

```

1  %% Machine Learning Online Class - Exercise 2: Logistic Regression
2  %
3  % Instructions
4  % -----
5  %
6  % This file contains code that helps you get started on the logistic
7  % regression exercise. You will need to complete the following
8  % • functions
9  % in this exercise:
10 %
11 %     sigmoid.m
12 %     costFunction.m
13 %     predict.m
14 %     costFunctionReg.m
15 %
16 % For this exercise, you will not need to change any code in this file,
17 % or any other files other than those mentioned above.
18 %
19 %% Initialization
20 clear ; close all; clc
21
22 %% Load Data
23 % The first two columns contains the exam scores and the third column
24 % contains the label.
25
26 data = load('ex2data1.txt');
27 X = data(:, [1, 2]); y = data(:, 3);
28
29 %% ===== Part 1: Plotting =====
30 % We start the exercise by first plotting the data to understand the
31 % the problem we are working with.
32
33 fprintf(['Plotting data with + indicating (y = 1) examples and o ' ...
34         'indicating (y = 0) examples.\n']);
35
36 plotData(X, y);
37
38 % Put some labels
39 hold on;
40 % Labels and Legend
41 xlabel('Exam 1 score')
42 ylabel('Exam 2 score')
43
44 % Specified in plot order
45 legend('Admitted', 'Not admitted')
46 hold off;
47
48 fprintf('\nProgram paused. Press enter to continue.\n');
49 pause;
50
51

```

```

51
52 %% ===== Part 2: Compute Cost and Gradient =====
53 % In this part of the exercise, you will implement the cost and
54 % gradient
55 % for logistic regression. You need to complete the code in
56 % costFunction.m
57 % Setup the data matrix appropriately, and add ones for the intercept
58 % term
59
60 [m, n] = size(X);
61
62 % Add intercept term to x and X_test
63 X = [ones(m, 1) X];
64
65 % Initialize fitting parameters
66 initial_theta = zeros(n + 1, 1);
67
68 % Compute and display initial cost and gradient
69 [cost, grad] = costFunction(initial_theta, X, y);
70
71 fprintf('Cost at initial theta (zeros): %f\n', cost);
72 fprintf('Expected cost (approx): 0.693\n');
73 fprintf('Gradient at initial theta (zeros): \n');
74 fprintf(' %f \n', grad);
75 fprintf('Expected gradients (approx):\n -0.1000\n -12.0092\n
76 -11.2628\n');
77
78 % Compute and display cost and gradient with non-zero theta
79 test_theta = [-24; 0.2; 0.2];
80 [cost, grad] = costFunction(test_theta, X, y);
81
82 fprintf('\nCost at test theta: %f\n', cost);
83 fprintf('Expected cost (approx): 0.218\n');
84 fprintf('Gradient at test theta: \n');
85 fprintf(' %f \n', grad);
86 fprintf('Expected gradients (approx):\n 0.043\n 2.566\n 2.647\n');
87
88 fprintf('\nProgram paused. Press enter to continue.\n');
89 pause;
90
91 %% ===== Part 3: Optimizing using fminunc =====
92 % In this exercise, you will use a built-in function (fminunc) to find
93 % the
94 % optimal parameters theta.
95
96 % Set options for fminunc
97 options = optimset('GradObj', 'on', 'MaxIter', 400);
98
99 % Run fminunc to obtain the optimal theta
100 % This function will return theta and the cost
101 [theta, cost] = ...

```

```

99     fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
100
101 % Print theta to screen
102 fprintf('Cost at theta found by fminunc: %f\n', cost);
103 fprintf('Expected cost (approx): 0.203\n');
104 fprintf('theta: \n');
105 fprintf(' %f \n', theta);
106 fprintf('Expected theta (approx):\n');
107 fprintf(' -25.161\n 0.206\n 0.201\n');
108
109 % Plot Boundary
110 plotDecisionBoundary(theta, X, y);
111
112 % Put some labels
113 hold on;
114 % Labels and Legend
115 xlabel('Exam 1 score')
116 ylabel('Exam 2 score')
117
118 % Specified in plot order
119 legend('Admitted', 'Not admitted')
120 hold off;
121
122 fprintf('\nProgram paused. Press enter to continue.\n');
123 pause;
124
125 %% ===== Part 4: Predict and Accuracies =====
126 % After learning the parameters, you'll like to use it to predict the
    • outcomes
127 % on unseen data. In this part, you will use the logistic regression
    • model
128 % to predict the probability that a student with score 45 on exam 1 and
129 % score 85 on exam 2 will be admitted.
130 %
131 % Furthermore, you will compute the training and test set accuracies of
132 % our model.
133 %
134 % Your task is to complete the code in predict.m
135
136 % Predict probability for a student with score 45 on exam 1
137 % and score 85 on exam 2
138
139 prob = sigmoid([1 45 85] * theta);
140 fprintf(['For a student with scores 45 and 85, we predict an admission
    • ' ...
141         'probability of %f\n'], prob);
142 fprintf('Expected value: 0.775 +/- 0.002\n\n');
143

