NAME: DUONG DUY CHIEN
Student ID: 605415156

Regularization with Logistic Regression

I. Theory:

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = g(\theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n)$$

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

Cost function:

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

Gradient descent algorithm:

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

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

with m= number of samples

j= 1....n : number of feartures

$NOTE : the$ fomula of gradient is the same with linear regression, but the hypothesis is different

The way to implement by vectorization is quite similar to the homework 1 so that I won't mention it again in here.

II. Results:

The new dataset points are bigger than the first part of exercise (as shown in Figure 1). So that to fit with the data better we create more polynomial terms of x_1 and x_2 up to sixth power

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

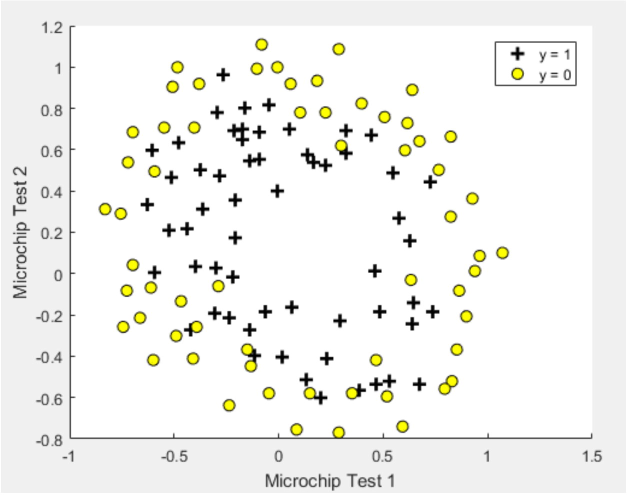

Figure 1 Plot of training data

After create more features, I apply regularization to prevents over fitting. With $\lambda = 1$, I got the good decision boundary (as shown in Figure 2) .

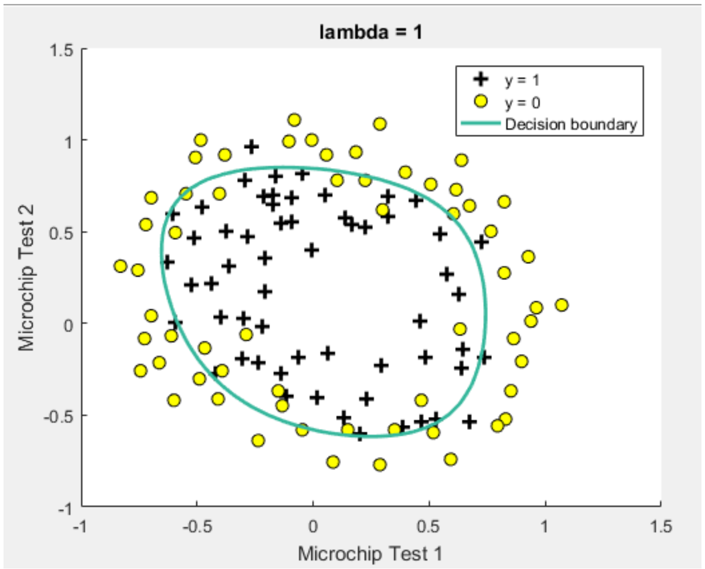

Figure 2 Training data with decision boundary (lambda=1)

In order to understand how regularization prevents overfitting. I also drew two decision boundaries with $\lambda = 0$ and $\lambda = 100$ which were shown in Figure 3 and 4 respectively.

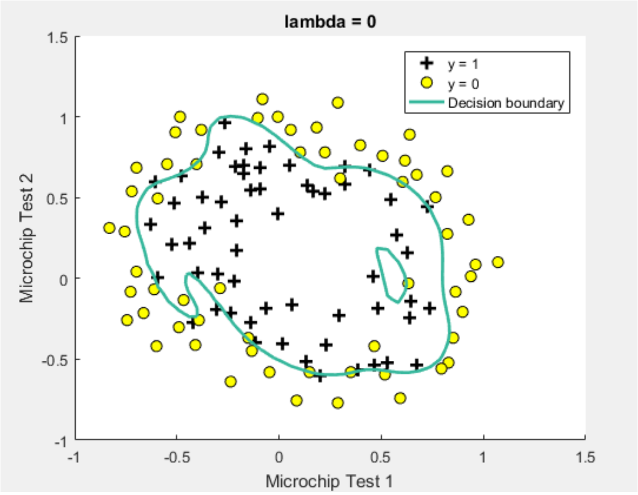

Figure 3 Training data with decision boundary (lambda=0)

In Figure 3, because $\lambda = 0$ means that I did not use regularization which help to reduce the effect of features to the cost function. So that I got the overfitting, because I have so many features and don't have enough training data.

In Figure 4, because $\lambda = 100$ means that I reduced the effect of features to the cost function too much. So that almost features will become zero. I got under fitting.

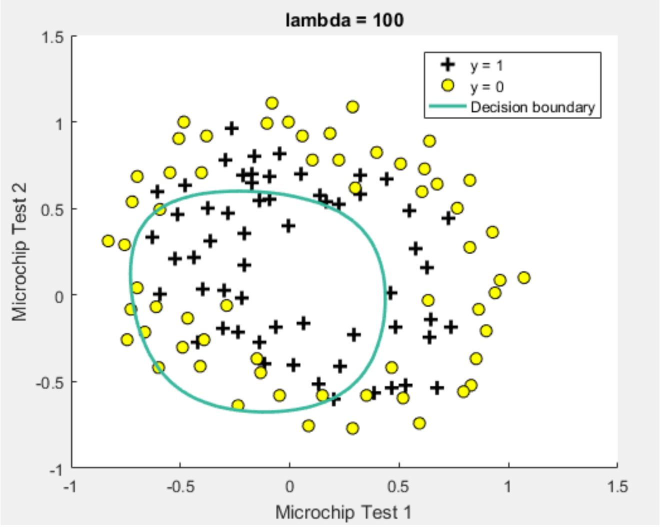

Figure 4 Training data with decision boundary (lambda=100)