

## Problem 1

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

```
1 function [J, grad] = costFunctionLogisticRegression(theta, X, y, lambda)
2 % costFunctionLogisticRegression Compute cost and gradient for logistic regression with
   regularization
3 % [J, grad] = costFunctionLogisticRegression(theta, X, y, lambda) computes the cost of
   using
4 % theta as the parameter for regularized logistic regression and the
5 % gradient of the cost w.r.t. to the parameters.
6
7 % number of training examples
8 n = length(y);
9
10 % pre-allocate space for gradient
11 %grad = zeros(size(theta));
12 %dont need a vector of 0s
13
14 % Logistic Regression Cost Function
15 J = (1/n)*sum(-y.*(log(sigmoid(X*theta))) -(1-y).*log(1-(sigmoid(X*theta)))) + (lambda/(2*n))
   *sum(theta(2:end).^2);
16
17 theta_0 = theta;
18 theta_0(1) = 0;
19 grad = (1/n)*X'*(sigmoid(X*theta)-y) + (lambda/n)*theta_0;
20 end
```

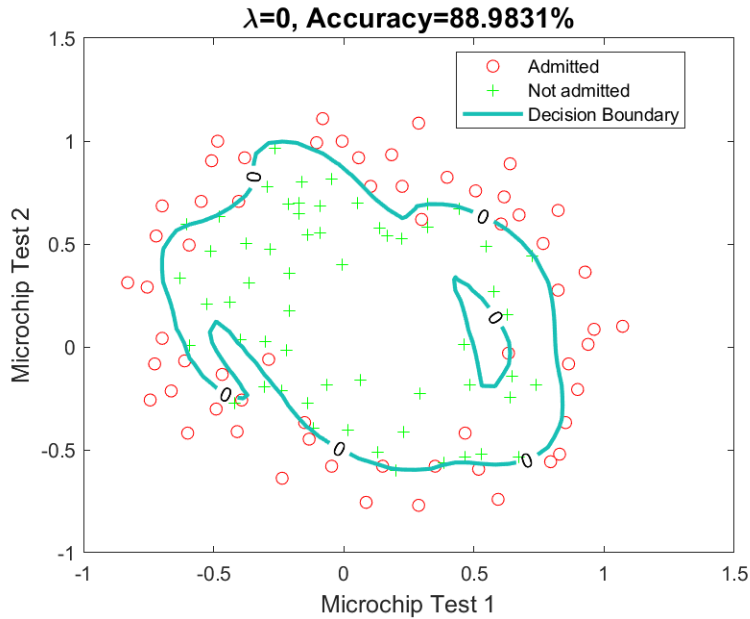
Listing 1: costFunctionLogisticRegression.m

## Problem 2

$\lambda=0$

```
1 %% lambda = 0
2 lambda_0 = 0;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_0, J_0, exit_flag_0] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_0)), initial_theta,
7         options);
8
9 TPTN_0 = 0;
10 TPTNFPPN_0 = 0;
11
12 y_hat_0 = sigmoid(Xdata*theta_0);
13 for index_y_hat_0 = 1:size(y_hat_0)
14     if y_hat_0(index_y_hat_0) >= 0.5
15         y_hat_0(index_y_hat_0) = 1;
16     else
17         y_hat_0(index_y_hat_0) = 0;
18     end
19     if y_hat_0(index_y_hat_0) == y(index_y_hat_0)
20         TPTN_0 = TPTN_0 + 1;
21         TPTNFPPN_0 = TPTNFPPN_0 + 1;
22     else
23         TPTNFPPN_0 = TPTNFPPN_0 + 1;
24     end
25 end
26
27 accuracy_0 = TPTN_0/TPTNFPPN_0;
28
29 title_string = ['{\lambda}= ', num2str(lambda_0), ', Accuracy=', num2str(accuracy_0*100), '%']
30 ];
31 title(title_string, 'fontsize', 14);
32 xlabel('Microchip Test 1', 'fontsize', 12)
33 ylabel('Microchip Test 2', 'fontsize', 12)
34 legend('Admitted', 'Not admitted', 'Decision Boundary', 'location', 'best')
35 print -dpng hwk4_problem2_lambda_0_plot.png
```

