Machine Learning - hw2

Huiting Hong A061610

November 26, 2017

Abstract

This assignment is going to discuss 3 problems, which are Bayesian-Inference-for-Gaussian, Bayesian-Linear-Regression and Logistic-Regression respectively. This report will focus more on discussing the problem while implementation details please go to my github repository[1] to see more detail.

1 Bayesian Inference for Gaussian

1.1 Derive the posterior distribution of precision matrix Λ

Please see the hw1-1.pdf file for derivation detail.

1.2 Find the MAP solution to Λ for N = 10, 100 and 500

The idea on solving the Λ is to maximize the posterior distribution, $p(\Lambda|X)$. To find the index which maximize the posterior distribution, we take the first derivative of the $p(\Lambda|X)$ with respect to Λ, and assign it to zero. After several calculation (Please see hw1-2.pdf for more details), we can find the Λ which help us achieve the maximum of the posterior distribution.

The calculation result concludes:

$$\Lambda = (\nu_\Lambda - D - 1) * W_\Lambda \tag{1}$$

and the results are as follows:

$$\Lambda_{MAP} = \begin{bmatrix} 3.66978277 & -1.57678899 \\ -1.57678899 & 1.08166042 \end{bmatrix}, for N = 10$$

$$\Lambda_{MAP} = \begin{bmatrix} 6.11438285 & -2.6848374 \\ -2.6848374 & 2.14480209 \end{bmatrix}, for N = 100$$

$$\Lambda_{MAP} = \begin{bmatrix} 7.12490328 & -3.19947052 \\ -3.19947052 & 2.56317049 \end{bmatrix}, for N = 500$$

Too see more coding implementation detail please refer to my github repository[1].

2 Bayesian Linear Regression

2.1 Compute m_N, S_N

Base on the formula given in the text book, since we already assume $p(w) = \mathcal{N}(w|m_0 = 0,\ S_0^{-1} = 10^{-6}I)$, we can get:

$$m_N = \beta * S_N * \Phi^T * t \tag{2}$$

$$S_N^{-1} = \alpha * I + \beta * \Phi^T * \Phi \tag{3}$$

which $\beta = 1, \alpha = 10^{-6}$. Here we define

$$\Phi = \begin{bmatrix} \phi_0 & \phi_1 & \dots & \phi_{M-1} \end{bmatrix}$$

where the basis function, $\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$, the logistic sigmoid function, $\sigma(a)$, is defined in (3.6) in text book. The parameter setting for basis function is: $\mu_j = \frac{2j}{M}, s = 0.1, j = 0, 1, ...M-1, whichM = 7$.

The results are as follows:

$$m_N \approx \begin{bmatrix} 0.655 & 6.579 & 4.457 & -5.177 & 0.293 & -2.701 & -12.9357 \end{bmatrix}, forN = 10$$

$$S_N \approx \begin{bmatrix} 11.648 & -44.711 & 56.945 & -52.718 & 47.298 & -22.842 & 4.932 \\ -44.711 & 184.346 & -241.193 & 224.347 & -201.414 & 97.274 & -21.003 \\ 56.945 & -241.193 & 323.890 & -310.620 & 280.565 & -135.584 & 29.278 \\ -52.718 & 224.347 & -310.620 & 315.362 & -290.107 & 140.761 & -30.440 \\ 47.298 & -201.414 & 280.565 & -290.107 & 273.563 & -136.589 & 30.116 \\ -22.842 & 97.274 & -135.584 & 140.761 & -136.589 & 74.111 & -19.792 \\ 4.932 & -21.003 & 29.278 & -30.440 & 30.116 & -19.792 & 9.046 \end{bmatrix}, forN = 10$$

$$m_N \approx \begin{bmatrix} -1.294 & 14.930 & -7.773 & 8.308 & -12.021 & 2.834 & -13.789 \end{bmatrix}, forN = 15$$

$$S_N \approx \begin{bmatrix} 3.906 & -11.794 & 11.781 & -6.607 & 3.739 & -1.453 & 0.527 \\ -11.794 & 44.378 & -48.982 & 27.878 & -15.826 & 6.156 & -2.234 \\ 11.781 & -48.982 & 57.647 & -35.413 & 20.680 & -8.101 & 2.948 \\ -6.607 & 27.878 & -35.413 & 26.819 & -17.834 & 7.398 & -2.772 \\ 3.739 & -15.826 & 20.680 & -17.834 & 14.841 & -8.766 & 3.958 \\ -1.453 & 6.156 & -8.101 & 7.398 & -8.766 & 10.054 & -6.802 \\ 0.527 & -2.234 & 2.948 & -2.772 & 3.958 & -6.802 & 6.249 \end{bmatrix}, forN = 15$$

