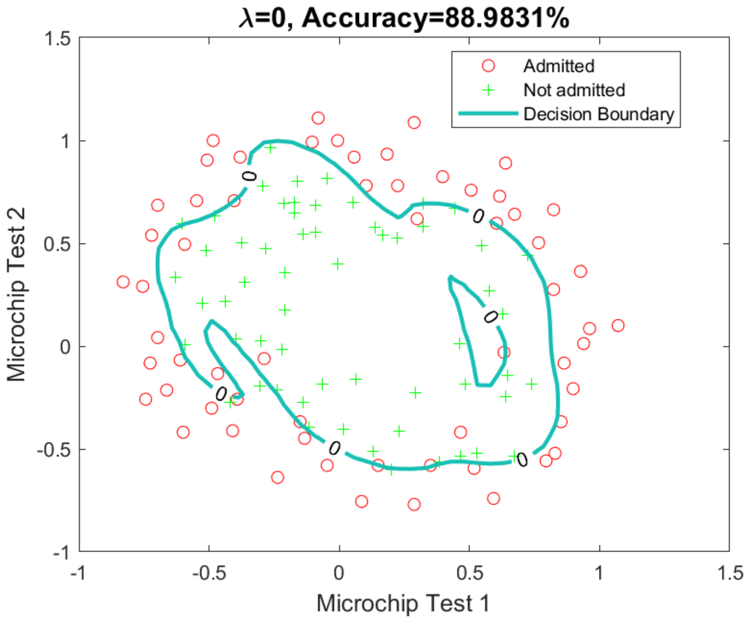
Listing 1: costFunctionLogisticRegression.m

Problem 2

λ=0

```
%% lambda = 0
lambda_0 = 0;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_0, J_0, exit_flag_0] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_0)), initial_theta, options);
plotDecisionBoundary(theta_0, Xdata, y, degree);

TPTN_0 = 0;
TPTNFPFN_0 = 0;

y_hat_0 = sigmoid(Xdata*theta_0);
for index_y_hat_0 = 1:size(y_hat_0)
    if y_hat_0(index_y_hat_0) >= 0.5
        y_hat_0(index_y_hat_0) = 1;
    else
        y_hat_0(index_y_hat_0) = 0;
    end
    if y_hat_0(index_y_hat_0) == y(index_y_hat_0)
        TPTN_0 = TPTN_0 + 1;
        TPTNFPFN_0 = TPTNFPFN_0 + 1;
    else
        TPTNFPFN_0 = TPTNFPFN_0 + 1;
    end
end

accurracy_0 = TPTN_0/TPTNFPFN_0;

title_string = ['{\lambda}=', num2str(lambda_0),', Accuracy=', num2str(accurracy_0*100),'%'];
title(title_string,'fontsize',14);
xlabel('Microchip Test 1','fontsize',12)
ylabel('Microchip Test 2','fontsize',12)
legend('Admitted', 'Not admitted', 'Decision Boundary','location', 'best')

print -dpng hwk4_problem2_lambda_0_plot.png
```