Listing 2: problem\_2.m



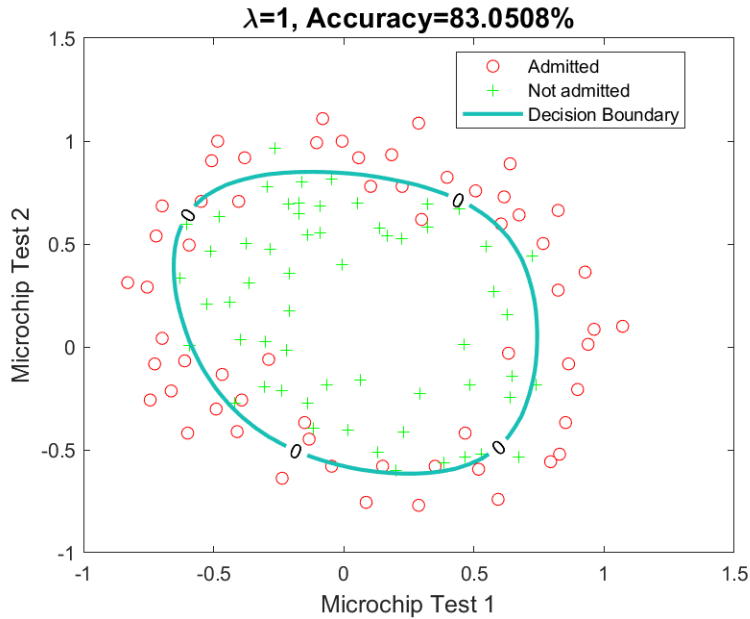
$\lambda=1$

```

1 %% lambda = 1
2 lambda_1 = 1;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_1, J_1, exit_flag_1] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_1)), initial_theta,
7         options);
8
9 plotDecisionBoundary(theta_1, Xdata, y, degree);
10
11
12 TPTN_1 = 0;
13 TPTNFPPN_1 = 0;
14
15 y_hat_1 = sigmoid(Xdata*theta_1);
16 for index_y_hat_1 = 1:size(y_hat_1)
17     if y_hat_1(index_y_hat_1) >= 0.5
18         y_hat_1(index_y_hat_1) = 1;
19     else
20         y_hat_1(index_y_hat_1) = 0;
21     end
22     if y_hat_1(index_y_hat_1) == y(index_y_hat_1)
23         TPTN_1 = TPTN_1 + 1;
24     else
25         TPTNFPPN_1 = TPTNFPPN_1 + 1;
26     end
27 end
28
29 accuracy_1 = TPTN_1/TPTNFPPN_1;
30
31 title_string = ['{\lambda}= ', num2str(lambda_1), ', Accuracy= ', num2str(accuracy_1 * 100), '%
32 '];
33 title(title_string, 'fontsize', 14);
34 xlabel('Microchip Test 1', 'fontsize', 12)
35 ylabel('Microchip Test 2', 'fontsize', 12)
36 legend('Admitted', 'Not admitted', 'Decision Boundary', 'location', 'best')
37
38 print -dpng hwk4_problem2_lambda_1_plot.png

```

Listing 3: problem\_2.m



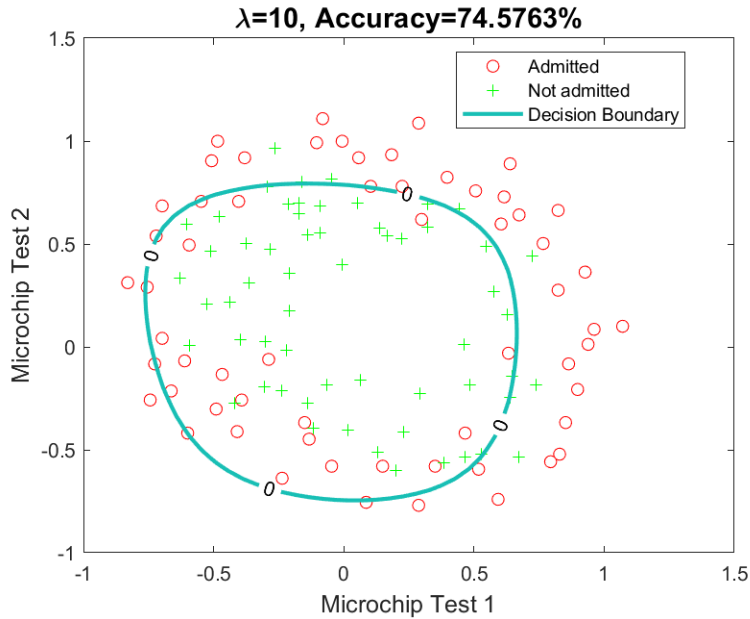
$\lambda=10$

```

1 %% lambda = 10
2 lambda_10 = 10;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_10, J_10, exit_flag_10] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_10)), initial_theta,
7         options);
8
9 plotDecisionBoundary(theta_10, Xdata, y, degree);
10
11 TPTN_10 = 0;
12 TPTNFPFN_10 = 0;
13
14 y_hat_10 = sigmoid(Xdata*theta_10);
15 for index_y_hat_10 = 1:size(y_hat_10)
16     if y_hat_10(index_y_hat_10) >= 0.5
17         y_hat_10(index_y_hat_10) = 1;
18     else
19         y_hat_10(index_y_hat_10) = 0;
20     end
21     if y_hat_10(index_y_hat_10) == y(index_y_hat_10)
22         TPTN_10 = TPTN_10 + 1;
23     else
24         TPTNFPFN_10 = TPTNFPFN_10 + 1;
25     end
26 end
27 accuracy_10 = TPTN_10/TPTNFPFN_10;
28
29 title_string = ['{\lambda}= ', num2str(lambda_10), ', Accuracy=', num2str(accuracy_10 * 100),
30     '%'];
31 title(title_string, 'fontsize', 14);
32 xlabel('Microchip Test 1', 'fontsize', 12)
33 ylabel('Microchip Test 2', 'fontsize', 12)
34 legend('Admitted', 'Not admitted', 'Decision Boundary', 'location', 'best')
35 print -dpng hwk4-problem2_lambda_10-plot.png

