ITU, Computer Engineering Dept. BLG454E - Learning From Data HW2

Group Name:Samaritan

Yunus Güngör – 150150701

Bengisu Güresti – 150150107

Q1)

In this homework logistic regression used to create a linear line that separates class 1 and class 2's data. Given data has 2 features and 2 different classes.

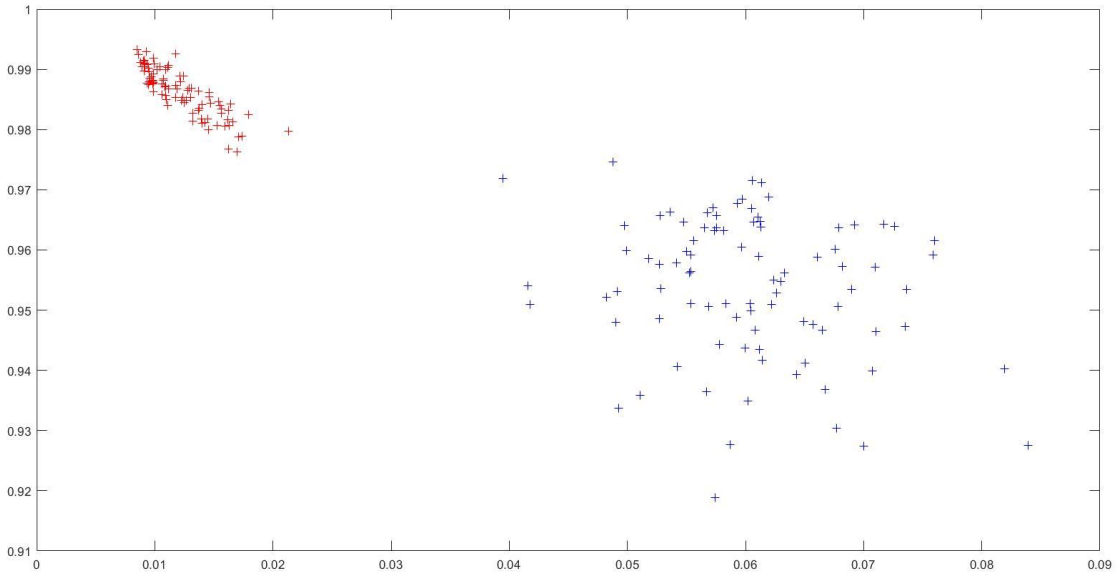

Figure 1.1: Given data (red represents class 0 and blue represents class 1)

Since it is a binary classification logistic regression can be applied easily. Decision boundary calculated as

$$g\big(f(x_1, x_2)\big) = \frac{1}{2}$$

$g(s)$ is sigmoid function and $f(x_1, x_2)$ is model that is going to be constructed.

$$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$$

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

Decision boundary is where sigmoid function equals $\frac{1}{2}$.

$$(1.2) \quad g\big(f(x_1, x_2)\big) = \frac{1}{1 + e^{-f(x_1, x_2)}} = 0.5$$

To plot line in two dimensions x_2 must be written in the form of $f(x_2)$.

$$0.5 + 0.5e^{-f(x_1, x_2)} = 1$$

$$0.5e^{-f(x_1,x_2)} = 0.5$$

$$e^{-f(x_1,x_2)} = 1$$

$$-f(x_1,x_2) = 0$$

$$w_0 + w_1x_1 + w_2x_2 = 0$$

$$(1.3) \qquad x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

Equation 1.2 leads to equation 1.3. Even though equation 1.3 does not have duty while calculating w_0, w_1, w_2 it is necessary to plot decision line in 2 dimensions.

While calculating w_0, w_1, w_2 gradient descent used to minimize cross-entropy function. Calculations continued until $\Delta w_j < 1.5$ because of the complexity of the algorithm and lack of computing power in personal computer. Every step calculated as:

$$\Delta w_j = -\eta\, \partial E\, \partial w_j = \eta \sum_t^n (r^t - y^t) * x_j^t$$

Where n is training data count , j is feature dimension and η is a constant to control learning rate. Note that a feature added to the system where $x_j^t = 1$ to get fixed constant w_0. And in this experiment η is taken as 0.01. In figure 1.4 gradient approach can be seen easily. And in figure 1.5 final decision boundary can be seen. More info about the approach can be found at source code.