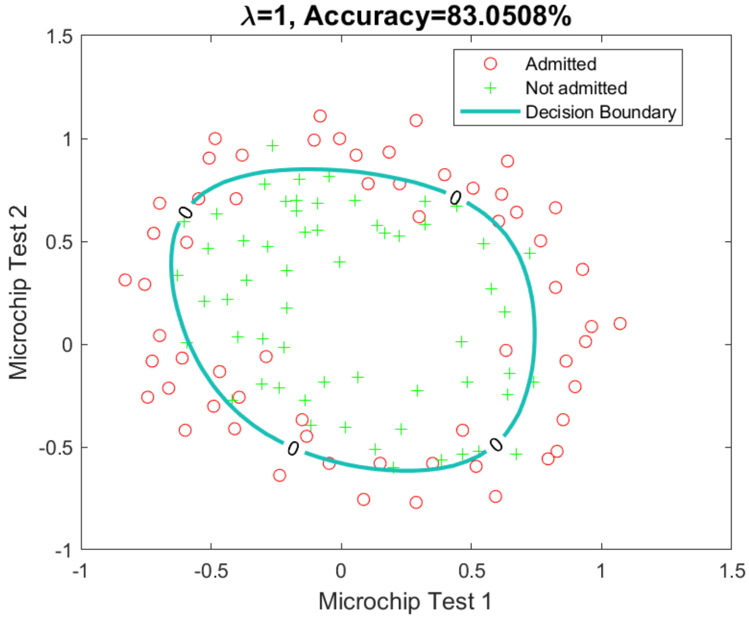
Listing 2: problem_2.m



λ=1

```
%% lambda = 1
lambda_1 = 1;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_1, J_1, exit_flag_1] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_1)), initial_theta, options);
plotDecisionBoundary(theta_1, Xdata, y, degree);

TPTN_1 = 0;
TPTNFPFN_1 = 0;

y_hat_1 = sigmoid(Xdata*theta_1);
for index_y_hat_1 = 1:size(y_hat_1)
    if y_hat_1(index_y_hat_1) >= 0.5
        y_hat_1(index_y_hat_1) = 1;
    else
        y_hat_1(index_y_hat_1) = 0;
    end
    if y_hat_1(index_y_hat_1) == y(index_y_hat_1)
        TPTN_1 = TPTN_1 + 1;
        TPTNFPFN_1 = TPTNFPFN_1 + 1;
    else
        TPTNFPFN_1 = TPTNFPFN_1 + 1;
    end
end

accurracy_1 = TPTN_1/TPTNFPFN_1;

title_string = ['{\lambda}=', num2str(lambda_1),', Accuracy=', num2str(accurracy_1 * 100),'%'];
title(title_string,'fontsize',14);
xlabel('Microchip Test 1','fontsize',12)
ylabel('Microchip Test 2','fontsize',12)
legend('Admitted', 'Not admitted', 'Decision Boundary','location', 'best')

print -dpng hwk4_problem2_lambda_1_plot.png
```

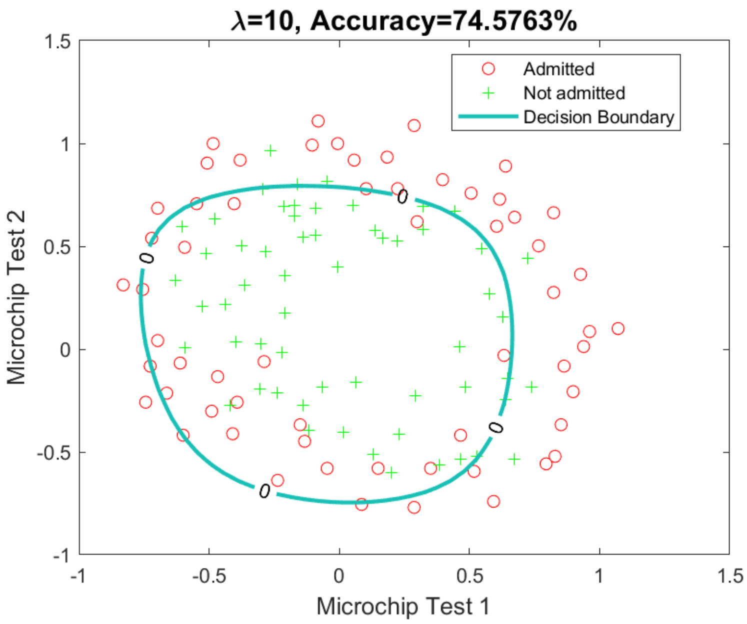
Listing 3: problem_2.m



λ=10

```
%% lambda = 10
lambda_10 = 10;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_10, J_10, exit_flag_10] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_10)), initial_theta, options);
plotDecisionBoundary(theta_10, Xdata, y, degree);

TPTN_10 = 0;
TPTNFPFN_10 = 0;

y_hat_10 = sigmoid(Xdata*theta_10);
for index_y_hat_10 = 1:size(y_hat_10)
    if y_hat_10(index_y_hat_10) >= 0.5
        y_hat_10(index_y_hat_10) = 1;
    else
        y_hat_10(index_y_hat_10) = 0;
    end
    if y_hat_10(index_y_hat_10) == y(index_y_hat_10)
        TPTN_10 = TPTN_10 + 1;
        TPTNFPFN_10 = TPTNFPFN_10 + 1;
    else
        TPTNFPFN_10 = TPTNFPFN_10 + 1;
    end
end

accurracy_10 = TPTN_10/TPTNFPFN_10;

title_string = ['{\lambda}=', num2str(lambda_10),', Accuracy=', num2str(accurracy_10 * 100), '%'];
title(title_string,'fontsize',14);
xlabel('Microchip Test 1','fontsize',12)
ylabel('Microchip Test 2','fontsize',12)
legend('Admitted', 'Not admitted', 'Decision Boundary','location', 'best')

print -dpng hwk4_problem2_lambda_10_plot.png
```

