# COMS W4771 - Machine Learning 1st Exersice

Georgios Koloventzos - gk2409

October 1, 2015

## 1 Problem 1

The exercise at the beginning asks for the best degree using some empirical risk. So for having some empirical results we will run the polyreg function in a variety of degrees. In general we use the degrees from 1 to 50. We choose this as the final plot it resembles the one in the example Function polyreg does exactly this when 3 arguments are given. We choose this as the final plot it resembles the one in the example.

The code for this problem is the `polyeg.m` and `hm1_1.m`. The result for the empirical is that the lowest error comes with a 11th degree polynomial.

If we print all the empirical risks that we computed, we will see that the lowest error is with the polynomial degrees 8,9,10,11 and the ones with degrees 12,13,14,15,16 are really close. The errors are 0.0113 and 0.0114 respectively. So these degrees are reasonable.

In order to do 2-fold cross validation we split our data into 2 sets. I use the randperm function to create an array with the numbers from 1 to 500. This will split the data in 2 random sets. Then we will run the polyreg function 2 times one with the 1st set as training and the 2nd as testing and vice versa. Then we will average the 2 errors between these 2 runs to give us the best degree of polynomial. From our empirical measurement I believe that it should be between 8th and 15th degree (the plot is in figure 1).

The result of this procedure varies. This is because the training and testing data differ in each run. But still at my runs, it is never lower than 8 or more than 15. This coincides with the empirical risk we already found ( the plot for best dimension is figure 2).

## 2 Problem 2

The Matlab files for this exercise are `multireg.m` and `hm1_2.m`

I use the least square loss function to penalize in order to use the gradient function that we have from the slides(slide 9, class3x.pdf).

For the 2-fold cross validation I use the same techinque as the Problem 1, dividing the dataset into 2 sets. I use one set as training and the other as test and vice versa. The plot that is created is far from what we have seen in class. In this exercise I split the dataset in half and not in random. The $\lambda$ that minimizes the testing error is 950 (figure 3). From the plot we can conclude that the our function that does the penalize does not have underfitting (it is too small).

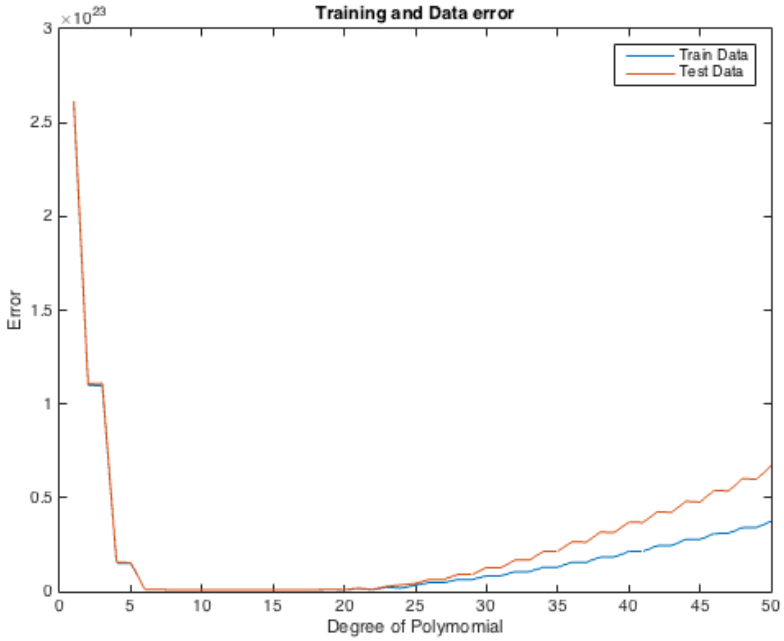


Figure 1: Training and Testing loss

## 3 Problem 3

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

$$g(x) = \frac{1}{1 + e^{-x}} = y$$
$$y + y * e^{-x} = 1 \Rightarrow e^{-x} = \frac{1 - y}{y} \Rightarrow$$
$$e^x = \frac{y}{1 - y} \Rightarrow \ln(e^x) = \ln(\frac{y}{1 - y}) \Rightarrow$$
$$x = \ln(\frac{y}{1 - y}) \Rightarrow g(y) = \ln(\frac{y}{1 - y})$$