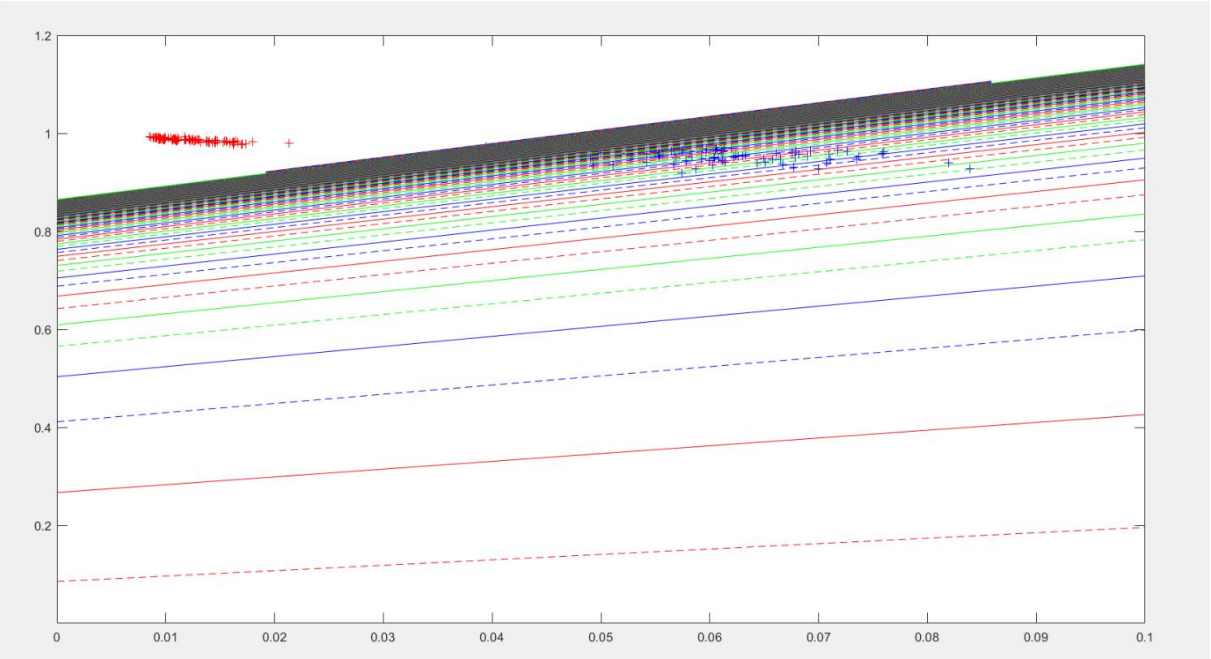

Figure 1.4 : Every step of decision boundary

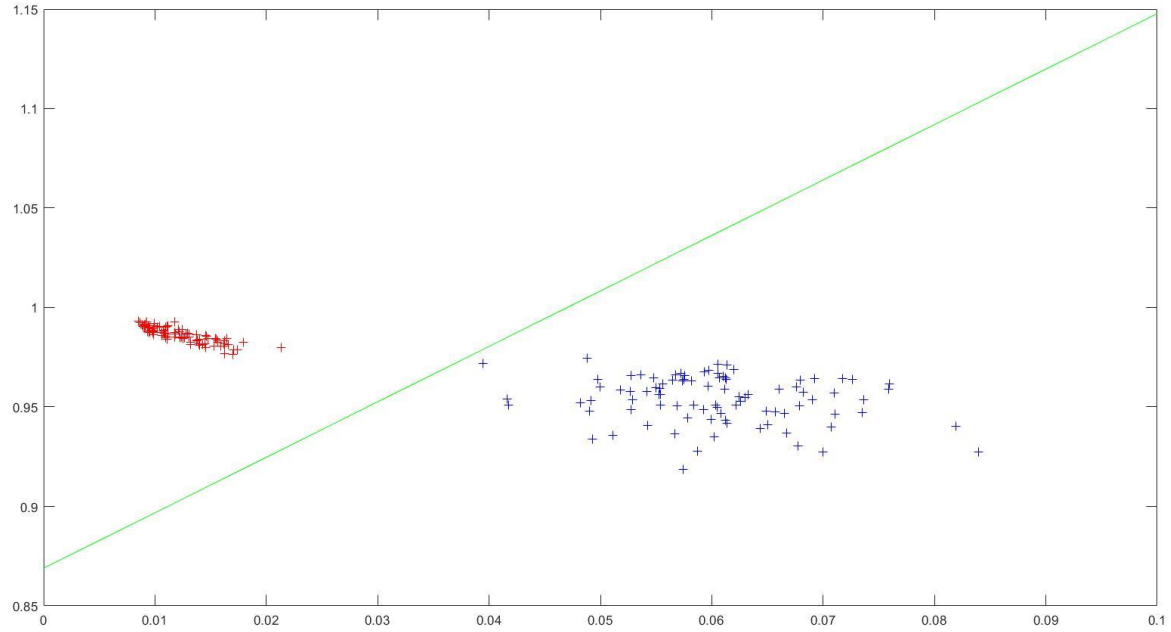

Figure 1.5:Calculated decision boundary

Calculated function is:

$$f(x_1, x_2) = 6{,}48525069490166 + 20{,}7421446604477x_1 - 7{,}47039008181205x_2$$