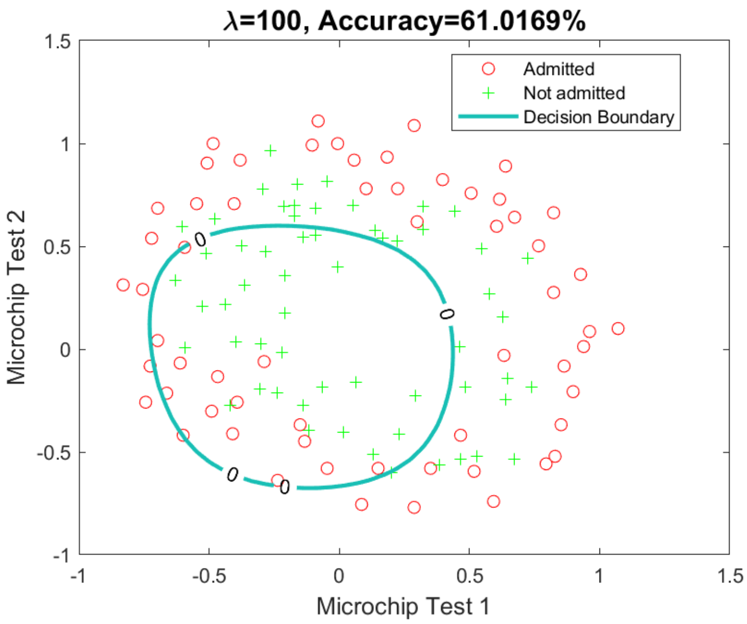
Listing 4: problem_2.m



λ=100

```
%% lambda = 100
lambda_100 = 100;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_100, J_100, exit_flag_100] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_100)), initial_theta,
    options);
plotDecisionBoundary(theta_100, Xdata, y, degree);

TPTN_100 = 0;
TPTNFPFN_100 = 0;

y_hat_100 = sigmoid(Xdata*theta_100);
for index_y_hat_100 = 1:size(y_hat_100)
    if y_hat_100(index_y_hat_100) >= 0.5
        y_hat_100(index_y_hat_100) = 1;
    else
        y_hat_100(index_y_hat_100) = 0;
    end
    if y_hat_100(index_y_hat_100) == y(index_y_hat_100)
        TPTN_100 = TPTN_100 + 1;
        TPTNFPFN_100 = TPTNFPFN_100 + 1;
    else
        TPTNFPFN_100 = TPTNFPFN_100 + 1;
    end
end

accurracy_100 = TPTN_100/TPTNFPFN_100;

title_string = ['{\lambda}=', num2str(lambda_100),', Accuracy=', num2str(accurracy_100 *
    100),'%'];
title(title_string,'fontsize',14);
xlabel('Microchip Test 1','fontsize',12)
ylabel('Microchip Test 2','fontsize',12)
legend('Admitted', 'Not admitted', 'Decision Boundary','location', 'best')

print -dpng hwk4_problem2_lambda_100_plot.png
```

Listing 5: problem_2.m

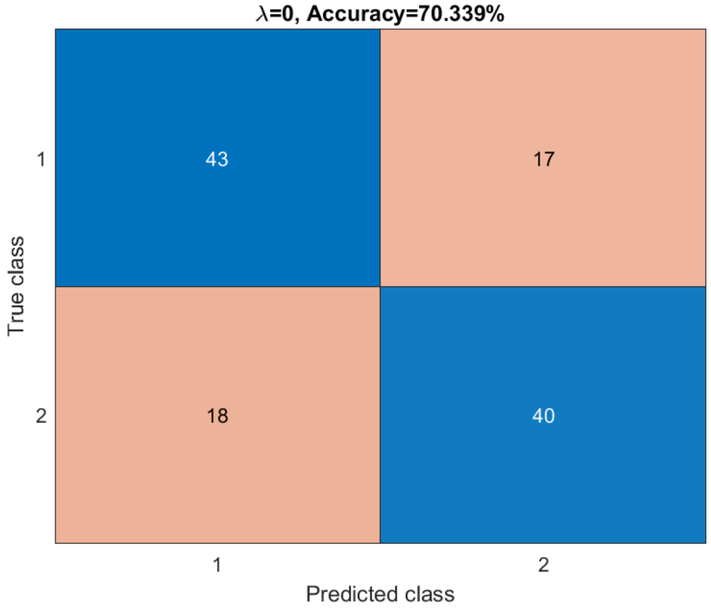

λ=100, Accuracy=61.0169%
Admitted
Not admitted
Decision Boundary
Microchip Test 1
Microchip Test 2
-1
-0.5
0
0.5
1
1.5
0

