LOGISTIC REGRESSION

Machine Learning implemented in MATLAB - Report 2

Author: Alberto Ibarrondo Date: 25/04/2017

1 INTRODUCTION

The objective of this study is implement logistic regression and apply it to two different datasets. Later on, regularization is included.

1.1 Files included in this study

- *MachLearnInMATLAB_2_LogisticRegression.m*- MATLAB script for the whole implementation of the study. The script sets up the dataset for the problems and makes calls to the rest of the functions. The first part of the study implements logistic regression with two variables, while the second part covers logist regression with multiple variables and regularization.
- *data1.txt* – Training set for the logistic regression
- *data2.txt* – Training set for the log. regression with regularization
- *computeCost.m* - Function to compute the cost of logistic regression
- *costFunctionReg.m* - Regularized Logistic Regression Cost
- *mapFeature.m* - Function to generate polynomial features
- *sigmoid.m* - Implementation of Sigmoid Function
- plotLogReg.m - Plot the data with 2 parameters and labels 0/1
- plotDecisionBoundary.m - Plot the decision boundary as well

2 Logistic Regression

In this part of the study we build a logistic regression model to predict whether a student gets admitted into a university.

Suppose an administrator of a university department who wants to determine each applicant's chance of admission based on their results on two exams. He has historical data from previous applicants to use as a training set for logistic regression. For each training example, there is the applicant's scores on two exams and the admissions decision.

We will build a classification model that estimates an applicant's probability of admission based the scores from those two exams.

2.1 Importing and Visualizing the Data

Before starting on any task, it is often useful to understand the data by visualizing it. For a dataset with only two features a Scatter plot is suitable.

We import the data into the variables X and y:

- **X**: two columns with the exam scores
- **y**: the label (1 is accepted, 0 is not accepted)

Next, we create a scatter plot of the data:

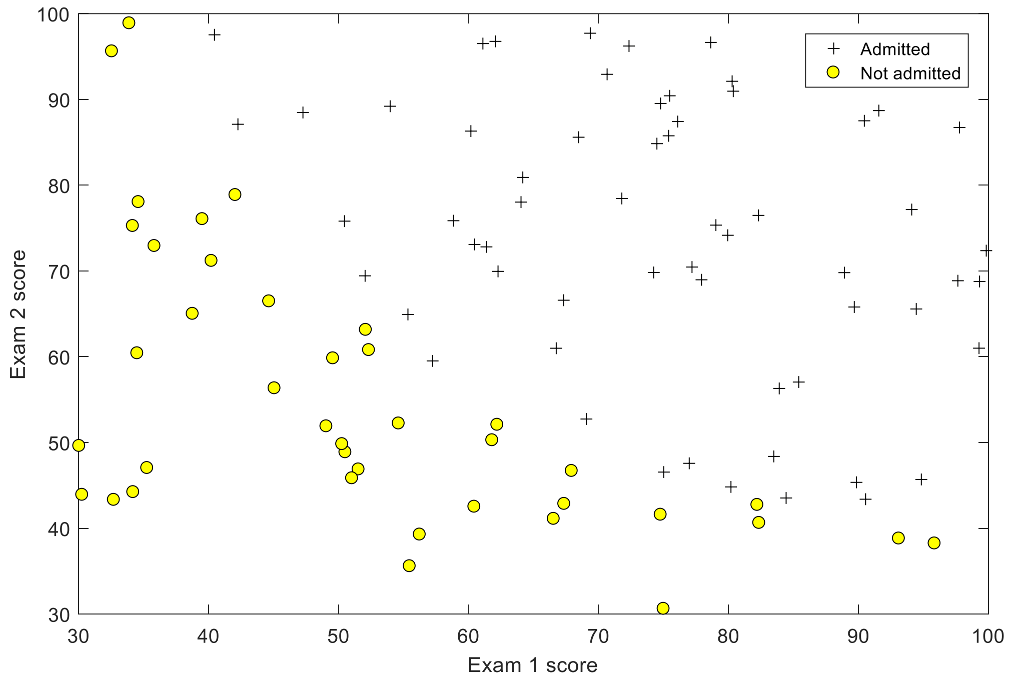

Figure 1 - Scatter plot of training data

2.2 Cost Function and Gradient

2.2.1 Sigmoid Function

Before we start with the actual cost function, let's remember that the logistic regression hypothesis is defined as:

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

where function g is the sigmoid function (although several other activation functions can be used). The sigmoid function is defined as:

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

We will be using **sigmoid.m** to host the function

2.2.2 Implementation

Recall that the cost function in logistic regression is:

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

The gradient, on the other hand, is:

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

Let' s implement this in **costFunction.m**:

```
function [J, grad] = costFunction(theta, X, y)
%COSTFUNCTION Compute cost and gradient for logistic regression
%   J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the
%   parameter for logistic regression and the gradient of the cost
%   w.r.t. to the parameters.

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

    % Cost
    J=1/m*(-y'*log(sigmoid(X*theta))-(1-y)'*log(1-sigmoid(X*theta)));

    % Gradient
    grad=1/m*(sigmoid(X*theta)-y)'*X;
end
```

2.3 Optimization using fminunc

In the previous study, we found optimal parameters of linear regression model with gradient descent. This time, instead of taking gradient descent steps, we will use built-in function called fminunc: optimization solver that finds the minimum of an unconstrained function. When applied to logistic regression, our aim is to optimize the cost function $J(\theta)$ with respect to θ:

```
opts = optimset('GradObj', 'on', 'MaxIter', 400);     %Set options

%  Convex optim to obtain the optimal theta
[theta, cost] = ...
    fminunc(@(t)(costFunction(t, X, y)), init_theta, opts);
```

Our result is the following:

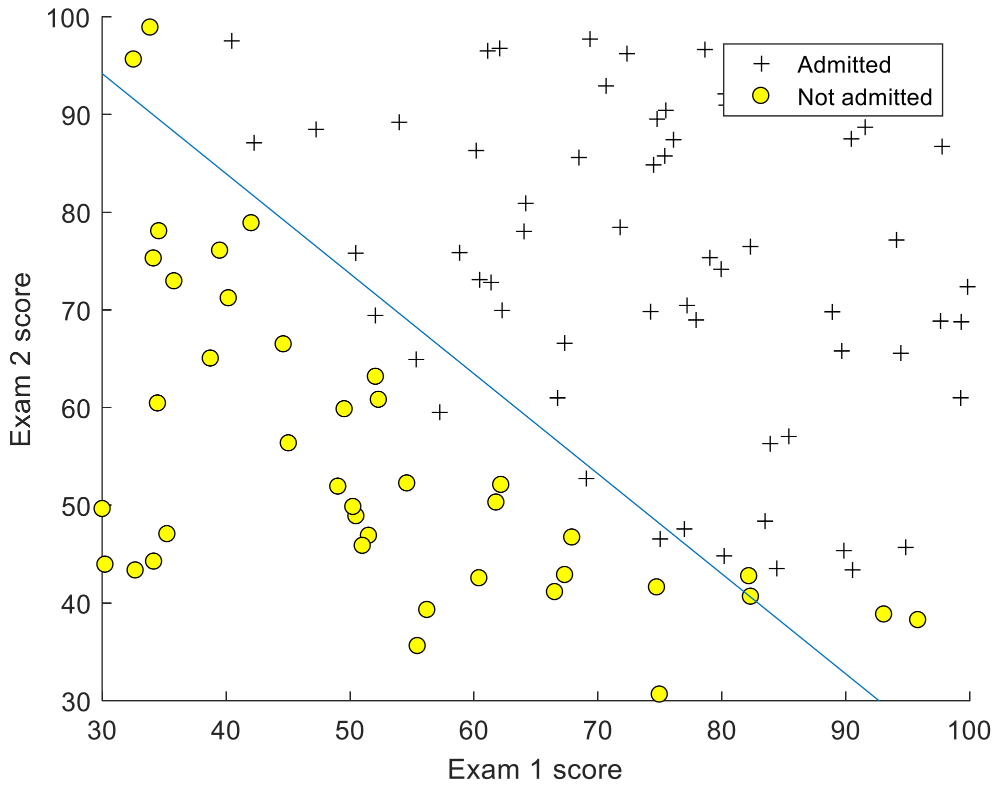

Figure 2 - Linear frontier for logistic regression

We can perfectly distinguish the frontier that labels 1 or 0 depending on both exam scores. Nevertheless, the model is linear, showing some underfitting.

2.4 Prediction and accuracies

After learning the parameters, we can predict whether a particular student will be admitted. Another way to evaluate the quality of the parameters we have found is to see how well the learned model predicts on the whole training set, this is, calculate the accuracy in the predictions.