```

Listing 4: problem\_2.m



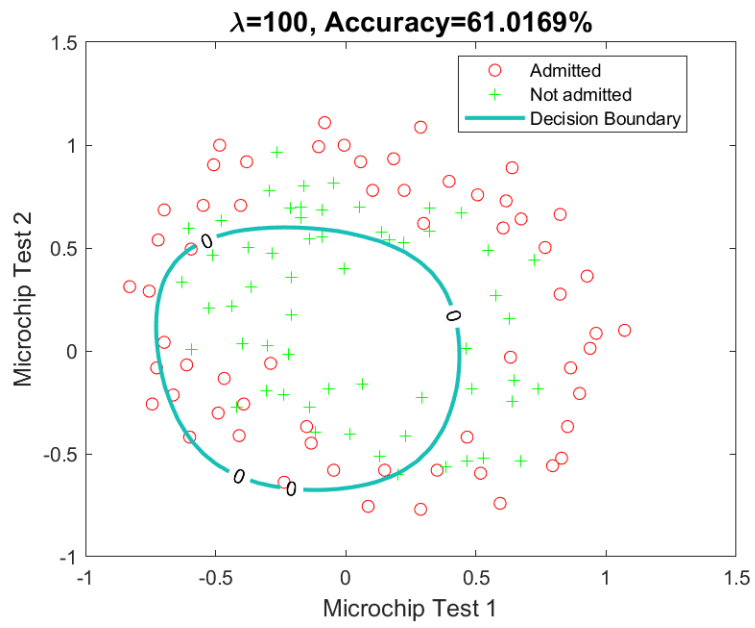
$\lambda=100$

```

1 %% lambda = 100
2 lambda_100 = 100;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_100, J_100, exit_flag_100] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_100)), initial_theta,
7         options);
8
9 plotDecisionBoundary(theta_100, Xdata, y, degree);
10
11
12 TPTN_100 = 0;
13 TPTNFPFN_100 = 0;
14
15 y_hat_100 = sigmoid(Xdata*theta_100);
16 for index_y_hat_100 = 1:size(y_hat_100)
17     if y_hat_100(index_y_hat_100) >= 0.5
18         y_hat_100(index_y_hat_100) = 1;
19     else
20         y_hat_100(index_y_hat_100) = 0;
21     end
22     if y_hat_100(index_y_hat_100) == y(index_y_hat_100)
23         TPTN_100 = TPTN_100 + 1;
24     else
25         TPTNFPFN_100 = TPTNFPFN_100 + 1;
26     end
27 end
28
29 accuracy_100 = TPTN_100/TPTNFPFN_100;
30
31 title_string = ['\lambda=', num2str(lambda_100), ', Accuracy=', num2str(accuracy_100 *
32     100), '%'];
33 title(title_string, 'fontsize', 14);
34 xlabel('Microchip Test 1', 'fontsize', 12)
35 ylabel('Microchip Test 2', 'fontsize', 12)
36 legend('Admitted', 'Not admitted', 'Decision Boundary', 'location', 'best')
37
38 print -dpng hwk4_problem2_lambda_100_plot.png