Problem 3

λ=0

```
%% lambda = 0
lambda_0 = 0;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_0, J_0, exit_flag_0] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_0)), initial_theta, options);

TPTN_0 = 0;
TPTNFPFN_0 = 0;

y_hat_0 = sigmoid(Xdata*theta_0);
for index_y_hat_0 = 1:size(y_hat_0)
    if y_hat_0(index_y_hat_0) >= 0.5
        y_hat_0(index_y_hat_0) = 1;
    else
        y_hat_0(index_y_hat_0) = 0;
    end
    if y_hat_0(index_y_hat_0) == y(index_y_hat_0)
        TPTN_0 = TPTN_0 + 1;
        TPTNFPFN_0 = TPTNFPFN_0 + 1;
    else
        TPTNFPFN_0 = TPTNFPFN_0 + 1;
    end
end

accurracy_0 = TPTN_0/TPTNFPFN_0;

% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
method='LogisticRegression';
%lambda=1
for i = 1:numberOfFolds
TestIndex_0 = find(CVindex == i);
TrainIndex_0 = find(CVindex ~= i);
TrainDataCV_0 = Xdata(TrainIndex_0,:);
TrainDataGT_0 = y(TrainIndex_0);
TestDataCV_0 = Xdata(TestIndex_0,:);
TestDataGT_0 = y(TestIndex_0);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
    case 'LogisticRegression'
    % for Logistic Regression, we need to solve for theta
    % Insert code here to solve for theta...
    [theta_0_train, J_0_train, exit_flag_0_train] = ...
        fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_0, TrainDataGT_0, lambda_0)), initial_theta, options);
    %plotDecisionBoundary(theta_0_train, TrainDataCV_0, TrainDataGT_0, degree);
    % Using TestDataCV, compute testing set prediction using
    % the model created
    % for Logistic Regression, the model is theta
    % Insert code here to see how well theta works...
    TestDataPred_0 = TestDataCV_0 * theta_0_train > 0.5;
    case 'KNN'
        disp('KNN not implemented yet')
    otherwise
        error('Unknown classification method')
end
predictionLabels_0(TestIndex_0,:) = double(TestDataPred_0);
end
confusionMatrix_0 = confusionmat(y,predictionLabels_0);
accuracy_0_cf = sum(diag(confusionMatrix_0))/sum(sum(confusionMatrix_0));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method,lambda_0,accuracy_0_cf*100));
%fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix_0);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix_0(i,j));
end
fprintf('\n');
end

figure
title_string_0 = ['{\lambda}=', num2str(lambda_0),', Accuracy=', num2str(accuracy_0_cf * 100),'%'];
confusionchart(confusionMatrix_0, 'Title', title_string_0);
print -dpng hwk4_problem3_lambda_0_plot.png
```

Listing 6: problem_3.m

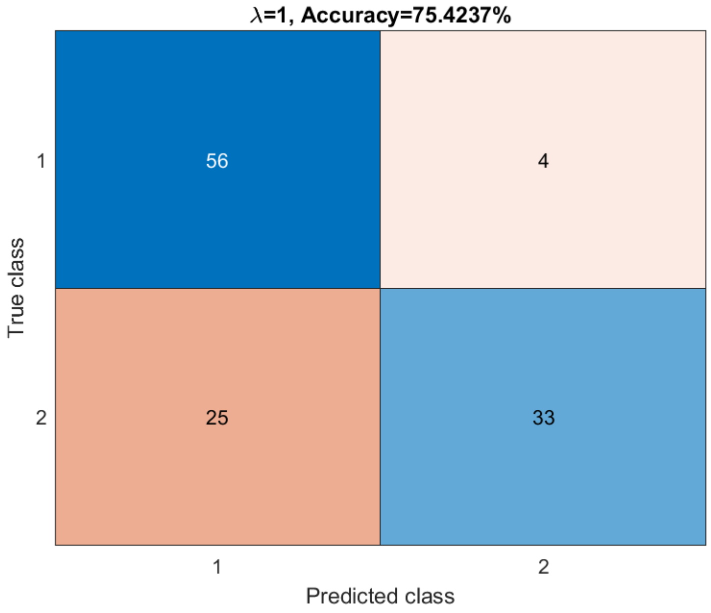


λ=1

```
%% lambda = 1
lambda_1 = 1;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_1, J_1, exit_flag_1] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_1)), initial_theta, options);
%plotDecisionBoundary(theta_1, Xdata, y, degree);

TPTN_1 = 0;
```

```
TPTNFPFN_1 = 0;

y_hat_1 = sigmoid(Xdata*theta_1);
for index_y_hat_1 = 1:size(y_hat_1)
    if y_hat_1(index_y_hat_1) >= 0.5
        y_hat_1(index_y_hat_1) = 1;
    else
        y_hat_1(index_y_hat_1) = 0;
    end
    if y_hat_1(index_y_hat_1) == y(index_y_hat_1)
        TPTN_1 = TPTN_1 + 1;
        TPTNFPFN_1 = TPTNFPFN_1 + 1;
    else
        TPTNFPFN_1 = TPTNFPFN_1 + 1;
    end
end

accurracy_1 = TPTN_1/TPTNFPFN_1;

% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
method='LogisticRegression';
%lambda=1
for i = 1:numberOfFolds
TestIndex_1 = find(CVindex == i);
TrainIndex_1 = find(CVindex ~= i);
TrainDataCV_1 = Xdata(TrainIndex_1,:);
TrainDataGT_1 = y(TrainIndex_1);
TestDataCV_1 = Xdata(TestIndex_1,:);
TestDataGT_1 = y(TestIndex_1);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
    case 'LogisticRegression'
    % for Logistic Regression, we need to solve for theta
    % Insert code here to solve for theta...
    [theta_1_train, J_1_train, exit_flag_1_train] = ...
        fminunc(@(t)(costFunctionLogisticRegression(t,TrainDataCV_1, TrainDataGT_1, lambda_1)), initial_theta, options);
    %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
    % Using TestDataCV, compute testing set prediction using
    % the model created
    % for Logistic Regression, the model is theta
    % Insert code here to see how well theta works...
    TestDataPred_1 = TestDataCV_1 * theta_1_train > 0.5;
    case 'KNN'
        disp('KNN not implemented yet')
    otherwise
        error('Unknown classification method')
end
predictionLabels_1(TestIndex_1,:) = double(TestDataPred_1);
end
confusionMatrix_1 = confusionmat(y,predictionLabels_1);
accuracy_1_cf = sum(diag(confusionMatrix_1))/sum(sum(confusionMatrix_1));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method,lambda_1,accuracy_1_cf*100));
%fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix_1);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix_1(i,j));
end
fprintf('\n');
end

figure
title_string_1 = ['{\lambda}=', num2str(lambda_1),', Accuracy=', num2str(accuracy_1_cf *
    100),'%'];
confusionchart(confusionMatrix_1, 'Title', title_string_1);
print -dpng hwk4_problem3_lambda_1_plot.png
```

