Binary Classification using 2-layer Neural Network

Department of Computer Software Engineering
College of Engineering, Hanyang University
Gaon Choi(최가온), Student ID: 2019009261

Introduction

The purpose of this experiment is to solve the Linear Regression problem based on a newly designed model architecture. In this experiment, we used 2-layered neural network. The diagram below represents the architecture of 2-layer neural network.

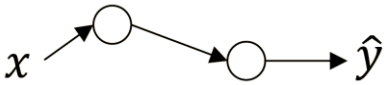

Figure 1. Model Architecture

A new dataset was used to train the model and estimate model's performance. The dataset generation method is as follows.

- Random value between 0 and 360 is extracted from Uniform Distribution and determined as an x value.
- The value extracted from No.1 is passed into radian conversion function and then put into the cosine function.
- If the cosine value is positive, y is 1, otherwise 0(if it is negative).

However, with the dataset created with the ways above, the performance of our model was extremely lower. (it was near 50% regardless of other hyper-parameter such as epochs, α, etc.) After the process of input normalization, mapping the input from $X \in [0, 360]$, to $X \in [0, 2\pi]$, the accuracy of our model was always higher than 95% when $\alpha \geq 0.5$ and epoch = 10,000. With the structure above, the model accuracy was about 50.0%, which means no learning happened. In this experiment, we added one more node to the hidden layer, to improve the accuracy. Cosine dataset cannot be classified with one sigmoid function. We visually scattered the whole dataset and concluded that at least two neurons are required to fully train the model.

Experiment

1. Estimated unknown function parameters W & b

We performed 100 independent experiments with epoch=10,000, α=2, creating new datasets each time. The average of train accuracy was 99.5073%, and the average of test accuracy was 99.515%.The left-side figure represents the distribution of w1 and w2. The right-side figure represents the distribution of b1 and b2. (Figure-2)

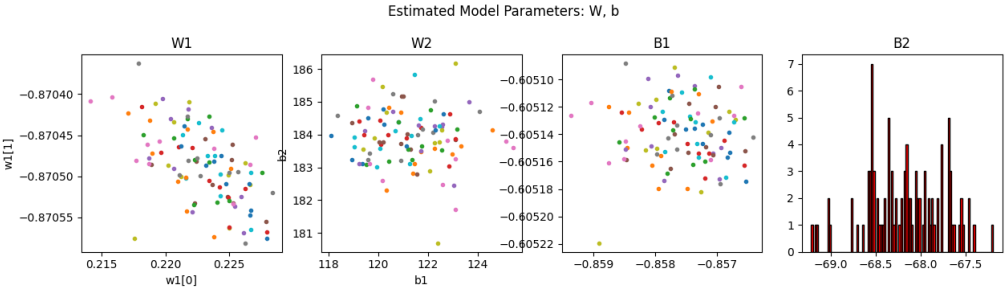

Figure 2. Estimated Parameters

2. Empirically determined (best) hyper parameter, α

When the range of x is [0, 360], the accuracy was found to be under 50% for all α values. When the range of x is normalized, converting to a radian value, this phenomenon was significantly improved, and almost all of the accuracy was ≥ 99% when epoch=10,000, only when $\alpha \geq 0.1$. As the value of α gets higher, the test cost decreased steeper. However,

when the α is higher(≥10), the fluctuation phenomenon got intensified. (Figure-3)

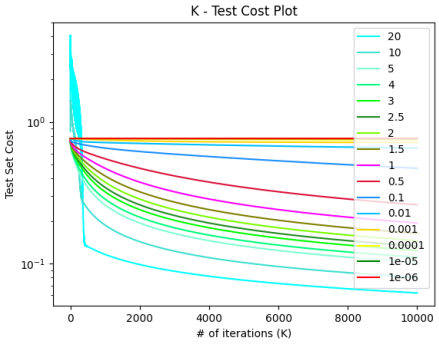

Figure 3. K - Test Cost Plot

3. Accuracy

3-1. Accuracy comparison by the value of m (the size of train set) (α=2.5)

One hundred independent experiments were performed per each set. While keeping the value of n and k constant, the trend of accuracy when m is gradually increased is as follows.

	m=10, n=1000, K=5000	m=100, n=1000, K=5000	m=10000, n=1000, K=5000
Accuracy (with 'm' train samples)	66.5%	75.26%	95.09%
Accuracy (with 'n' test samples)	53.30%	63.18%	95.03%

3-2. Accuracy comparison by the value of K (the number of iterations) (α=2.5)

One hundred independent experiments were performed on three cases where the value of K was 10, 100, and 5000, respectively. For each experiment, the dataset was newly created and proceeded every time. The accuracy below is the average of the results derived from 100 independent experiments.

	m=10000, n=1000, K=10	m=10000, n=1000, K=100	m=10000, n=1000, K=5000
Accuracy (with 'm' train samples)	40.67%	68.43%	97.30%
Accuracy (with 'n' test samples)	40.39%	68.53%	97.24%

The figure-4 represents our model and the data distribution with different α value. In can be visually confirmed that the larger the value of α, the better the model fits the data. When α=0.1, our model got more than 90% accuracy, although the visualized version is far from the data distribution.

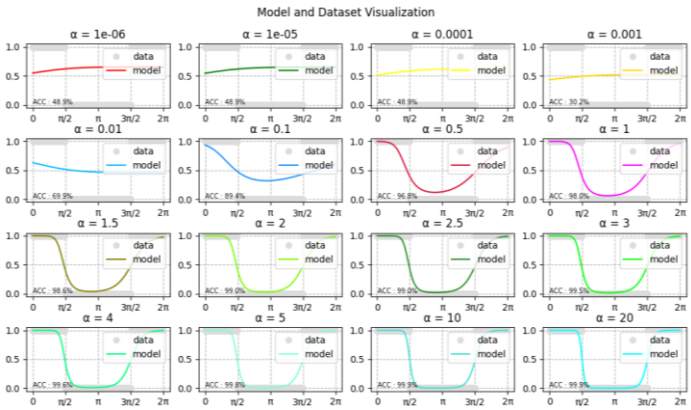

Figure 4. model & test data distribution

Discussion

We implemented a neural network, analyzed the dimensions of each matrix, and directly followed the partial differential computation process, which helped to have a clear understanding of gradient descent, forward propagation, and back propagation. Analyzing the dataset was helpful to design a model. Cosine data cannot be classified with only one sigmoid, thus using more than two nodes contributed to improving accuracy.