Calculated accuracy is : 100%. Since there was not much data avalible this value can be misleading.

Q2) If learning rate is too high iterations will step back and forth constantly.

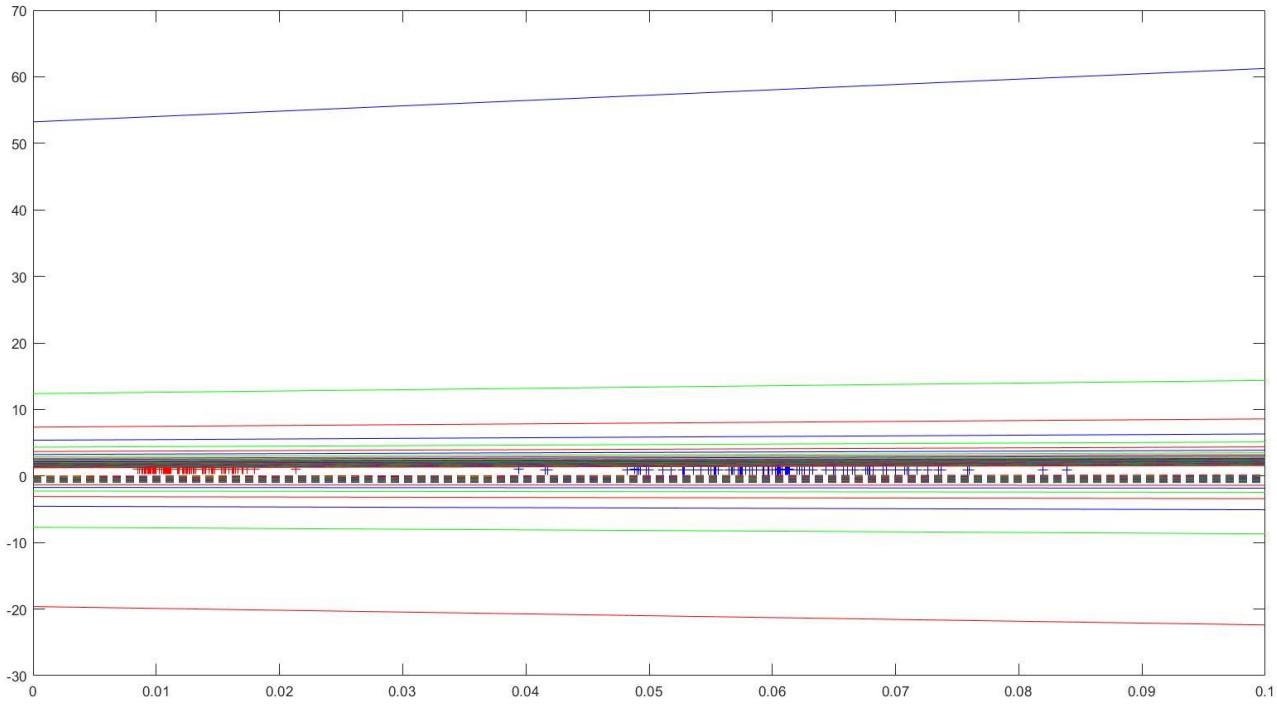

Figure 2.1: When $\eta = 0.2$ iterations steps back and forth

It can be seen in the figure 2.1, when learning rate is too high iteration number we need to get a good model increses because gradient descent happens back and fort. In figure 2.1 Data is between $0.9 - 1$ and each line is an iteration step.

When learning rate is too low iteration will not converge fast enough or will not converge at all. This situation can be seen at figure 2.2.