```

Listing 5: problem\_2.m



## Problem 3

$\lambda=0$

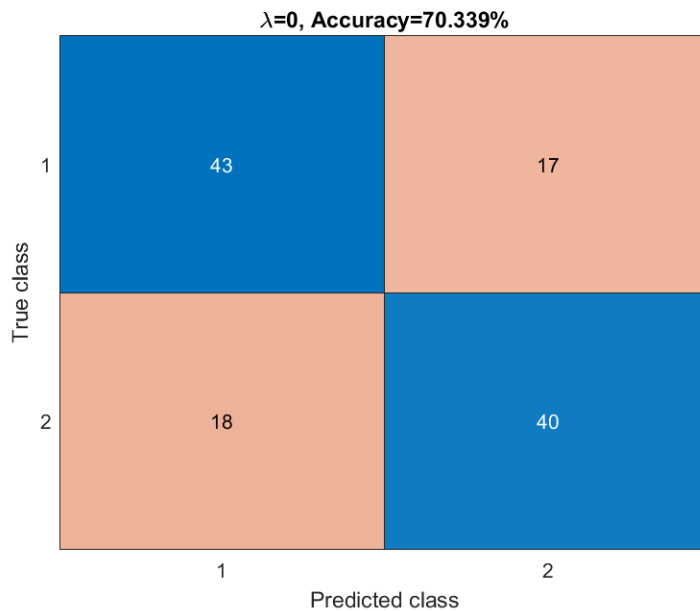
```
1 %% lambda = 0
2 lambda_0 = 0;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_0, J_0, exit_flag_0] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_0)), initial_theta,
7         options);
8
9 TPTN_0 = 0;
10 TPTNFPFN_0 = 0;
11
12 y_hat_0 = sigmoid(Xdata*theta_0);
13 for index_y_hat_0 = 1:size(y_hat_0)
14     if y_hat_0(index_y_hat_0) >= 0.5
15         y_hat_0(index_y_hat_0) = 1;
16     else
17         y_hat_0(index_y_hat_0) = 0;
18     end
19     if y_hat_0(index_y_hat_0) == y(index_y_hat_0)
20         TPTN_0 = TPTN_0 + 1;
21     else
22         TPTNFPFN_0 = TPTNFPFN_0 + 1;
23     end
24 end
25
26 accuracy_0 = TPTN_0/TPTNFPFN_0;
27
28 % the matlab functions you want to use are crossvalind.m and confusionmat.m_
29 % Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
30 % y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label
31     for
32 %each dataset sample
33 % numberOfFolds- the number of folds for k-fold cross validation
34 numberOfFolds=5;
35 rng(2000); %random number generator seed
36 CVindex = crossvalind('Kfold',y, numberOfFolds);
37 method='LogisticRegression';
38 %lambda=1
39 for i = 1:numberOfFolds
40     TestIndex_0 = find(CVindex == i);
41     TrainIndex_0 = find(CVindex ~= i);
42     TrainDataCV_0 = Xdata(TrainIndex_0,:);
43     TrainDataGT_0 = y(TrainIndex_0);
44     TestDataCV_0 = Xdata(TestIndex_0,:);
45     TestDataGT_0 = y(TestIndex_0);
46 %
47 %build the model using TrainDataCV and TrainDataGT
48 %test the built model using TestDataCV
49 %
50 switch method
51     case 'LogisticRegression'
52         % for Logistic Regression, we need to solve for theta
53         % Insert code here to solve for theta...
54         [theta_0_train, J_0_train, exit_flag_0_train] = ...
55             fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_0, TrainDataGT_0,
56                 lambda_0)), initial_theta, options);
57         %plotDecisionBoundary(theta_0_train, TrainDataCV_0, TrainDataGT_0, degree);
58         % Using TestDataCV, compute testing set prediction using
59         % the model created
60         % for Logistic Regression, the model is theta
61         % Insert code here to see how well theta works...
62         TestDataPred_0 = TestDataCV_0 * theta_0_train > 0.5;
```

```

61     case 'KNN'
62         disp('KNN not implemented yet')
63     otherwise
64         error('Unknown classification method')
65 end
66 predictionLabels_0(TestIndex_0,:) = double(TestDataPred_0);
67 end
68 confusionMatrix_0 = confusionmat(y,predictionLabels_0);
69 accuracy_0_cf = sum(diag(confusionMatrix_0))/sum(sum(confusionMatrix_0));
70 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n',method,lambda_0,accuracy_0_cf
    *100));
71 fprintf('Confusion Matrix:\n');
72 [r c] = size(confusionMatrix_0);
73 for i=1:r
74     for j=1:r
75         fprintf('%6d ',confusionMatrix_0(i,j));
76     end
77     fprintf('\n');
78 end
79
80 figure
81 title_string_0 = ['{\lambda}= ', num2str(lambda_0), ', Accuracy=', num2str(accuracy_0_cf *
    100), '%'];
82 confusionchart(confusionMatrix_0, 'Title', title_string_0);
83 print -dpng hwk4_problem3_lambda_0_plot.png

```

Listing 6: problem\_3.m



**$\lambda=1$**

```

1 %% lambda = 1
2 lambda_1 = 1;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_1, J_1, exit_flag_1] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_1)), initial_theta ,
    options);
7 %plotDecisionBoundary(theta_1, Xdata, y, degree);
8
9 TPTN_1 = 0;

```



```

10 TPTNFPFN_1 = 0;
11
12 y_hat_1 = sigmoid(Xdata*theta_1);
13 for index_y_hat_1 = 1:size(y_hat_1)
14     if y_hat_1(index_y_hat_1) >= 0.5
15         y_hat_1(index_y_hat_1) = 1;
16     else
17         y_hat_1(index_y_hat_1) = 0;
18     end
19     if y_hat_1(index_y_hat_1) == y(index_y_hat_1)
20         TPTN_1 = TPTN_1 + 1;
21         TPTNFPFN_1 = TPTNFPFN_1 + 1;
22     else
23         TPTNFPFN_1 = TPTNFPFN_1 + 1;
24     end
25 end
26
27 accuracy_1 = TPTN_1/TPTNFPFN_1;
28
29 % the matlab functions you want to use are crossvalind.m and confusionmat.m_
30 % Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
31 % y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label
32     for
33 %each dataset sample
34 % numberOfFolds- the number of folds for k-fold cross validation
35 numberOfFolds=5;
36 rng(2000); %random number generator seed
37 CVindex = crossvalind('Kfold',y, numberOfFolds);
38 method='LogisticRegression';
39 %lambda=1
40 for i = 1:numberOfFolds
41     TestIndex_1 = find(CVindex == i);
42     TrainIndex_1 = find(CVindex ~= i);
43     TrainDataCV_1 = Xdata(TrainIndex_1,:);
44     TrainDataGT_1 = y(TrainIndex_1);
45     TestDataCV_1 = Xdata(TestIndex_1,:);
46     TestDataGT_1 = y(TestIndex_1);
47 %
48 %build the model using TrainDataCV and TrainDataGT
49 %test the built model using TestDataCV
50 %
51 switch method
52     case 'LogisticRegression'
53         % for Logistic Regression, we need to solve for theta
54         % Insert code here to solve for theta...
55         [theta_1_train, J_1_train, exit_flag_1_train] = ...
56             fminunc(@(t)(costFunctionLogisticRegression(t,TrainDataCV_1, TrainDataGT_1, lambda_1
57             )), initial_theta, options);
58         %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
59         % Using TestDataCV, compute testing set prediction using
60         % the model created
61         % for Logistic Regression, the model is theta
62         % Insert code here to see how well theta works...
63         TestDataPred_1 = TestDataCV_1 * theta_1_train > 0.5;
64     case 'KNN'
65         disp('KNN not implemented yet')
66     otherwise
67         error('Unknown classification method')
68 end
69 predictionLabels_1(TestIndex_1,:) = double(TestDataPred_1);
70
71 confusionMatrix_1 = confusionmat(y, predictionLabels_1);
72 accuracy_1_cf = sum(diag(confusionMatrix_1))/sum(sum(confusionMatrix_1));
73 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n',method,lambda_1,accuracy_1_cf
74 *100));
75 %fprintf('Confusion Matrix:\n');
76 [r c] = size(confusionMatrix_1);
77 for i=1:r

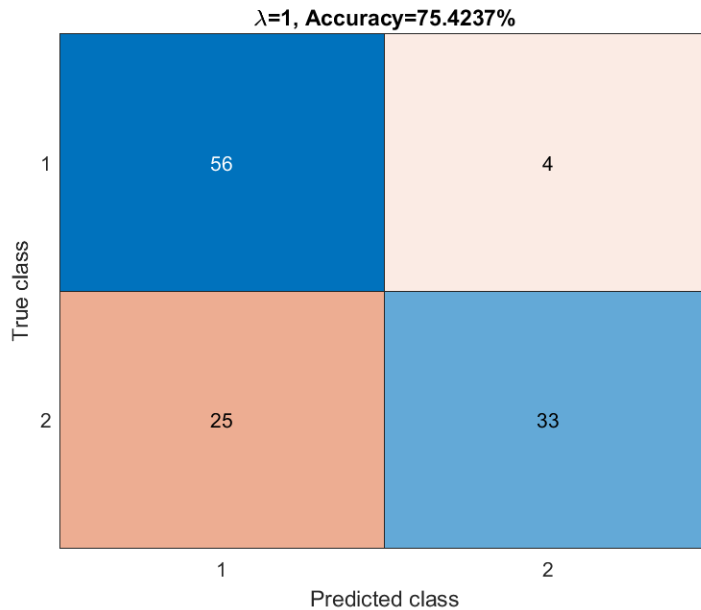
```

```

75 for j=1:r
76 fprintf( '%6d ', confusionMatrix_1(i,j));
77 end
78 fprintf( '\n');
79 end
80
81 figure
82 title_string_1 = [ '\lambda=', num2str(lambda_1), ', Accuracy=', num2str(accuracy_1_cf *
100), '%'];
83 confusionchart( confusionMatrix_1, 'Title', title_string_1);
84 print -dpng hwk4_problem3_lambda_1_plot.png