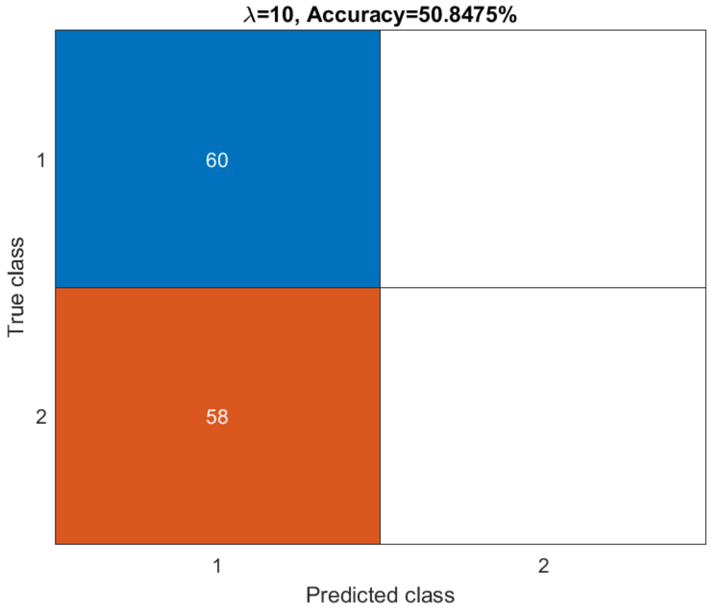
Listing 7: problem_3.m



λ=10

```
%% lambda = 10
lambda_10 = 10;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_10, J_10, exit_flag_10] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_10)), initial_theta,
    options);
%plotDecisionBoundary(theta_10, Xdata, y, degree);

TPTN_10 = 0;
TPTNFPFN_10 = 0;

y_hat_10 = sigmoid(Xdata*theta_10);
for index_y_hat_10 = 1:size(y_hat_10)
    if y_hat_10(index_y_hat_10) >= 0.5
        y_hat_10(index_y_hat_10) = 1;
    else
        y_hat_10(index_y_hat_10) = 0;
    end
    if y_hat_10(index_y_hat_10) == y(index_y_hat_10)
        TPTN_10 = TPTN_10 + 1;
        TPTNFPFN_10 = TPTNFPFN_10 + 1;
    else
        TPTNFPFN_10 = TPTNFPFN_10 + 1;
    end
end

accurracy_10 = TPTN_10/TPTNFPFN_10;

% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
method='LogisticRegression';
%lambda=1
for i = 1:numberOfFolds
TestIndex_10 = find(CVindex == i);
TrainIndex_10 = find(CVindex ~= i);
TrainDataCV_10 = Xdata(TrainIndex_10,:);
TrainDataGT_10 = y(TrainIndex_10);
TestDataCV_10 = Xdata(TestIndex_10,:);
TestDataGT_10 = y(TestIndex_10);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
    case 'LogisticRegression'
    % for Logistic Regression, we need to solve for theta
    % Insert code here to solve for theta...
    [theta_10_train, J_10_train, exit_flag_10_train] = ...
        fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_10, TrainDataGT_10, lambda_10)), initial_theta, options);
    %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
    % Using TestDataCV, compute testing set prediction using
    % the model created
    % for Logistic Regression, the model is theta
    % Insert code here to see how well theta works...
    TestDataPred_10 = TestDataCV_10 * theta_10_train > 0.5;
    case 'KNN'
        disp('KNN not implemented yet')
    otherwise
        error('Unknown classification method')
end
predictionLabels_10(TestIndex_10,:) = double(TestDataPred_10);
end
confusionMatrix_10 = confusionmat(y,predictionLabels_10);
accuracy_10_cf = sum(diag(confusionMatrix_10))/sum(sum(confusionMatrix_10));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method,lambda_10,accuracy_10_cf*100));
%fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix_10);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix_10(i,j));
end
fprintf('\n');
end

figure
title_string_10 = ['{\lambda}=', num2str(lambda_10),', Accuracy=', num2str(accuracy_10_cf*100),'%'];
confusionchart(confusionMatrix_10, 'Title', title_string_10);
print -dpng hwk4_problem3_lambda_10_plot.png
```