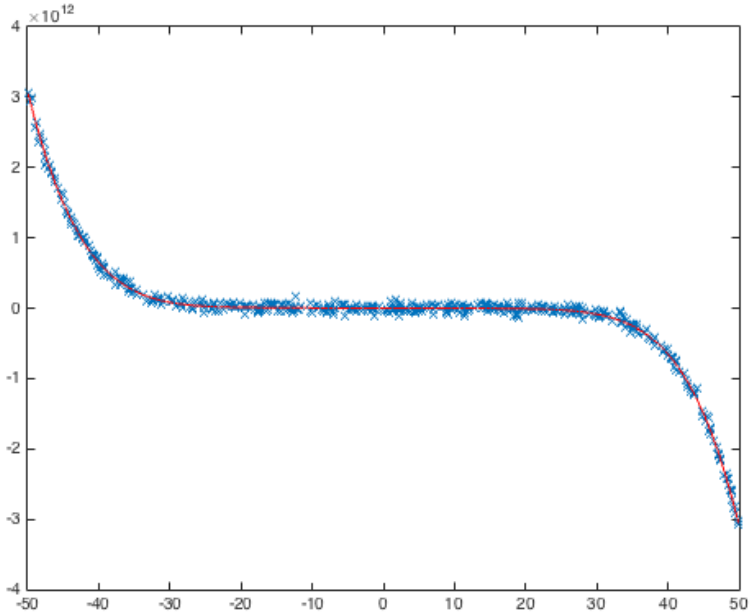


Figure 2: Data with the best polynomial from 2 fold.

## 4 Problem 4

Here is the derivations I did in order to an easier function inside the gradient:

$$\sum_{i=1}^{n}\Big(\frac{1-y_i}{1-f(x_i;\theta)}-\frac{y_i}{f(x_i;\theta)}\Big)f'(x_i;\theta) \Leftrightarrow \sum_{i=1}^{n}\Big(\frac{1-y_i}{1-g(\theta^T x)}-\frac{y_i}{g(\theta^T x)}\Big)g'(\theta^T x) \Leftrightarrow$$

$$\sum_{i=1}^{n}\Big(\frac{1-y_i}{1-g(\theta^T x)}-\frac{y_i}{g(\theta^T x)}\Big)(\theta^T x)'(g'(\theta^T x)) \Leftrightarrow \sum_{i=1}^{n}\Big(\frac{1-y_i}{1-g(\theta^T x)}-\frac{y_i}{g(\theta^T x)}\Big) * x * g(\theta^T x)(1-g(\theta^T x)) \Leftrightarrow$$

$$\sum_{i=1}^{n}\Big((1-y_i) * g(\theta^T x) - (y_i * (1-g(\theta^T x)) * x\Big) \Leftrightarrow \sum_{i=1}^{n}\Big((g(\theta^T x)-y_i)x\Big)$$

The Matlab files for this exercise are `logreg.m` and `hm1_4.m` You can find the Desicion boundary figure 4. The binary classification error 5. The empirical risk figure **??**.

For tolerance 0.0001 and step 0.1 the algorithm needs 217797 iterations.