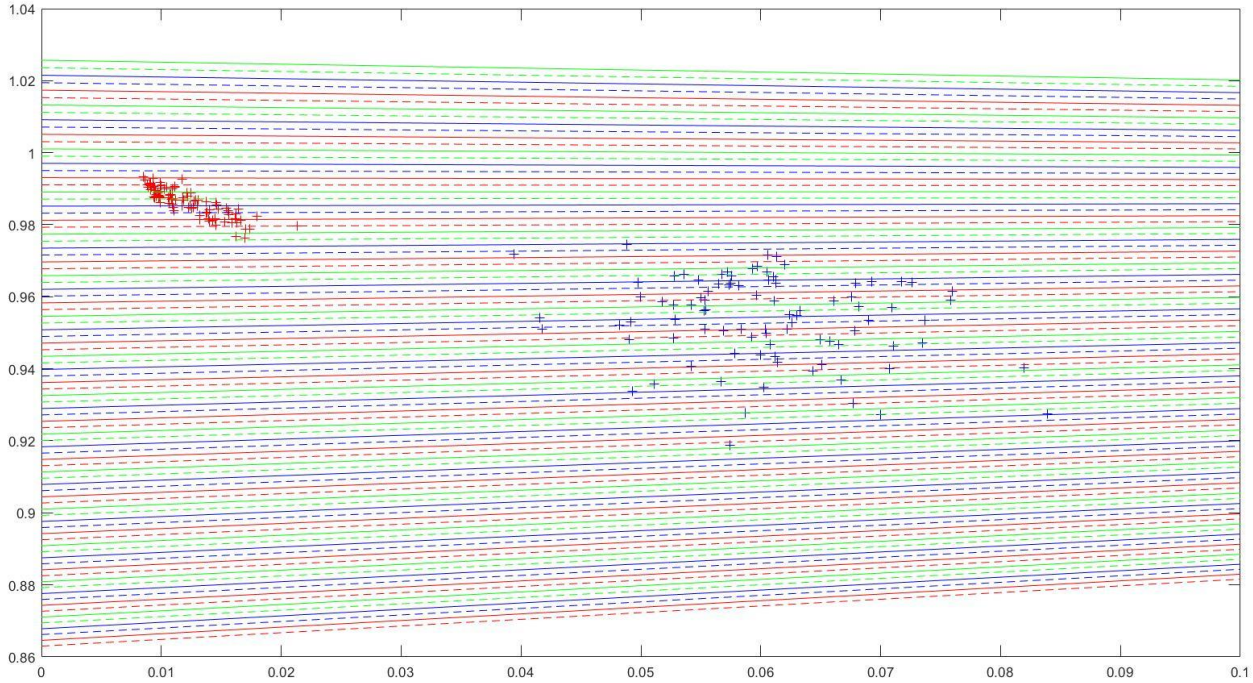

Figure 2.2: When $\eta = 0.00001$ iterations does not converge.

Q3)

$$logit\big(P(C_1|x)\big) = \log\left(\frac{P(C_1|x)}{1 - P(C_1|x)}\right) = w^T x + w^0$$

This function can be assumed true for 2 variables. Using of the classes as a base class, this function can be extended for multiple classes. This base function can be called C_K. Assuming,

$$\log\left(\frac{p(x|C_i)}{p(x|C_K)}\right) = w_i^T x + w_{i0}$$

$$(3.1) \quad \frac{P(C_i|x)}{P(C_K|x)} = e^{w_i^T x + w_{i0}}$$

Replacing w_{i0} with $w_{i0}^0 + \log\left(\frac{P(C_i)}{P(C_K)}\right)$ gives the equation

$$\sum_{i=1}^{K-1} \frac{P(C_i|x)}{P(C_K|x)} = \frac{P(C_K|x)}{1 - P(C_K|x)} = \sum_{i=1}^{K-1} \exp\left[w_i^T x + w_{i0}\right]$$

$$(3.2)\quad P(C_K|x) = 1/(1 + \sum_{i=1}^{K-1} \exp\left[w_i^T x + w_{i0}\right])$$

Using 3.1 and 3.2 probabilty of being a class can be calculated as:

$$P(C_i|x) = \frac{\exp\left[w_i^T x + w_{i0}\right]}{1 + \sum_{i=1}^{K-1} \exp\left[w_i^T x + w_{i0}\right]}$$

Normalizing probibilites, we can write:

$$y_i = P\hat{}(C_i|x) = \frac{\exp\left[w_i^T x + w_{i0}\right]}{\sum_{i=1}^{K} \exp\left[w_i^T x + w_{i0}\right]}$$

Sample likelihood is:

$$\prod_t \prod_i (y_i^t)^{r_i^t}$$

And using cross entropy; error function is:

$$-\sum_t \sum_i r_i^t log y_i^t$$

Appliying gradient descent:

$$\Delta w_j = \eta \sum_t (r_j^t - y_j^t) x^t$$