```

Listing 7: problem\_3.m



$\lambda=10$

```

1 %% lambda = 10
2 lambda_10 = 10;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_10, J_10, exit_flag_10] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_10)), initial_theta,
7     options);
8 %plotDecisionBoundary(theta_10, Xdata, y, degree);
9
10 TPTN_10 = 0;
11 TPTNFPPN_10 = 0;
12
13 y_hat_10 = sigmoid(Xdata*theta_10);
14 for index_y_hat_10 = 1:size(y_hat_10)
15     if y_hat_10(index_y_hat_10) >= 0.5
16         y_hat_10(index_y_hat_10) = 1;
17     else
18         y_hat_10(index_y_hat_10) = 0;
19     end
20     if y_hat_10(index_y_hat_10) == y(index_y_hat_10)
21         TPTN_10 = TPTN_10 + 1;
22     else
23         TPTNFPPN_10 = TPTNFPPN_10 + 1;
24     end
25 end

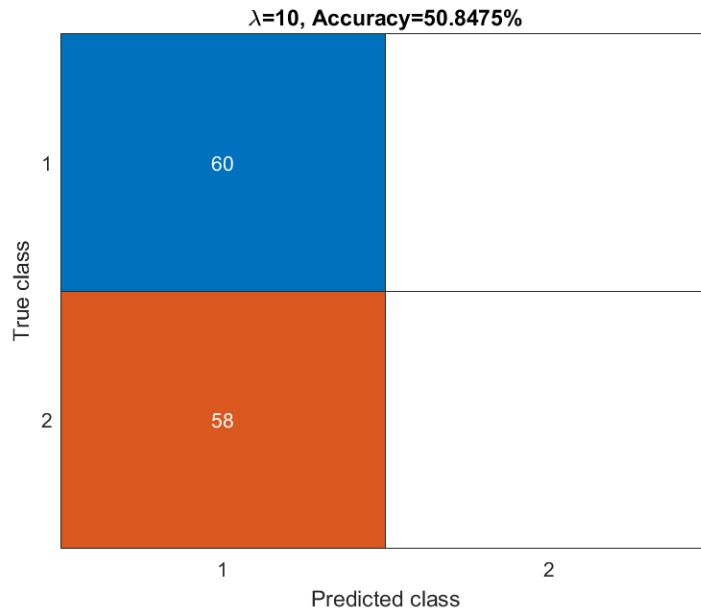
```

```

24     end
25 end
26
27 accuracy_10 = TPTN_10/TPTNFPFN_10;
28
29 % the matlab functions you want to use are crossvalind.m and confusionmat.m_
30 % Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
31 % y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label
    for
32 %each dataset sample
33 % numberOfFolds- the number of folds for k-fold cross validation
34 numberOfFolds=5;
35 rng(2000); %random number generator seed
36 CVindex = crossvalind('Kfold',y, numberOfFolds);
37 method='LogisticRegression';
38 %lambda=1
39 for i = 1:numberOfFolds
40 TestIndex_10 = find(CVindex == i);
41 TrainIndex_10 = find(CVindex ~= i);
42 TrainDataCV_10 = Xdata(TrainIndex_10,:);
43 TrainDataGT_10 = y(TrainIndex_10);
44 TestDataCV_10 = Xdata(TestIndex_10,:);
45 TestDataGT_10 = y(TestIndex_10);
46 %
47 %build the model using TrainDataCV and TrainDataGT
48 %test the built model using TestDataCV
49 %
50 switch method
51     case 'LogisticRegression'
52         % for Logistic Regression, we need to solve for theta
53         % Insert code here to solve for theta...
54         [theta_10_train, J_10_train, exit_flag_10_train] = ...
55             fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_10, TrainDataGT_10,
56                 lambda_10)), initial_theta, options);
57         %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
58         % Using TestDataCV, compute testing set prediction using
59         % the model created
60         % for Logistic Regression, the model is theta
61         % Insert code here to see how well theta works...
62         TestDataPred_10 = TestDataCV_10 * theta_10_train > 0.5;
63     case 'KNN'
64         disp('KNN not implemented yet')
65     otherwise
66         error('Unknown classification method')
67 end
68 predictionLabels_10(TestIndex_10,:) = double(TestDataPred_10);
69 confusionMatrix_10 = confusionmat(y,predictionLabels_10);
70 accuracy_10_cf = sum(diag(confusionMatrix_10))/sum(sum(confusionMatrix_10));
71 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n',method,lambda_10,accuracy_10_cf
72     *100));
73 %fprintf('Confusion Matrix:\n');
74 [r c] = size(confusionMatrix_10);
75 for i=1:r
76     for j=1:c
77         fprintf('%6d ',confusionMatrix_10(i,j));
78     end
79     fprintf('\n');
80 end
81 figure
82 title_string_10 = ['{\lambda}= ', num2str(lambda_10), ', Accuracy=', num2str(accuracy_10_cf *
83     100), '%'];
84 confusionchart(confusionMatrix_10, 'Title', title_string_10);
85 print -dpng hwk4_problem3_lambda_10-plot.png