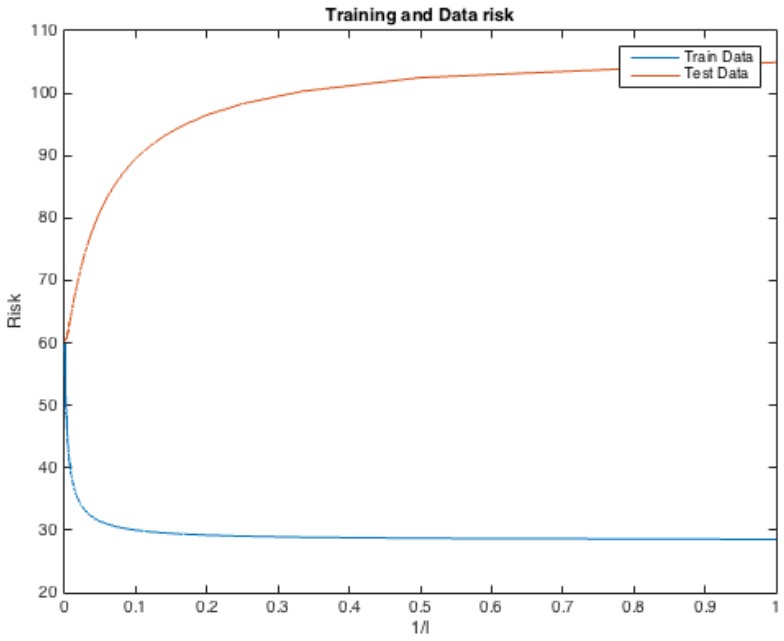


Figure 3: Training and Testing Risk

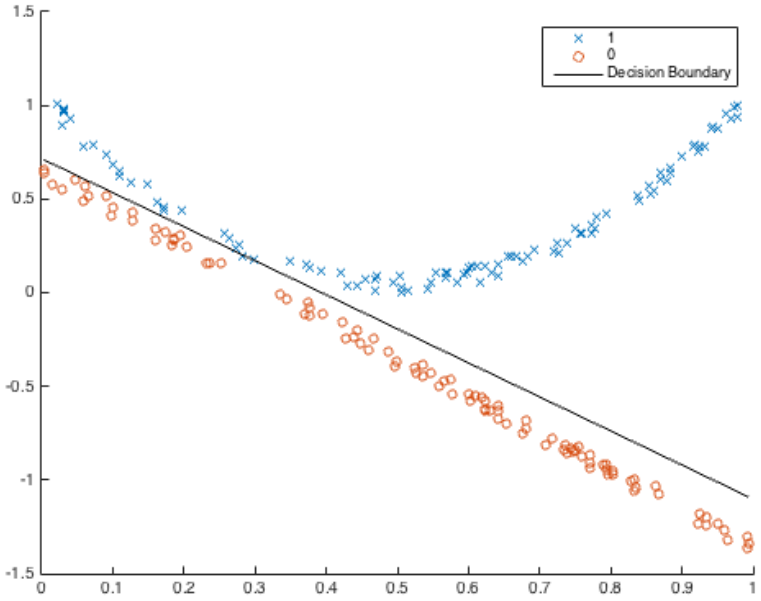


Figure 4: Decision Boundary

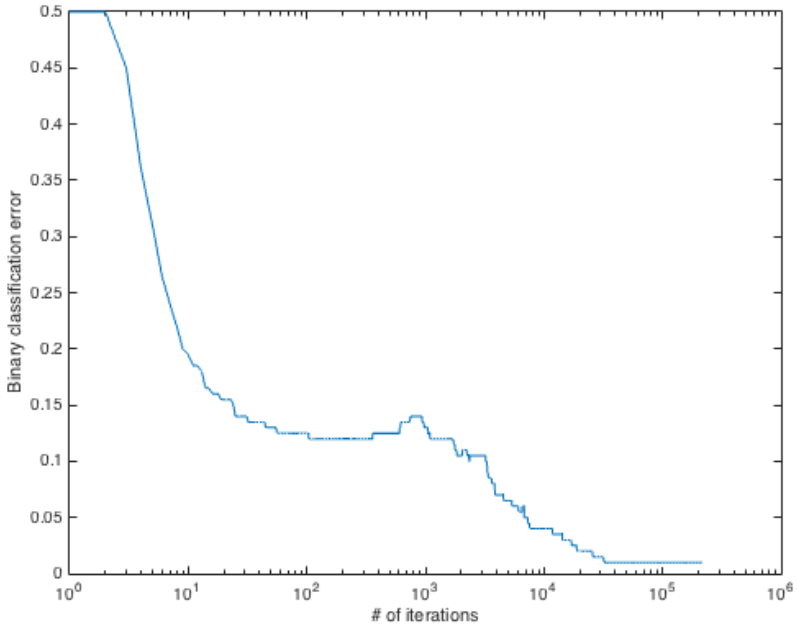


Figure 5: Binary Classification Error

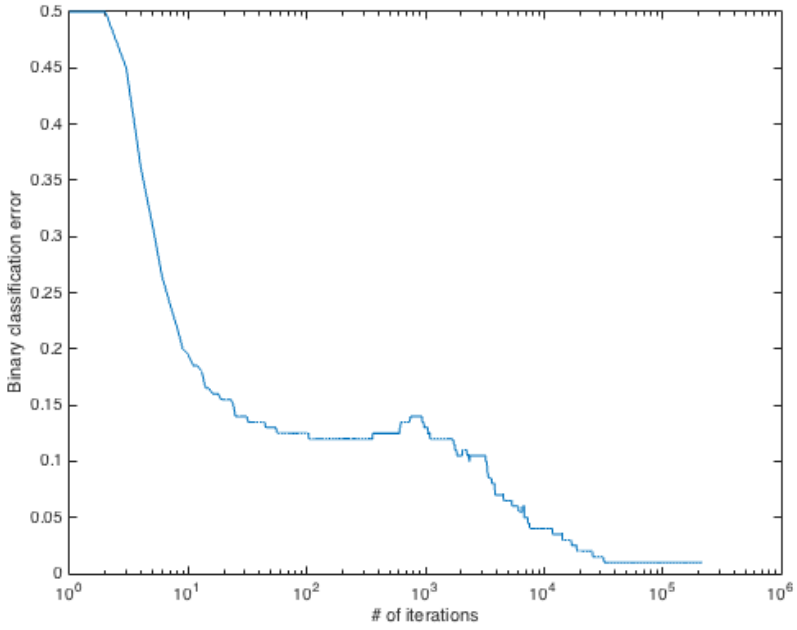


Figure 6: Empirical Risk