$$m_N \approx \begin{bmatrix} -1.913 & 17.122 & -9.553 & 8.012 & -10.108 & -1.712 & -9.891 \end{bmatrix}, forN = 30$$

$$S_N \approx \begin{bmatrix} 1.364 & -3.596 & 3.410 & -2.037 & 1.167 & -0.393 & 0.102 \\ -3.596 & 11.940 & -12.899 & 7.901 & -4.552 & 1.538 & -0.402 \\ 3.410 & -12.899 & 15.575 & -10.904 & 6.583 & -2.255 & 0.593 \\ -2.037 & 7.901 & -10.904 & 10.316 & -7.440 & 2.797 & -0.770 \\ 1.167 & -4.552 & 6.583 & -7.440 & 7.484 & -4.464 & 1.512 \\ -0.393 & 1.538 & -2.255 & 2.797 & -4.464 & 4.733 & -2.539 \\ 0.102 & -0.402 & 0.593 & -0.770 & 1.512 & -2.539 & 2.265 \end{bmatrix}, forN = 30$$

$$m_N \approx \begin{bmatrix} 0.256 & 9.392 & 0.077 & 0.422 & -4.636 & -4.208 & -9.332 \end{bmatrix}, forN = 80$$

$$S_N \approx \begin{bmatrix} 0.422 & -0.775 & 0.524 & -0.265 & 0.139 & -0.065 & 0.022 \\ -0.775 & 1.869 & -1.695 & 0.943 & -0.511 & 0.241 & -0.084 \\ 0.524 & -1.695 & 2.191 & -1.712 & 1.048 & -0.509 & 0.179 \\ -0.265 & 0.943 & -1.712 & 2.176 & -1.816 & 0.971 & -0.350 \\ 0.139 & -0.511 & 1.048 & -1.816 & 2.193 & -1.593 & 0.638 \\ -0.065 & 0.241 & -0.509 & 0.971 & -1.593 & 1.761 & -0.982 \\ 0.022 & -0.084 & 0.179 & -0.350 & 0.638 & -0.982 & 0.801 \end{bmatrix}, for N = 80$$

Too see more coding implementation detail please refer to my code[3].

2.2 Generate five curve samples from the p(w|t)

First we randomly sample five W from the posterior distribution, and then draw the curve by applying the formula:

$$y(x, w) = W^T * \Phi(x) \tag{4}$$

The results are as follows:

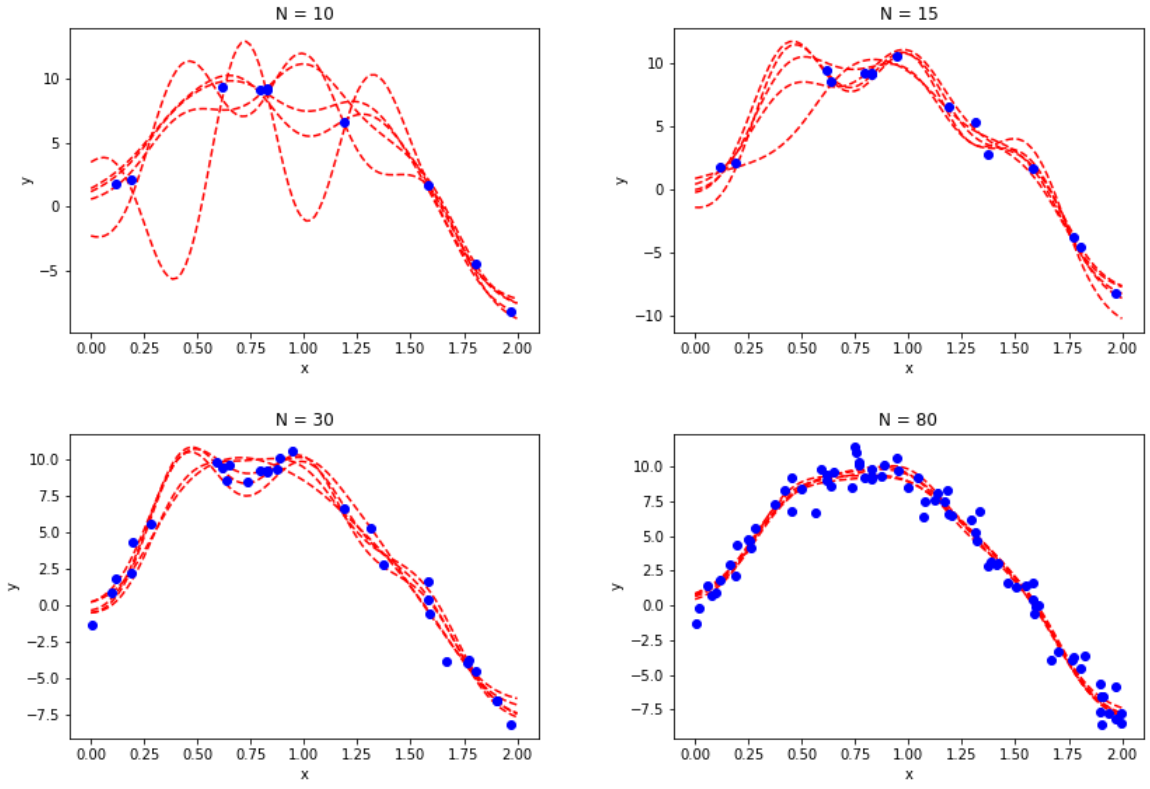

Figure 1: The five curve samples from the posterior distribution, we could see that as the N increases, the sample result will fit better to the data.

2.3 Plot the predictive distribution, p(t|x,t,α, β)

From the formula in text book, we could know that :

$$mean = m_N^T * \phi(x) \tag{5}$$

$$variance(\sigma_N^2(x)) = \frac{1}{\beta} + \phi(x)^T * S_N * \phi(x) \tag{6}$$

The results are as follows:

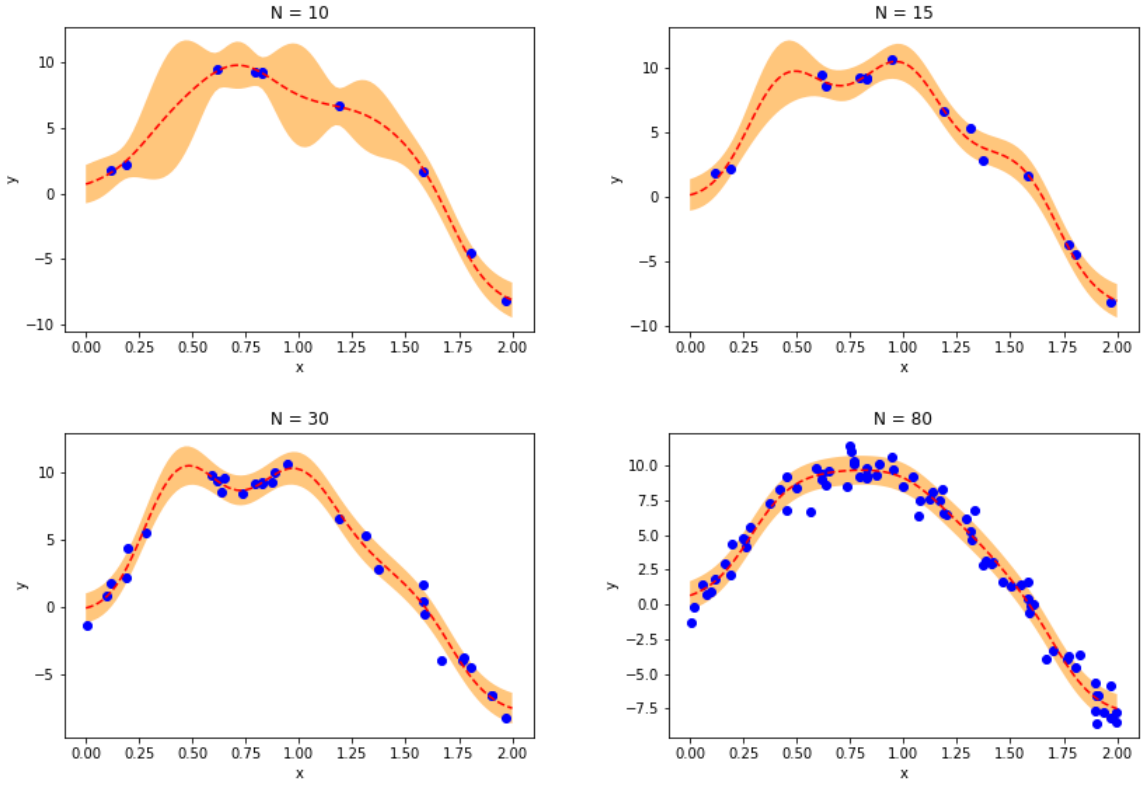

Figure 2: The target t are drawn in blue dots, and the red curve is the mean curve base on equation(5). The orange region is the the region of variance with one standard deviation on either side of the mean curve.