```

Listing 8: problem.3.m



$\lambda=100$

```

1 %% lambda = 100
2 lambda_100 = 100;
3 % Specifying function with the @(t) allows fminunc to call our costFunction
4 % The t is an input argument, in this case initial_theta
5 [theta_100, J_100, exit_flag_100] = ...
6     fminunc(@(t)(costFunctionLogisticRegression(t, Xdata, y, lambda_100)), initial_theta,
7     options);
8
9 TPTN_100 = 0;
10 TPTNFPPN_100 = 0;
11
12 y_hat_100 = sigmoid(Xdata*theta_100);
13 for index_y_hat_100 = 1:size(y_hat_100)
14     if y_hat_100(index_y_hat_100) >= 0.5
15         y_hat_100(index_y_hat_100) = 1;
16     else
17         y_hat_100(index_y_hat_100) = 0;
18     end
19     if y_hat_100(index_y_hat_100) == y(index_y_hat_100)
20         TPTN_100 = TPTN_100 + 1;
21         TPTNFPPN_100 = TPTNFPPN_100 + 1;
22     else
23         TPTNFPPN_100 = TPTNFPPN_100 + 1;
24     end
25 end
26
27 accuracy_100 = TPTN_100/TPTNFPPN_100;
28
29 % the matlab functions you want to use are crossvalind.m and confusionmat.m
30 % Xdata- A vector of feature, nxD, one set of attributes for each dataset sample
31 % y- A vector of ground truth labels, nx1 (each class has a unique integer value), one label
32     for
33 %each dataset sample
34 % numberOfFolds- the number of folds for k-fold cross validation
35 numberOfFolds=5;
36 rng(2000); %random number generator seed
37 CVindex = crossvalind('Kfold',y, numberOfFolds);
38 method='LogisticRegression';

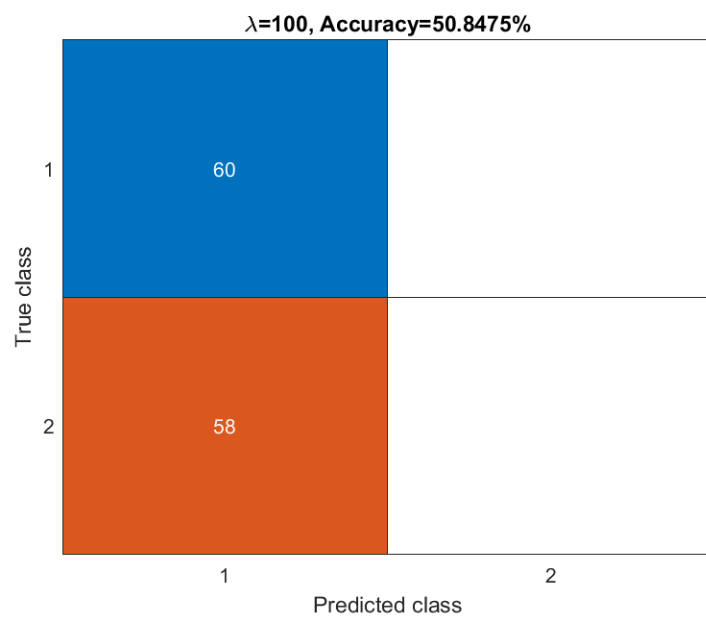
```

```

38 %lambda=1
39 for i = 1:numberOfFolds
40 TestIndex_100 = find(CVindex == i);
41 TrainIndex_100 = find(CVindex ~= i);
42 TrainDataCV_100 = Xdata(TrainIndex_100,:);
43 TrainDataGT_100 = y(TrainIndex_100);
44 TestDataCV_100 = Xdata(TestIndex_100,:);
45 TestDataGT_100 = y(TestIndex_100);
46 %
47 %build the model using TrainDataCV and TrainDataGT
48 %test the built model using TestDataCV
49 %
50 switch method
51     case 'LogisticRegression'
52         % for Logistic Regression, we need to solve for theta
53         % Insert code here to solve for theta...
54         [theta_100_train, J_100_train, exit_flag_100_train] = ...
55             fminunc(@(t)(costFunctionLogisticRegression(t, TrainDataCV_100, TrainDataGT_100,
56                 lambda_100)), initial_theta, options);
57         %plotDecisionBoundary(theta_1_train, TrainDataCV_1, TrainDataGT_1, degree);
58         % Using TestDataCV, compute testing set prediction using
59         % the model created
60         % for Logistic Regression, the model is theta
61         % Insert code here to see how well theta works...
62         TestDataPred_100 = TestDataCV_100 * theta_100_train > 0.5;
63     case 'KNN'
64         disp('KNN not implemented yet')
65     otherwise
66         error('Unknown classification method')
67 end
68 predictionLabels_100(TestIndex_100,:) = double(TestDataPred_100);
69 confusionMatrix_100 = confusionmat(y, predictionLabels_100);
70 accuracy_100_cf = sum(diag(confusionMatrix_100))/sum(sum(confusionMatrix_100));
71 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n', method, lambda_100, accuracy_100_cf
72     *100));
73 %fprintf('Confusion Matrix:\n');
74 [r c] = size(confusionMatrix_100);
75 for i=1:r
76     for j=1:c
77         fprintf('%6d ', confusionMatrix_100(i,j));
78     end
79     fprintf('\n');
80 end
81 figure
82 title_string_100 = ['{\lambda}= ', num2str(lambda_100), ', Accuracy= ', num2str(accuracy_100_cf
83     * 100), '%'];
84 confusionchart(confusionMatrix_100, 'Title', title_string_100);
85 print -dpng hwk4_problem3_lambda_100_plot.png