Listing 8: problem_3.m

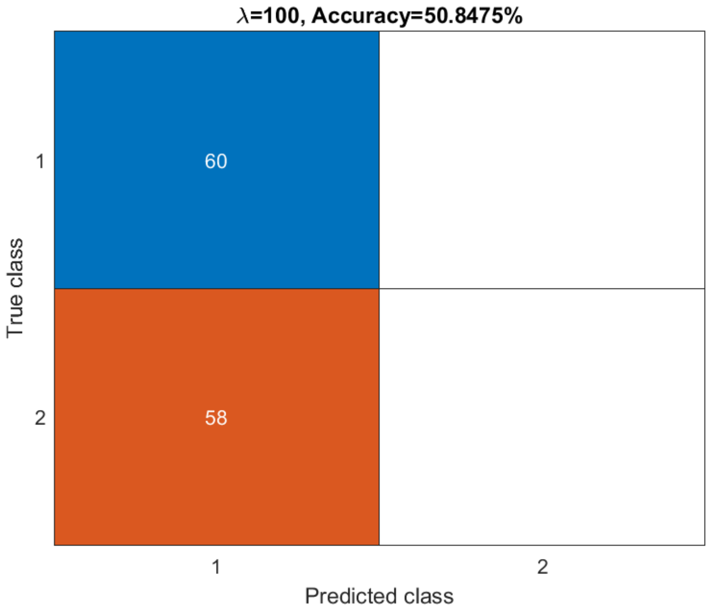


λ=100

```
%% lambda = 100
lambda_100 = 100;
% Specifying function with the @(t) allows fminunc to call our costFunction
% The t is an input argument, in this case initial_theta
[theta_100, J_100, exit_flag_100] = ...
    fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_100)), initial_theta, options);
%plotDecisionBoundary(theta_100, Xdata, y, degree);

TPTN_100 = 0;
TPTNFPFN_100 = 0;

y_hat_100 = sigmoid(Xdata*theta_100);
for index_y_hat_100 = 1:size(y_hat_100)
    if y_hat_100(index_y_hat_100) >= 0.5
        y_hat_100(index_y_hat_100) = 1;
    else
        y_hat_100(index_y_hat_100) = 0;
    end
    if y_hat_100(index_y_hat_100) == y(index_y_hat_100)
        TPTN_100 = TPTN_100 + 1;
        TPTNFPFN_100 = TPTNFPFN_100 + 1;
    else
        TPTNFPFN_100 = TPTNFPFN_100 + 1;
    end
end

accurracy_100 = TPTN_100/TPTNFPFN_100;

% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
method='LogisticRegression';
```

```
%lambda=1
for i = 1:numberOfFolds
TestIndex_100 = find(CVindex == i);
TrainIndex_100 = find(CVindex ~= i);
TrainDataCV_100 = Xdata(TrainIndex_100,:);
TrainDataGT_100 = y(TrainIndex_100);
TestDataCV_100 = Xdata(TestIndex_100,:);
TestDataGT_100 = y(TestIndex_100);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
    case 'LogisticRegression'
    % for Logistic Regression, we need to solve for theta
    % Insert code here to solve for theta...
    [theta_100_train, J_100_train, exit_flag_100_train] = ...
        fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_100, TrainDataGT_100, lambda_100)), initial_theta, options);
    %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
    % Using TestDataCV, compute testing set prediction using
    % the model created
    % for Logistic Regression, the model is theta
    % Insert code here to see how well theta works...
    TestDataPred_100 = TestDataCV_100 * theta_100_train > 0.5;
    case 'KNN'
        disp('KNN not implemented yet')
    otherwise
        error('Unknown classification method')
end
predictionLabels_100(TestIndex_100,:) = double(TestDataPred_100);
end
confusionMatrix_100 = confusionmat(y,predictionLabels_100);
accuracy_100_cf = sum(diag(confusionMatrix_100))/sum(sum(confusionMatrix_100));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method,lambda_100,accuracy_100_cf*100));
%fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix_100);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix_100(i,j));
end
fprintf('\n');
end

figure
title_string_100 = ['{\lambda}=', num2str(lambda_100),', Accuracy=', num2str(accuracy_100_cf * 100),'%'];
confusionchart(confusionMatrix_100, 'Title', title_string_100);
print -dpng hwk4_problem3_lambda_100_plot.png
```

