ITE4053 – practice1 report

Junyeong Park

1 Methods

In these experiments, we used dataset consisted of two-dimensional inputs and one binary output. Each sample of dataset has an output of 1 when the sum of two elements of input is positive, otherwise 0. And we used the simple logistic regression model as follows:

$$f(\mathbf{x}) = \frac{1}{1 + \exp\left(-\mathbf{w}^\top \mathbf{x} - b\right)} \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ is the parameters of the model, and $\mathbf{x} \in \mathbb{R}^2$ is the input of the model. The range of the output of the model is between 0 to 1. If $f(\mathbf{x}) > 0.5$, we treated the output of the model as 1, and if $f(\mathbf{x}) \leq 0.5$, we treated it as 0.

We checked three things: the relation between hyperparameter α and performance, the relation between the number of training samples and performance, and the relation between the number of iterations and performance. In these experiments, hyperparameter α candidates are 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 4, 8, 10, 15 and 20.

2 Results

2.1 Estimated parameters

The norm of parameter $\mathbf{w}$ is affected by the hyperparameter α. The greater α, the bigger the norm of the parameter $\mathbf{w}$. The correlation coefficient is about 0.975. Conversely, there is relatively less correlation between the bias and the hyperparameter α. The correlation coefficient is about -0.139.

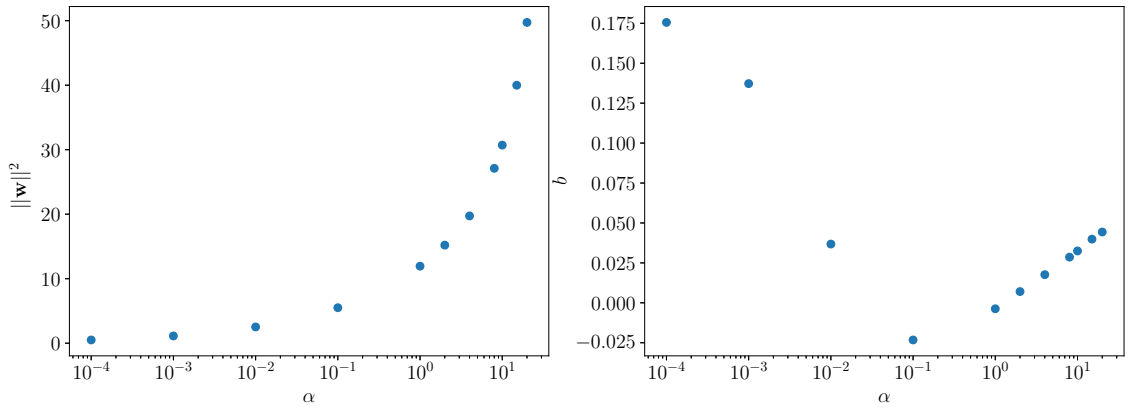

Figure 1: Left: norm of parameter $\mathbf{w}$ according to hyperparameter α. Right: bias according to hyperparameter α

The best parameters that have the smallest cost value for the test dataset are as follows:

$$\mathbf{w} = \begin{bmatrix} 35.199 & 35.131 \end{bmatrix}^\top, \; b = 0.044 \tag{2}$$

2.2 Best hyperparameter α

To get the best hyperparameter, we tested various candidates. In Fig. 2, the larger α, the faster it converges to a small cost value. However, when the hyperparameter α is too large, the model has a numerical issue, floating point overflow. Therefore, we found that $\alpha = 15$ is the best parameter for this task.

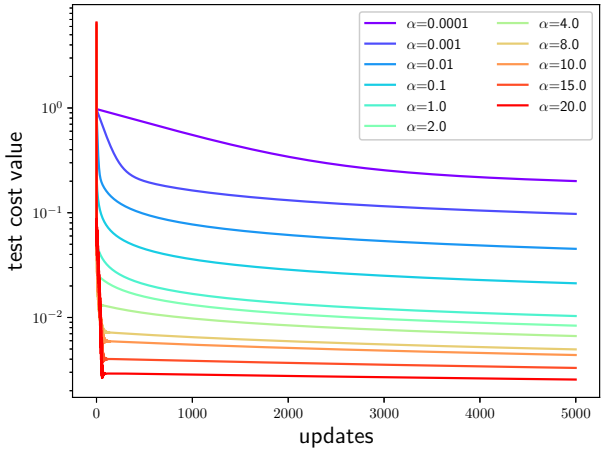

Figure 2: Model training performances for each hyperparameter.

2.3 Accuracy

We frozen random seeds to reduce noise that is not an associated variable of the experiments. Below results are from the same datasets and the same initial parameters.

	(10, 1000, 5000)	(100, 1000, 5000)	(10000, 1000, 5000)
Accuracy (training set)	100	100	99.7
Accuracy (test set)	88.4	98.8	99.7

Table 1: Performances according to the number of training samples. (m, n, K) denotes that the model was updated K times and that m training samples, and n test samples were used.

	(10000, 1000, 10)	(10000, 1000, 100)	(1000, 1000, 5000)
Accuracy (training set)	65.64	74.82	99.3
Accuracy (test set)	66.10	75.40	99.1

Table 2: Performances according to the number of updates. (m, n, K) denotes that the model was updated K times and that m training samples, and n test samples were used.

3 Discussion

In these experiments, we found four things. First, the large hyperparameter α causes the norm of the weights of the model to become quite large. It can help a model to converge fast, and also promote model instability. Figure 2 shows that in the case of the large α, test cost is decreased fast but relatively large in early. Hence, if the task or model is more complex, the large step size is able to annoy the training. Second, to get a model that has good generalization performance, we need a huge training dataset. In Table 1, using the small dataset ($m = 10$ or $m = 100$), accuracy for training samples is perfect, but accuracy with test samples is bad. This means that the model trained with a small dataset has a biased perspective that interprets the distribution of the entire data wrongly, so that it does not properly deal with novel data. Therefore, we need as many possible as samples to train the model. Third, doing more updates can increase the performance of the model, then if possible, we have to do more and more update iterations. Lastly, in Fig. 1, for all conditions, cost value is saturated in the long run. Hence, if performance is not improved, we can stop training early to save computational cost.