```

```
1  function g = sigmoid(z)
2  %SIGMOID Compute sigmoid function
3  %   g = SIGMOID(z) computes the sigmoid of z.
4
5  % You need to return the following variables correctly
6  g = zeros(size(z));
7
8  % ===== YOUR CODE HERE =====
9  % Instructions: Compute the sigmoid of each value of z (z can be a matrix,
10 %                vector or scalar).
11
12  g = 1./(1 + e.^(-z));
13
14
15
16  % =====
17
18  end
19
```

```

1  function [J, grad] = costFunction(theta, X, y)
2  %COSTFUNCTION Compute cost and gradient for logistic regression
3  %   J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the
4  %   parameter for logistic regression and the gradient of the cost
5  %   w.r.t. to the parameters.
6
7  % Initialize some useful values
8  m = length(y); % number of training examples
9
10 % You need to return the following variables correctly
11 J = 0;
12 grad = zeros(size(theta));
13
14 % ===== YOUR CODE HERE =====
15 % Instructions: Compute the cost of a particular choice of theta.
16 %               You should set J to the cost.
17 %               Compute the partial derivatives and set grad to the partial
18 %               derivatives of the cost w.r.t. each parameter in theta
19 %
20 % Note: grad should have the same dimensions as theta
21 %
22
23 h = sigmoid(X*theta); % función de hipótesis
24
25 J = (-1/m) * sum( y .* log(h) + (1 - y) .* log(1 - h) ); % m=numero de
    • ejemplos
26
27 for i = 1:m
28     grad = grad + ( h(i) - y(i) ) * X(i, :)';
29 end
30
31 % gradient = nx1 column vector
32 grad = (1/m) * grad;
33
34
35
36
37
38
39
40
41 % =====
42
43 end
44

```

```

1  function [JconRegul, GradconRegul] = costFunctionReg(theta, X, y, lambda)
2  %COSTFUNCTIONREG Compute cost and gradient for logistic regression with
  • regularization
3  % J = COSTFUNCTIONREG(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  % Initialize some useful values
8  m = length(y); % number of training examples
9
10 % You need to return the following variables correctly
11 J = 0;
12 grad = zeros(size(theta));
13
14 % ===== YOUR CODE HERE =====
15 % Instructions: Compute the cost of a particular choice of theta.
16 %               You should set J to the cost.
17 %               Compute the partial derivatives and set grad to the partial
18 %               derivatives of the cost w.r.t. each parameter in theta
19
20
21 h = sigmoid(X*theta); %
22
23 J = (-1/m) * sum( y .* log(h) + (1 - y) .* log(1 - h) );
24
25 Regul = lambda/(2*m) * sum( theta(2:end).^2 );
26
27 JconRegul = J + Regul;
28
29 % compute the gradient
30 for i = 1:m
31     grad = grad + ( h(i) - y(i) ) * X(i, :)';
32 end
33
34 GradRegul = lambda/m * [0; theta(2:end)];
35
36 GradconRegul = (1/m) * grad + GradRegul;
37
38
39
40
41 % =====
42
43 end
44

```

```

1  function plotDecisionBoundary(theta, X, y)
2  %PLOTDECISIONBOUNDARY Plots the data points X and y into a new figure with
3  %the decision boundary defined by theta
4  % PLOTDECISIONBOUNDARY(theta, X,y) plots the data points with + for the
5  % positive examples and o for the negative examples. X is assumed to be
6  % a either
7  % 1) Mx3 matrix, where the first column is an all-ones column for the
8  % intercept.
9  % 2) MxN, N>3 matrix, where the first column is all-ones
10
11 % Plot Data
12 plotData(X(:,2:3), y);
13 hold on
14
15 if size(X, 2) <= 3
16     % Only need 2 points to define a line, so choose two endpoints
17     plot_x = [min(X(:,2))-2, max(X(:,2))+2];
18
19     % Calculate the decision boundary line
20     plot_y = (-1./theta(3)).*(theta(2).*plot_x + theta(1));
21
22     % Plot, and adjust axes for better viewing
23     plot(plot_x, plot_y)
24
25     % Legend, specific for the exercise
26     legend('Admitted', 'Not admitted', 'Decision Boundary')
27     axis([30, 100, 30, 100])
28 else
29     % Here is the grid range
30     u = linspace(-1, 1.5, 50);
31     v = linspace(-1, 1.5, 50);
32
33     z = zeros(length(u), length(v));
34     % Evaluate z = theta*x over the grid
35     for i = 1:length(u)
36         for j = 1:length(v)
37             z(i,j) = mapFeature(u(i), v(j))*theta;
38         end
39     end
40     z = z'; % important to transpose z before calling contour
41
42     % Plot z = 0
43     % Notice you need to specify the range [0, 0]
44     contour(u, v, z, [0, 0], 'LineWidth', 2)
45 end
46 hold off
47
48 end
49