Listing 9: problem_3.m

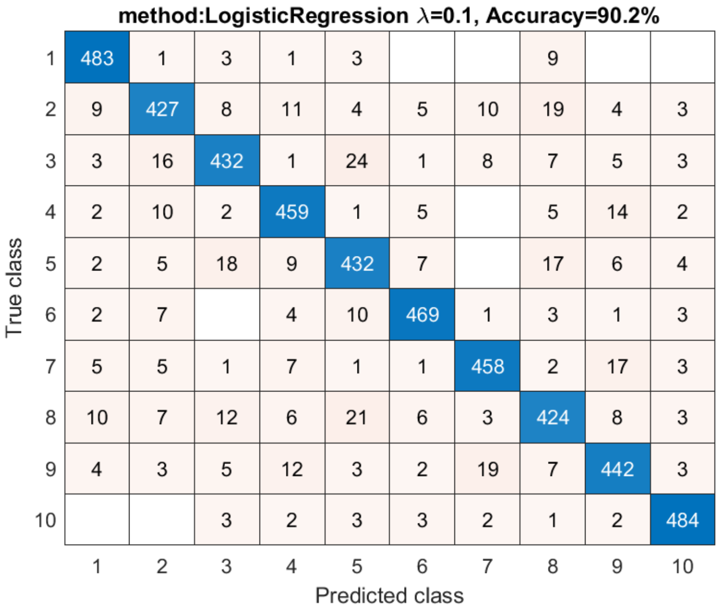

λ=100, Accuracy=50.8475%
True class
1
2
60
58
1
2
Predicted class

Problem 4

```
clear ; close all; clc
% Load Training Data- Andrew Ng Machine Learning MOOC
load('ex3data1.mat'); % training data stored in arrays X, y
n = size(X, 1);
num_labels = length(unique(y)); % 10 labels, from 1 to 10 (note ”0” is mapped to label 10)
% Randomly select 100 data points to display
rng(2000); %random number generator seed
rand_indices = randperm(n);
sel = X(rand_indices(1:100), :);
Xdata = [ones(n, 1) X];
% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
method='LogisticRegression'
lambda = 0.1;
for i = 1:numberOfFolds
TestIndex = find(CVindex == i);
TrainIndex = find(CVindex ~= i);
TrainDataCV = Xdata(TrainIndex,:);
TrainDataGT =y(TrainIndex);
TestDataCV = Xdata(TestIndex,:);
TestDataGT = y(TestIndex);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
case 'LogisticRegression'
% for Logistic Regression, we need to solve for theta
% Initialize fitting parameters
all_theta = zeros(num_labels, size(Xdata, 2));
for c=1:num_labels
% Set Initial theta
initial_theta = zeros(size(Xdata, 2), 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)(costFunctionLogisticRegression(t, TrainDataCV, (TrainDataGT == c), lambda)), ...
initial_theta, options);
all_theta(c,:) = theta;
end
% Using TestDataCV, compute testing set prediction using
% the model created
% for Logistic Regression, the model is theta
% Insert code here to see how well theta works...
all_pred = sigmoid(TestDataCV*all_theta');
[maxVal,maxIndex] = max(all_pred,[],2);
TestDataPred=maxIndex;
case 'KNN'
disp('KNN not implemented yet')
otherwise
error('Unknown classification method')
end
predictionLabels(TestIndex,:) =double(TestDataPred);
end
confusionMatrix = confusionmat(y,predictionLabels);
```

```
accuracy = sum(diag(confusionMatrix))/sum(sum(confusionMatrix));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method, lambda,accuracy*100));
fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix(i,j));
end
fprintf('\n');
end

figure
title_string = ['method:', method,' {\lambda}=', num2str(lambda),', Accuracy=', num2str(
    accuracy * 100),'%'];
confusionchart(confusionMatrix, 'Title', title_string);
print -dpng hwk4_problem4_plot.png
```