Too see more detail please refer to my code[3]

3 Logistic Regression

3.1 Let E(W) converge

Base on the Wine dataset given by TA, we are going to use online learning[5] to do the classification problem using logistic regression. Base on the Newton-Raphson algorithm

mentioned in the text book, we know that the formulas are:

$$W^{new} = W^{old} - H^{-1} * \nabla_{w_j} E(W) \tag{7}$$

which $H = \sum_{n=1}^{N} y_{nk} * (I_{kj} - y_{nj}) * \phi_n * \phi_n^T$ and $\nabla_{w_j} E(W) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) * \phi_n$.
For each update of W, we can then update the value of y, and then update the H,$\nabla_{w_j} E(W)$ and update W again, and so on and so forth. The learning curve and the accuracy on the train dataset are as follows:

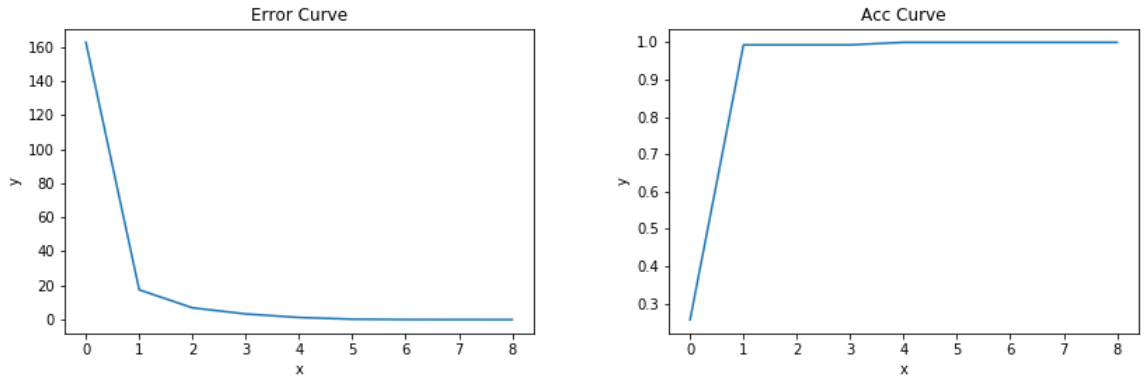

Figure 3: I set the epsilon to be 0.005, and as the learning curve shown in the left figure, the Error will decrease under 0.005 after 9 epochs. The accuracy curve on train dataset is shown in the right figure.

Too see more detail please refer to my code[4]

3.2 Classification result of test data

predict-result = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Too see more detail please refer to my code[4]

3.3 Plot the histogram of each variable in training data

Please see Figure[4].
Too see more detail please refer to my code[4]

3.4 Global minimum verification

Too see more detail please refer to hw3-4.pdf

3.5 Choose most contributive variables

From observation, I choose 7th and 10th to be the most contributive variables, which their histogram are shown in Figure[5].

We could see the plot of samples in 2D graph, which are shown in Figure[6].