```

```
1  function out = mapFeature(X1, X2)
2  % MAPFEATURE Feature mapping function to polynomial features
3  %
4  %   MAPFEATURE(X1, X2) maps the two input features
5  %   to quadratic features used in the regularization exercise.
6  %
7  %   Returns a new feature array with more features, comprising of
8  %   X1, X2, X1.^2, X2.^2, X1*X2, X1*X2.^2, etc..
9  %
10 %   Inputs X1, X2 must be the same size
11 %
12
13 degree = 6;
14 out = ones(size(X1(:,1)));
15 for i = 1:degree
16     for j = 0:i
17         out(:, end+1) = (X1.^(i-j)).*(X2.^j);
18     end
19 end
20
21 end
```



```

1  %% Machine Learning Online Class - Exercise 2: Logistic Regression
2  %
3  % Instructions
4  % -----
5  %
6  % This file contains code that helps you get started on the second part
7  % of the exercise which covers regularization with logistic regression.
8  %
9  % You will need to complete the following functions in this exercise:
10 %
11 %     sigmoid.m
12 %     costFunction.m
13 %     predict.m
14 %     costFunctionReg.m
15 %
16 % For this exercise, you will not need to change any code in this file,
17 % or any other files other than those mentioned above.
18 %
19
20 %% Initialization
21 clear ; close all; clc
22
23 %% Load Data
24 % The first two columns contains the X values and the third column
25 % contains the label (y).
26
27 data = load('ex2data2.txt');
28 X = data(:, [1, 2]); y = data(:, 3);
29
30 plotData(X, y);
31
32 % Put some labels
33 hold on;
34
35 % Labels and Legend
36 xlabel('Microchip Test 1')
37 ylabel('Microchip Test 2')
38
39 % Specified in plot order
40 legend('y = 1', 'y = 0')
41 hold off;
42
43
44 %% ===== Part 1: Regularized Logistic Regression =====
45 % In this part, you are given a dataset with data points that are not
46 % linearly separable. However, you would still like to use logistic
47 % regression to classify the data points.
48 %
49 % To do so, you introduce more features to use -- in particular, you
50 % • add
51 % polynomial features to our data matrix (similar to polynomial
52 % regression)

```

```

51 % Logistic Regression
52 %
53
54 % Add Polynomial Features
55
56 % Note that mapFeature also adds a column of ones for us, so the
  • intercept
57 % term is handled
58 X = mapFeature(X(:,1), X(:,2));
59
60 % Initialize fitting parameters
61 initial_theta = zeros(size(X, 2), 1);
62
63 % Set regularization parameter lambda to 1
64 lambda = 1;
65
66 % Compute and display initial cost and gradient for regularized logistic
67 % regression
68 [cost, grad] = costFunctionReg(initial_theta, X, y, lambda);
69
70 fprintf('Cost at initial theta (zeros): %f\n', cost);
71 fprintf('Expected cost (approx): 0.693\n');
72 fprintf('Gradient at initial theta (zeros) - first five values
  • only:\n');
73 fprintf(' %f \n', grad(1:5));
74 fprintf('Expected gradients (approx) - first five values only:\n');
75 fprintf(' 0.0085\n 0.0188\n 0.0001\n 0.0503\n 0.0115\n');
76
77 fprintf('\nProgram paused. Press enter to continue.\n');
78 pause;
79
80 % Compute and display cost and gradient with non-zero theta
81 test_theta = ones(size(X,2),1);
82 [cost, grad] = costFunctionReg(test_theta, X, y, lambda);
83
84 fprintf('\nCost at test theta: %f\n', cost);
85 fprintf('Expected cost (approx): 2.13\n');
86 fprintf('Gradient at test theta - first five values only:\n');
87 fprintf(' %f \n', grad(1:5));
88 fprintf('Expected gradients (approx) - first five values only:\n');
89 fprintf(' 0.3460\n 0.0851\n 0.1185\n 0.1506\n 0.0159\n');
90
91 fprintf('\nProgram paused. Press enter to continue.\n');
92 pause;
93
94 %% ===== Part 2: Regularization and Accuracies =====
95 % Optional Exercise:
96 % In this part, you will get to try different values of lambda and
97 % see how regularization affects the decision countart
98 %
99 % Try the following values of lambda (0, 1, 10, 100).
100 %

```

```

101 % How does the decision boundary change when you vary lambda? How does
102 % the training set accuracy vary?
103 %
104
105 % Initialize fitting parameters
106 initial_theta = zeros(size(X, 2), 1);
107
108 % Set regularization parameter lambda to 1 (you should vary this)
109 lambda = 1;
110
111 % Set Options
112 options = optimset('GradObj', 'on', 'MaxIter', 400);
113
114 % Optimize
115 [theta, J, exit_flag] = ...
116     fminunc(@(t)(costFunctionReg(t, X, y, lambda)), initial_theta,
117     •     options);
118
119 % Plot Boundary
120 plotDecisionBoundary(theta, X, y);
121 hold on;
122 title(sprintf('lambda = %g', lambda))
123
124 % Labels and Legend
125 xlabel('Microchip Test 1')
126 ylabel('Microchip Test 2')
127
128 legend('y = 1', 'y = 0', 'Decision boundary')
129 hold off;
130
131 % Compute accuracy on our training set
132 p = predict(theta, X);

```