Listing 10: problem_4.m



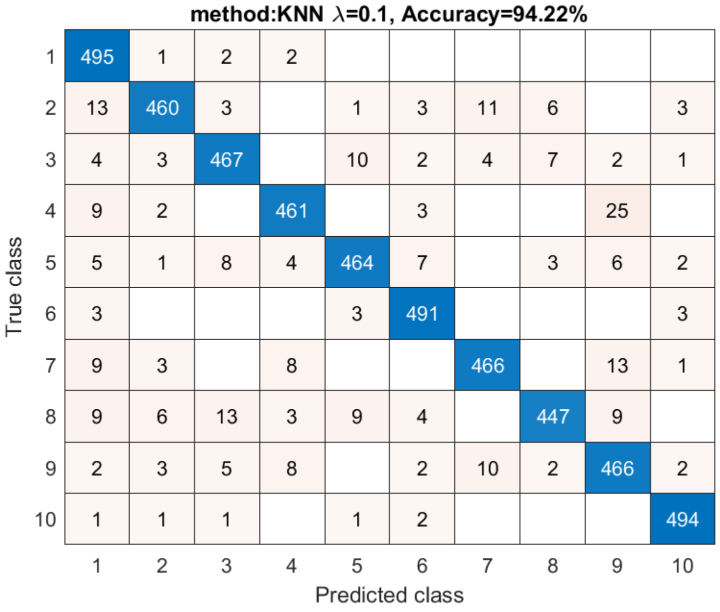
	1	2	3	4	5	6	7	8	9	10
1	483	1	3	1	3			9		
2	9	427	8	11	4	5	10	19	4	3
3	3	16	432	1	24	1	8	7	5	3
4	2	10	2	459	1	5		5	14	2
5	2	5	18	9	432	7		17	6	4
6	2	7		4	10	469	1	3	1	3
7	5	5	1	7	1	1	458	2	17	3
8	10	7	12	6	21	6	3	424	8	3
9	4	3	5	12	3	2	19	7	442	3
10			3	2	3	3	2	1	2	484

Problem 5

```
clear ; close all; clc
% Load Training Data- Andrew Ng Machine Learning MOOC
load('ex3data1.mat'); % training data stored in arrays X, y
n = size(X, 1);
num_labels = length(unique(y)); % 10 labels, from 1 to 10 (note "0" is mapped to label 10)
% Randomly select 100 data points to display
rng(2000); %random number generator seed
rand_indices = randperm(n);
sel = X(rand_indices(1:100), :);
Xdata = [ones(n, 1) X];
% the matlab functions you want to use are crossvalind.m and confusionmat.m_
% Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
% y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label for
%each dataset sample
% numberOfFolds- the number of folds for k-fold cross validation
numberOfFolds=5;
rng(2000); %random number generator seed
CVindex = crossvalind('Kfold',y, numberOfFolds);
%method='LogisticRegression'
method='KNN';
lambda = 0.1;
for i = 1:numberOfFolds
TestIndex = find(CVindex == i);
TrainIndex = find(CVindex ~= i);
TrainDataCV = Xdata(TrainIndex,:);
TrainDataGT =y(TrainIndex);
TestDataCV = Xdata(TestIndex,:);
TestDataGT = y(TestIndex);
%
%build the model using TrainDataCV and TrainDataGT
%test the built model using TestDataCV
%
switch method
case 'LogisticRegression'
% for Logistic Regression, we need to solve for theta
% Initialize fitting parameters
all_theta = zeros(num_labels, size(Xdata, 2));
for c=1:num_labels
% Set Initial theta
initial_theta = zeros(size(Xdata, 2), 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)(costFunctionLogisticRegression(t, TrainDataCV, (TrainDataGT == c), lambda)), ...
initial_theta, options);
all_theta(c,:) = theta;
end
% Using TestDataCV, compute testing set prediction using
% the model created
% for Logistic Regression, the model is theta
% Insert code here to see how well theta works...
all_pred = sigmoid(TestDataCV*all_theta');
[maxVal,maxIndex] = max(all_pred,[],2);
TestDataPred=maxIndex;
case 'KNN'
%disp('KNN not implemented yet')
k_value = 3;
Idx = knnsearch(TrainDataCV,TestDataCV, 'k', k_value);
Idx_gt = [TrainDataGT(Idx(:,1)) TrainDataGT(Idx(:,2)) TrainDataGT(Idx(:,3))];
Idx_gt_T = Idx_gt';
TestDataPred=mode(Idx_gt_T)';
```

```
otherwise
error('Unknown classificatio method')
end
predictionLabels(TestIndex,:) =double(TestDataPred);
end
confusionMatrix = confusionmat(y,predictionLabels);
accuracy = sum(diag(confusionMatrix))/sum(sum(confusionMatrix));
fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%%% \n',method, lambda,accuracy*100));
fprintf('Confusion Matrix:\n');
[r c] = size(confusionMatrix);
for i=1:r
for j=1:r
fprintf('%6d ',confusionMatrix(i,j));
end
fprintf('\n');
end

figure
title_string = ['method:', method,' {\lambda}=', num2str(lambda),', Accuracy=', num2str(accuracy * 100),'%'];
confusionchart(confusionMatrix, 'Title', title_string);
print -dpng hwk4_problem5_plot.png
```

Listing 11: problem_5.m



	1	2	3	4	5	6	7	8	9	10
1	495	1	2	2						
2	13	460	3		1	3	11	6		3
3	4	3	467		10	2	4	7	2	1
4	9	2		461		3			25	
5	5	1	8	4	464	7		3	6	2
6	3				3	491				3
7	9	3		8			466		13	1
8	9	6	13	3	9	4		447	9	
9	2	3	5	8		2	10	2	466	2
10	1	1	1		1	2				494