Programming Exercise 2: Logistic Regression

Carson Slater
Deep Learning - Spring 2026

February 5, 2026

1 Introduction

The primary objective of this exercise is to implement logistic regression and regularized logistic regression. Logistic regression is a classification algorithm used when the target variable is categorical. This report explores two cases: one linearly separable classroom dataset and a non-linearly separable microchip dataset that requires feature mapping and regularization.

2 Logistic Regression

In this part of the exercise, we build a logistic regression model to predict whether a student gets admitted into a university based on their results of two exams.

2.1 Sigmoid Function

The logistic regression hypothesis is defined as:

$$h_\theta(x) = g(\theta^T x)$$

where g is the sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

The implementation in `sigmoid.m` is as follows:

```
function g = sigmoid(z)
%SIGMOID Compute sigmoid function
  g = 1 ./ (1 + exp(-z));
end
```

Listing 1: sigmoid.m

2.2 Cost Function and Gradient

The cost function for logistic regression is:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

The gradient of the cost is a vector where the j^{th} element is defined as:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

The implementation in `costFunction.m` is as follows:

```
function [J, grad] = costFunction(theta, X, y)
  m = length(y);
  h = sigmoid(X * theta);
  J = (1 / m) * sum(-y .* log(h) - (1 - y) .* log(1 - h));
  grad = (1 / m) * (X' * (h - y));
end
```

Listing 2: costFunction.m

2.3 Learning Parameters using `fminunc`

Instead of gradient descent, we use MATLAB's `fminunc` to find the optimal parameters θ that minimize the cost function. This built-in function is more efficient and does not require choosing a learning rate.

2.4 Results and Visualization

After optimization, the model achieves a training accuracy of **89%**. Figure 1 shows the training data, and Figure 2 displays the learned decision boundary.

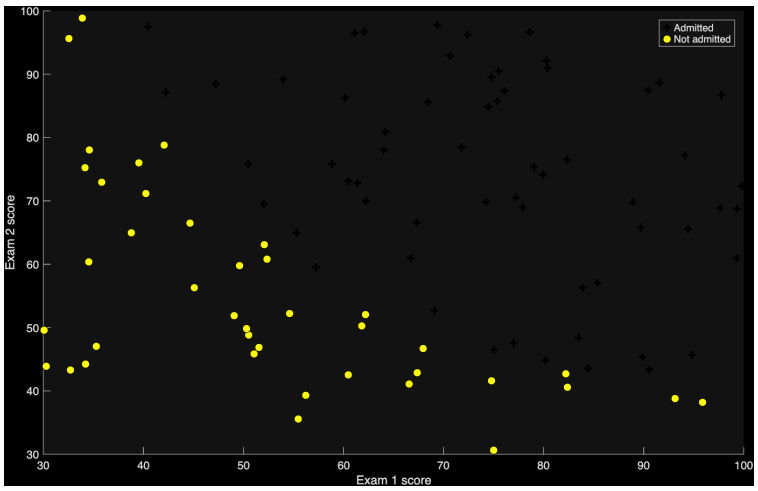

Figure 1: Scatter plot of university admission data.

3 Regularized Logistic Regression

In the second part, we implement regularized logistic regression to predict whether microchips from a fabrication plant pass quality assurance (QA) tests.

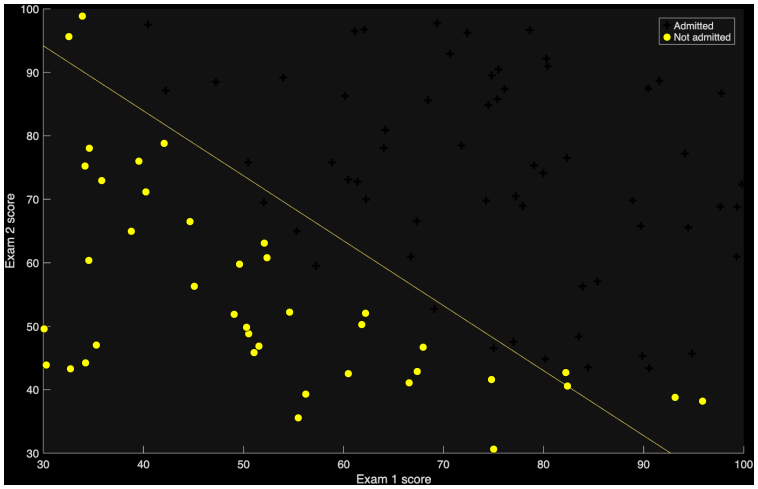

Figure 2: Decision boundary for logistic regression on the university admission dataset.

3.1 Feature Mapping

The data is not linearly separable, so we map the features into all polynomial terms of x_1 and x_2 up to the sixth power to create more features.

3.2 Regularized Cost Function and Gradient

The cost function for regularized logistic regression is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=2}^{n}\theta_j^2$$

Note that we do not regularize θ_0. The partial derivative for the regularized gradient is:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)} \quad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1$$

The implementation in `costFunctionReg.m` is as follows:

```
function [J, grad] = costFunctionReg(theta, X, y, lambda)
  m = length(y);
  h = sigmoid(X * theta);
  theta_reg = [0; theta(2:end)];
  J = (1 / m) * sum(-y .* log(h) - (1 - y) .* log(1 - h)) + (lambda / (2 *
    m)) * sum(theta_reg .^ 2);
  grad = (1 / m) * (X' * (h - y)) + (lambda / m) * theta_reg;
end
```

Listing 3: costFunctionReg.m

3.3 Results and Regularization Effects

With $\lambda = 1$, the model found a decision boundary (Figure 3) that balances fitting the data and preventing overfitting, achieving a training accuracy of approximately **83.1%**.