```

Listing 9: problem\_3.m



## Problem 4

```
1 clear ; close all ; clc
2 % Load Training Data— Andrew Ng Machine Learning MOOC
3 load('ex3data1.mat'); % training data stored in arrays X, y
4 n = size(X, 1);
5 num_labels = length(unique(y)); % 10 labels, from 1 to 10 (note "0" is mapped to label 10)
6 % Randomly select 100 data points to display
7 rng(2000); %random number generator seed
8 rand_indices = randperm(n);
9 sel = X(rand_indices(1:100), :);
10 Xdata = [ones(n, 1) X];
11 % the matlab functions you want to use are crossvalind.m and confusionmat.m_
12 % Xdata— A vector of feature, nxD, one set of attributes for each dataset sample
13 % y— A vector of ground truth labels, nx1 (each class has a unique integer value), one label
    for
14 %each dataset sample
15 % numberOfFolds— the number of folds for k-fold cross validation
16 numberOfFolds=5;
17 rng(2000); %random number generator seed
18 CVindex = crossvalind('Kfold',y, numberOfFolds);
19 method='LogisticRegression'
20 lambda = 0.1;
21 for i = 1:numberOfFolds
22 TestIndex = find(CVindex == i);
23 TrainIndex = find(CVindex ~= i);
24 TrainDataCV = Xdata(TrainIndex,:);
25 TrainDataGT = y(TrainIndex);
26 TestDataCV = Xdata(TestIndex,:);
27 TestDataGT = y(TestIndex);
28 %
29 %build the model using TrainDataCV and TrainDataGT
30 %test the built model using TestDataCV
31 %
32 switch method
33 case 'LogisticRegression'
34 % for Logistic Regression, we need to solve for theta
35 % Initialize fitting parameters
36 all_theta = zeros(num_labels, size(Xdata, 2));
37 for c=1:num_labels
38 % Set Initial theta
39 initial_theta = zeros(size(Xdata, 2), 1);
40 % Set options for fminunc
41 options = optimset('GradObj', 'on', 'MaxIter', 50);
42 % Run fmincg to obtain the optimal theta
43 % This function will return theta and the cost
44 [theta] = ...
45 fmincg (@(t)(costFunctionLogisticRegression(t, TrainDataCV, (TrainDataGT == c), lambda)),
    ...
46 initial_theta, options);
47 all_theta(c,:) = theta;
48 end
49 % Using TestDataCV, compute testing set prediction using
50 % the model created
51 % for Logistic Regression, the model is theta
52 % Insert code here to see how well theta works...
53 all_pred = sigmoid(TestDataCV*all_theta');
54 [maxVal,maxIndex] = max(all_pred,[],2);
55 TestDataPred=maxIndex;
56 case 'KNN'
57 disp('KNN not implemented yet')
58 otherwise
59 error('Unknown classification method')
60 end
61 predictionLabels(TestIndex,:) =double(TestDataPred);
62 end
63 confusionMatrix = confusionmat(y, predictionLabels);
```

```

64 accuracy = sum(diag(confusionMatrix))/sum(sum(confusionMatrix));
65 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n',method, lambda, accuracy*100));
66 fprintf('Confusion Matrix:\n');
67 [r c] = size(confusionMatrix);
68 for i=1:r
69 for j=1:c
70 fprintf('%6d ',confusionMatrix(i,j));
71 end
72 fprintf('\n');
73 end
74
75 figure
76 title_string = ['method:', method, ' {\lambda}= ', num2str(lambda), ', Accuracy=', num2str(
    accuracy * 100), '%'];
77 confusionchart(confusionMatrix, 'Title', title_string);
78 print -dpng hwk4_problem4_plot.png