We use the logistic regression model to predict the probability that a student with score 45 on exam 1 and score 85 on exam 2 will be admitted, obtaining a 77,4%. We also obtain a **training accuracy of 89%.**

3 REGULARIZED LOGISTIC REGRESSION

The second part of the study focuses on implementing regularized logistic regression to predict whether microchips from a fabrication plant passes quality assurance (QA) based on some tests. Suppose you are the product manager of the factory and you have the test results for some microchips on two different tests. From these two tests, we'd like to determine whether the microchips should be accepted or no.

3.1 Visualizing the data

Same as before, we start by plotting the data to have some insights on it:

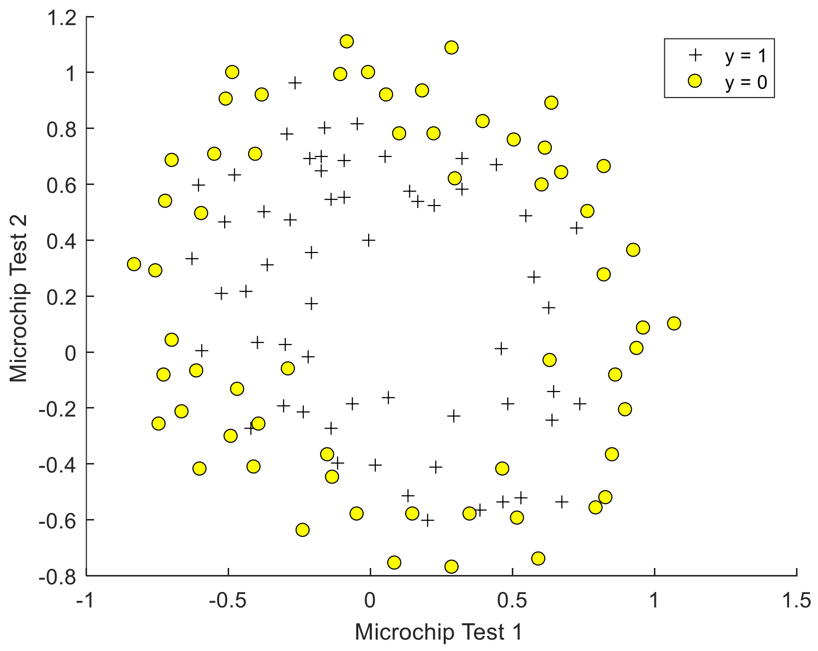

Figure 3 - Second dataset. Non linearly separable data

3.2 Feature mapping

One way to fit the data is to expand the features. In mapFeature.m we map the features into polynomial terms of x_1 and x_2 up to the sixth power:

```
function out = mapFeature(X1, X2, degree)
%   MAPFEATURE(X1, X2) maps the two input features
%   to quadratic features used in the regularization exercise.
%   Returns a new feature array with more features, comprising of
%   X1, X2, X1.^2, X2.^2, X1*X2, X1*X2.^2, etc..
%
out = ones(size(X1(:,1)));
       for i = 1:degree
           for j = 0:i
               out(:, end+1) = (X1.^(i-j)).*(X2.^j);
           end
       end
end
```

As a result of this mapping, our vector of two features (the scores on two QA tests) has been transformed into a 28-dimensional vector. A logistic regression classifier trained on this higher-dimension feature vector will have a more complex decision boundary and will appear nonlinear when drawn in our 2-dimensional plot. While the feature mapping allows us to build a more expressive classifier, it also more susceptible to overfitting.

3.3 Cost function and gradient

We implemented the cost function and gradient for regularized logistic regression in costFunctionReg.m to return the cost and gradient using regularization. The cost function will be:

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

The gradient, on the other hand, will be:

$$\frac{\partial J(\theta)}{\partial\theta} = \left(\frac{1}{m}\sum_{i=1}^{m}[\,h_\theta(x^{(i)}) - y^{(i)}]x_j^{(i)}\right) + \frac{\lambda}{m}\sum_{j=1}^{n}\theta_j$$

Similar to the previous section, you will use fminunc to learn the optimal parameters θ. If you have completed the cost and gradient for regularized logistic regression (costFunctionReg.m) correctly, you should be able to step through the next part of ex2 reg.m to learn the parameters θ using fminunc.

3.4 Model Selection based on Accuracy

To help us visualize the model learned by this classifier, we use plotDecisionBoundary.m which plots the (non-linear) decision boundary that separates the positive and negative examples.