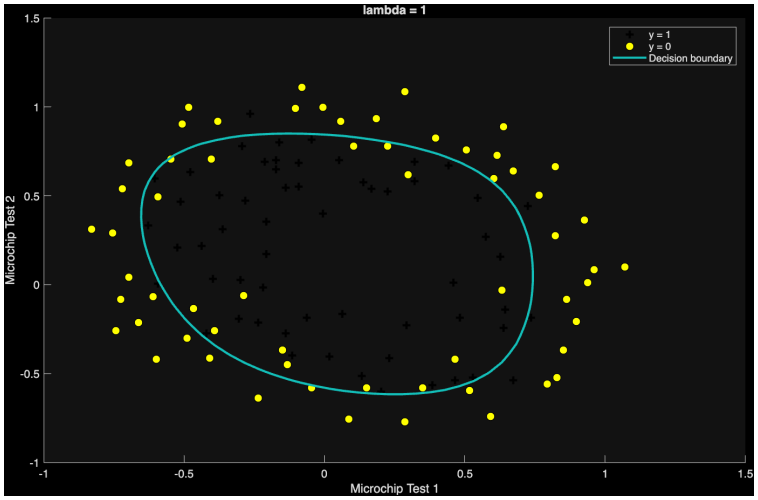

Figure 3: Decision boundary with $\lambda = 1$ for the regularized logistic regression case.

4 Execution Scripts

The following scripts provide the full context for the experiments conducted.

```
%% Initialization
clear ; close all; clc
data = load('ex2data1.txt');
X = data(:, [1, 2]); y = data(:, 3);
[m, n] = size(X);
X = [ones(m, 1) X];
initial_theta = zeros(n + 1, 1);
[cost, grad] = costFunction(initial_theta, X, y);
options = optimset('GradObj', 'on', 'MaxIter', 400);
[theta, cost] = fminunc(@(t)(costFunction(t, X, y)), initial_theta,
   options);
plotDecisionBoundary(theta, X, y);
```

Listing 4: ex2.m (Simplified)

```
%% Initialization
clear ; close all; clc
data = load('ex2data2.txt');
X = data(:, [1, 2]); y = data(:, 3);
X = mapFeature(X(:,1), X(:,2));
initial_theta = zeros(size(X, 2), 1);
lambda = 1;
[cost, grad] = costFunctionReg(initial_theta, X, y, lambda);
options = optimset('GradObj', 'on', 'MaxIter', 400);
[theta, J, exit_flag] = fminunc(@(t)(costFunctionReg(t, X, y, lambda)),
   initial_theta, options);
plotDecisionBoundary(theta, X, y);
```

Listing 5: ex2_reg.m (Simplified)

5 Console Output

The following is the console output from running the main scripts, verifying the correctness of the cost function and optimized parameters.

```
Plotting data with + indicating (y = 1) examples and o indicating (y = 0)
   examples.

Program paused. Press enter to continue.
Cost at initial theta (zeros): 0.693147
Expected cost (approx): 0.693
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842
Expected gradients (approx):
 -0.1000
 -12.0092
 -11.2628

Cost at test theta: 0.218330
Expected cost (approx): 0.218
Gradient at test theta:
 0.042903
 2.566234
 2.646797
Expected gradients (approx):
 0.043
 2.566
 2.647

Program paused. Press enter to continue.

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

Cost at theta found by fminunc: 0.203498
Expected cost (approx): 0.203
theta:
 -25.161343
 0.206232
 0.201472
Expected theta (approx):
 -25.161
 0.206
 0.201

Program paused. Press enter to continue.
For a student with scores 45 and 85, we predict an admission probability
   of 0.776291
Expected value: 0.775 +/- 0.002

```

```
Train Accuracy: 89.000000
Expected accuracy (approx): 89.0
```

Listing 6: ex2.m Console Output

```
Cost at initial theta (zeros): 0.693147
Expected cost (approx): 0.693
Gradient at initial theta (zeros) - first five values only:
 0.008475
 0.018788
 0.000078
 0.050345
 0.011501
Expected gradients (approx) - first five values only:
 0.0085
 0.0188
 0.0001
 0.0503
 0.0115

Program paused. Press enter to continue.

Cost at test theta (with lambda = 10): 3.164509
Expected cost (approx): 3.16
Gradient at test theta - first five values only:
 0.346045
 0.161352
 0.194796
 0.226863
 0.092186
Expected gradients (approx) - first five values only:
 0.3460
 0.1614
 0.1948
 0.2269
 0.0922

Program paused. Press enter to continue.

Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

Train Accuracy: 83.050847
Expected accuracy (with lambda = 1): 83.1 (approx)
```

Listing 7: ex2_reg.m Console Output

6 Conclusion

The implementation of logistic regression and regularization was successful. The models were able to capture the underlying patterns in both linearly separable and non-linearly separable datasets.

The optimized parameters yields high accuracy, and the regularized results demonstrate how the model complexity can be controlled to improve generalization.