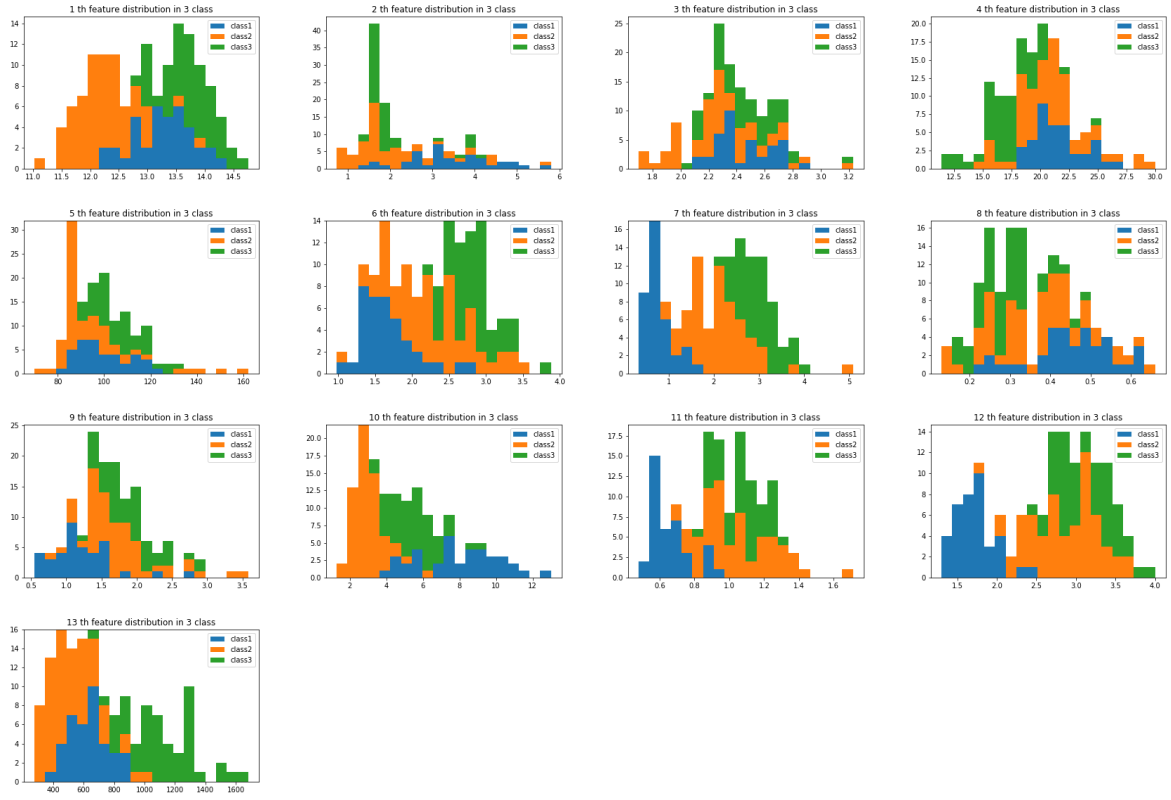

Figure 4: The distribution of each variable, from 1 to 13, which blue map to class 1, orange map to class2 and green map to class3

3.6 Use the 2 most contributive variables to calculate Error learning curve and predict on test set

In this section, I can't find the way to let the error curve converge, which the learning curve and accuracy on training dataset are as shown in Figure[7].

I feel that the reason why the Error can't converge might because I choose the wrong contributive variables or it do need more variables to help it achieve the global minimum, or it will stuck in the local minimum in this case and can't find the way to converge. (i.e. only 2 most contributive variables might not be enough.).

Base on the non-converge Error function result, we can still do prediction on the test dataset while the result should be rough and poor :
predict-result = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0]

References

[1] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw2.

[2] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw2/blob/development/Prob

[3] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw2/blob/development/Prob

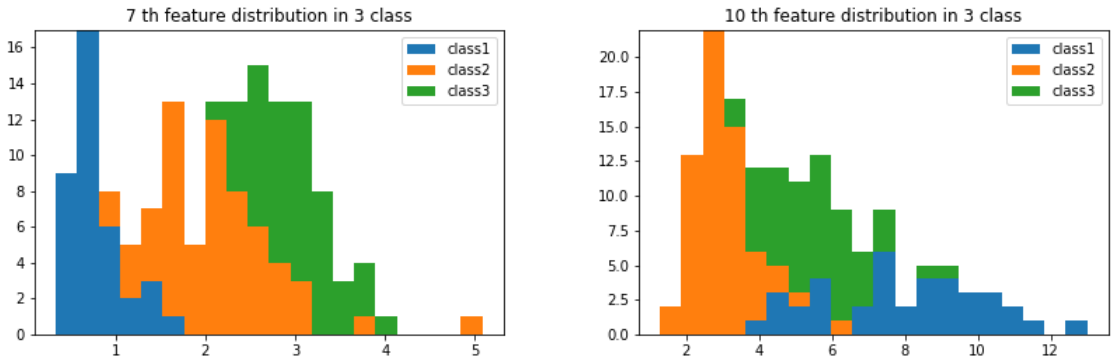

Figure 5: We could see that each class distribution is quite separated to each other compare to other variables

[4] https://github.com/w102060018w/2017_NCTU_MachineLearning_hw2/blob/development/Prob

[5] *Batch learning and Online learning* https://en.wikipedia.org/wiki/Online_machine_learning

[6] *Newton method* https://goo.gl/PG49z7.