We train the model with different learning rates and compare the prediction results. Based on the results and on the figures, we select **lambda=0.01** as the best learning rate, since it produces a high enough accuracy but it doesn't overfit the data the same way lower learning rates do.

```
% Compute accuracy on our training set
prediction = (sigmoid(X*theta)>=0.5);
accuracy = mean(double(prediction == y)) * 100;
```

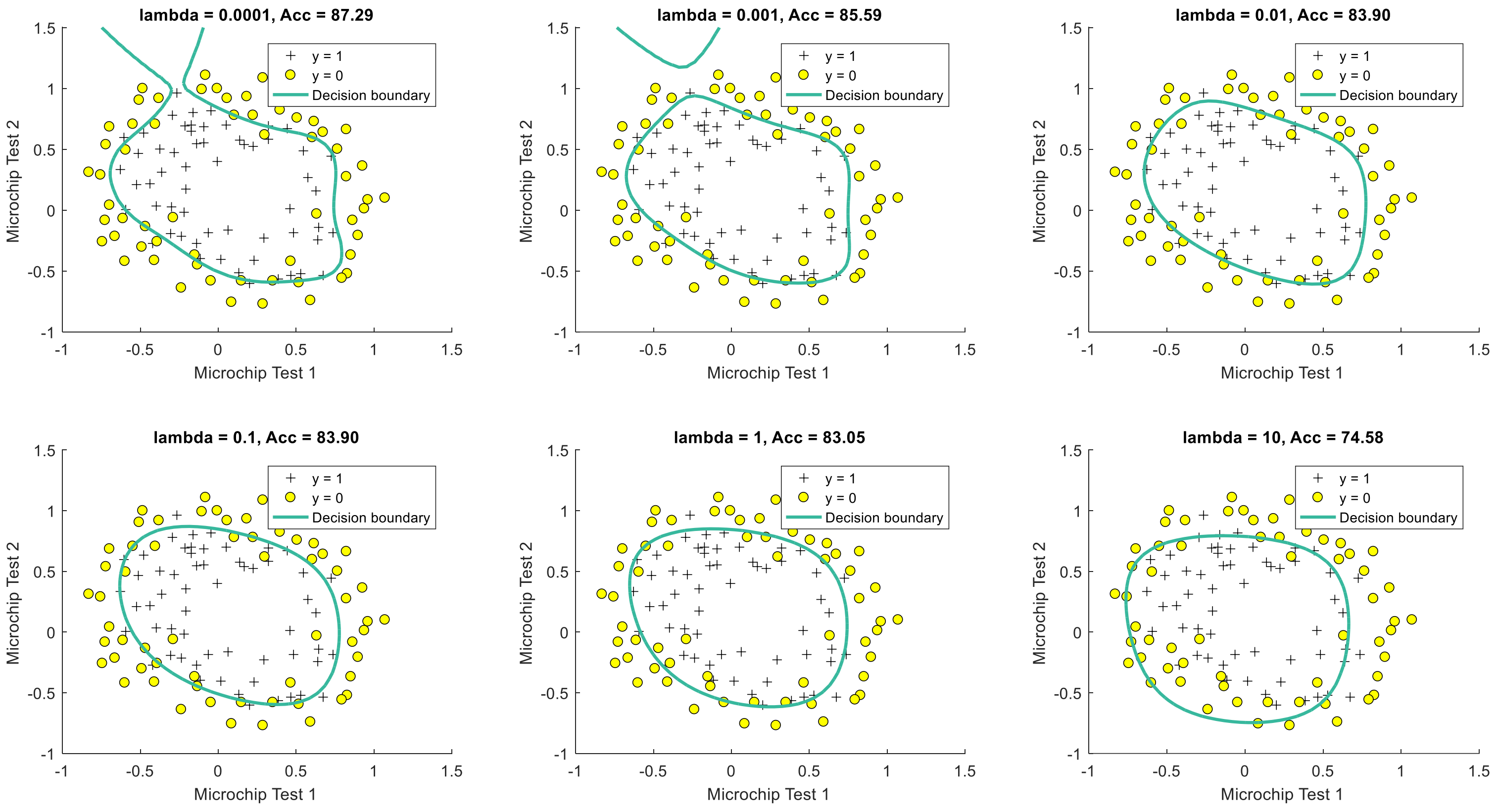

Figure 4 - Model Selection for Regularization based on Accuracy

4 Conclusion

This study successfully implements logistic regression in MATLAB, while introducing regularization and feature expansion as extra tools to be used in almost any classification model in the broad field of Machine Learning.