```

Listing 10: problem\_4.m

**method:LogisticRegression  $\lambda=0.1$ , Accuracy=90.2%**

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
	1	2	3	4	5	6	7	8	9	10

True class

Predicted class



## Problem 5

```
1 clear ; close all ; clc
2 % Load Training Data— Andrew Ng Machine Learning MOOC
3 load('ex3data1.mat'); % training data stored in arrays X, y
4 n = size(X, 1);
5 num_labels = length(unique(y)); % 10 labels, from 1 to 10 (note "0" is mapped to label 10)
6 % Randomly select 100 data points to display
7 rng(2000); %random number generator seed
8 rand_indices = randperm(n);
9 sel = X(rand_indices(1:100), :);
10 Xdata = [ones(n, 1) X];
11 % the matlab functions you want to use are crossvalind.m and confusionmat.m_
12 % Xdata— A vector of feature, nxD, one set of attributes for each dataset sample
13 % y— A vector of ground truth labels, nx1 (each class has a unique integer value), one label
    for
14 %each dataset sample
15 % numberOfFolds— the number of folds for k-fold cross validation
16 numberOfFolds=5;
17 rng(2000); %random number generator seed
18 CVindex = crossvalind('Kfold',y, numberOfFolds);
19 %method='LogisticRegression'
20 method='KNN';
21 lambda = 0.1;
22 for i = 1:numberOfFolds
23 TestIndex = find(CVindex == i);
24 TrainIndex = find(CVindex ~= i);
25 TrainDataCV = Xdata(TrainIndex,:);
26 TrainDataGT = y(TrainIndex);
27 TestDataCV = Xdata(TestIndex,:);
28 TestDataGT = y(TestIndex);
29 %
30 %build the model using TrainDataCV and TrainDataGT
31 %test the built model using TestDataCV
32 %
33 switch method
34 case 'LogisticRegression'
35 % for Logistic Regression, we need to solve for theta
36 % Initialize fitting parameters
37 all_theta = zeros(num_labels, size(Xdata, 2));
38 for c=1:num_labels
39 % Set Initial theta
40 initial_theta = zeros(size(Xdata, 2), 1);
41 % Set options for fminunc
42 options = optimset('GradObj', 'on', 'MaxIter', 50);
43 % Run fmincg to obtain the optimal theta
44 % This function will return theta and the cost
45 [theta] = ...
46 fmincg (@(t)(costFunctionLogisticRegression(t, TrainDataCV, (TrainDataGT == c), lambda)),
    ...
47 initial_theta, options);
48 all_theta(c,:) = theta;
49 end
50 % Using TestDataCV, compute testing set prediction using
51 % the model created
52 % for Logistic Regression, the model is theta
53 % Insert code here to see how well theta works...
54 all_pred = sigmoid(TestDataCV*all_theta');
55 [maxVal,maxIndex] = max(all_pred,[],2);
56 TestDataPred=maxIndex;
57 case 'KNN'
58 %disp('KNN not implemented yet')
59 k_value = 3;
60 Idx = knnsearch(TrainDataCV,TestDataCV, 'k', k_value);
61 Idx_gt = [TrainDataGT(Idx(:,1)) TrainDataGT(Idx(:,2)) TrainDataGT(Idx(:,3))];
62 Idx_gt_T = Idx_gt';
63 TestDataPred=mode(Idx_gt_T)';
```

```

64 otherwise
65 error('Unknown classificatio method')
66 end
67 predictionLabels(TestIndex,:) =double(TestDataPred);
68 end
69 confusionMatrix = confusionmat(y,predictionLabels);
70 accuracy = sum(diag(confusionMatrix))/sum(sum(confusionMatrix));
71 fprintf(sprintf('%s: Lambda = %d, Accuracy = %6.2f%%\n',method, lambda, accuracy*100));
72 fprintf('Confusion Matrix:\n');
73 [r c] = size(confusionMatrix);
74 for i=1:r
75 for j=1:c
76 fprintf('%6d ',confusionMatrix(i,j));
77 end
78 fprintf('\n');
79 end
80
81 figure
82 title_string = ['method:', method, ' {\lambda}= ', num2str(lambda), ', Accuracy=', num2str(
    accuracy * 100), '%'];
83 confusionchart(confusionMatrix, 'Title', title_string);
84 print -dpng hwk4_problem5_plot.png

```

Listing 11: problem\_5.m

**method:KNN  $\lambda=0.1$ , Accuracy=94.22%**

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

Predicted class