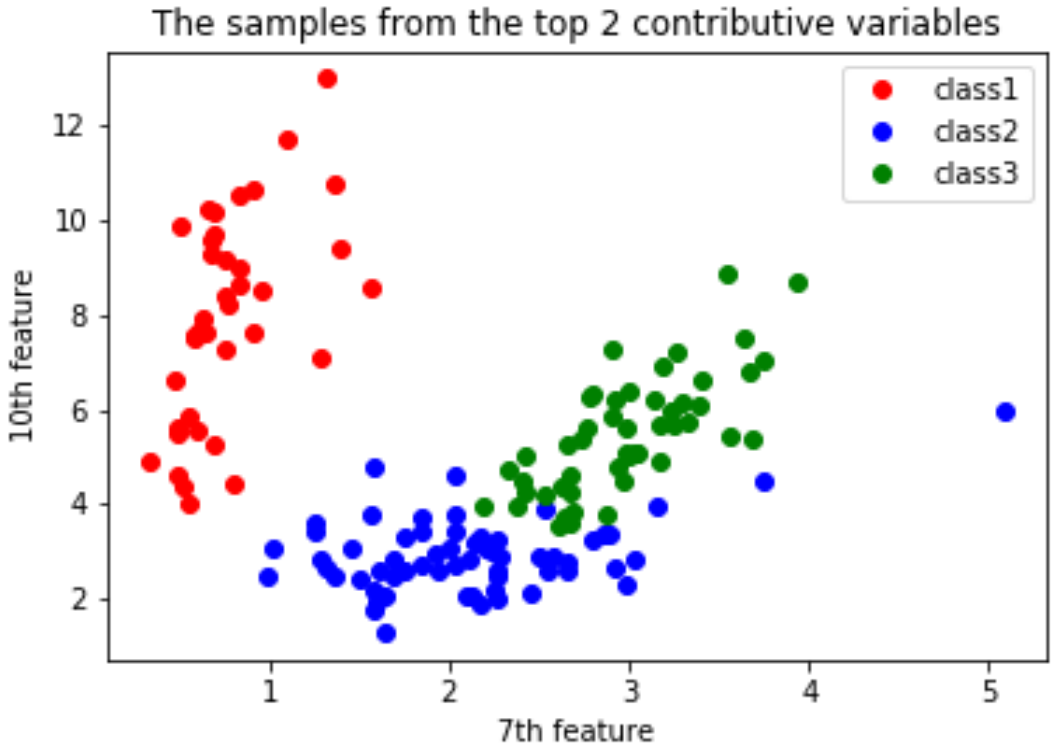

Figure 6: The samples from 7th and 10th variables plot in 2D graph

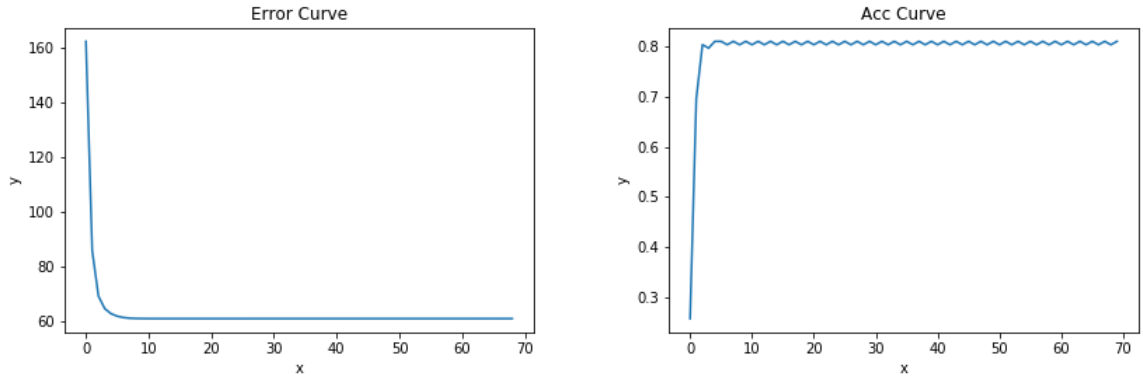

Figure 7: The Error curve and the accuracy on train dataset. The error will not converge under 0.005, and it will oscillate around 60, and so does the accuracy curve which also oscillates around 0.8.