Deep Learning in Data Science, DD2424

Short report on lab assignment 2 bonus
Two-layer neural network with multiple outputs

Hamid Dimyati

1. Introduction

The focus of this report is to implement several methods to achieve higher accuracy on the same model architecture and datasets as the main report on lab assignment 2. Some methods that will be presented in this report cover hyperparameter random search, increasing number of hidden nodes, and applying dropout. Further, an automatic method to find the best range of learning rate for cyclical learning rate is performed.

The tool used for this bonus assignment is also `python 3.7.3`, along with several packages including `numpy 1.16.4`, `pandas 0.25.3`, and `matplotlib 3.1.0`.

2. Results and discussion

2.1. Hyperparameter random search

After getting the optimal value of lambda (0.00197229) in the main task of the assignment, the next hyperparameter search would be the step size and total cycles. The methodology used in this experiment is similar to fine search presented in the main report. The detail process could be seen in Table 1.

Table 1. The best hyperparameter settings reached from random search technique

Iteration	Hyperparameters	
	Step size	Total cycle
0	[50, 1500]	[1, 10]
1	[1000, 1400]	[5, 9]
2	[1000, 1200]	[7, 7]
3	[1000, 1100]	[7, 7]
4	[1000, 1000]	[7, 7]
5	[1000, 1000]	[7, 7]

After 5 iterations, using cyclical learning rate $[1e-5, 1e-1]$, and batch size 100, it turns out that the best step size and total cycle are 1,000 and 7, respectively. The accuracy achieved on the validation dataset is 54.68%. However, those two best parameters would be adjusted in the next experiment to be able to end the learning at the lowest value of the learning rate.

2.2. Effect of the number of hidden nodes

Another idea to improve accuracy is to increase the number of hidden nodes. I varied the possible number of nodes to be tested on the data: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]. Figure 1 explains how the effect of increasing the number of hidden nodes to the model accuracy. As the number of hidden nodes goes up, the training accuracy could improve significantly. However, there would suffer an overfitting problem, as seen in Figure 1, when the number of hidden nodes reach 100, the validation accuracy could touch almost 55% while its training accuracy reaches almost 70%. One possible technique to handle this problem is to employ a dropout method, which will be explained in 2.3.

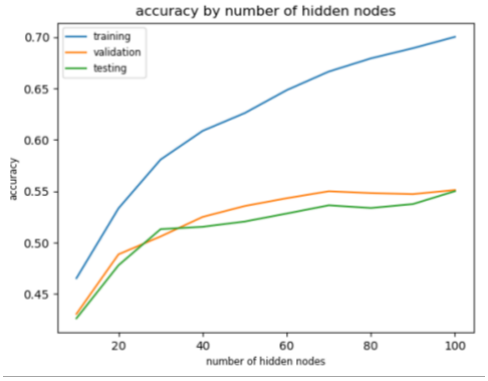

Figure 1. Accuracy of the training, validation and testing datasets as a function of the number of hidden nodes

2.3. Dropout

The idea behind this technique is to reduce a high number of hidden nodes by dropping out some random nodes based on a certain probability threshold in every iteration. Thus, there will be different models learned in every iteration, and the error would be the average of errors from all different models. Therefore, this dropout technique could also be viewed as an approximation to the ensemble method. Using cyclical learning rate $[1e-5, 1e-1]$, lambda 0.00197229, batch size 100, step size 1,125, 30 epochs, 6 cycles, and the probability of each node to be included as much 0.75, I got 63.32% accuracy on training data, 55.18% accuracy on validation data, and 54.04% accuracy on testing data. Figure 2 shows how the dropout could help handle the overfitting even though using the high number of hidden nodes.

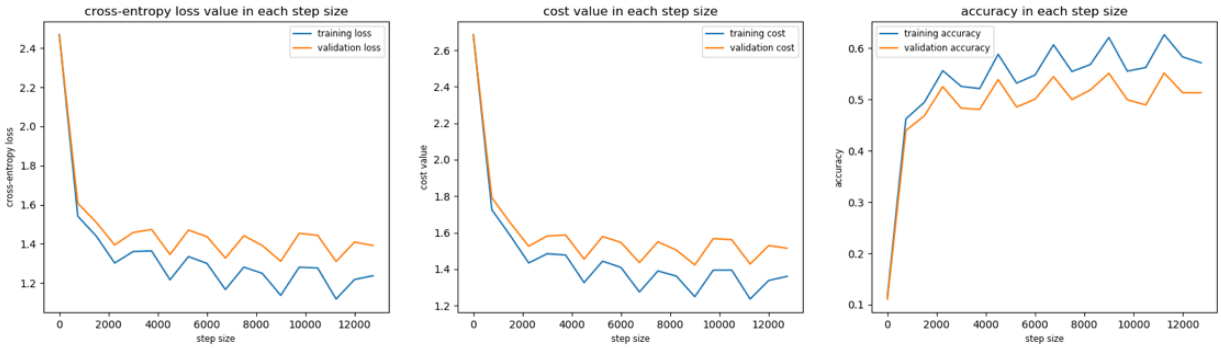

Figure 2. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from a model using dropout

2.4. Optimal range of learning rate

The last experiment to do is finding the best range of learning rates. Firstly, I search the best value within a broader range [1e-10, 1e-1], as shown in Figure 3.

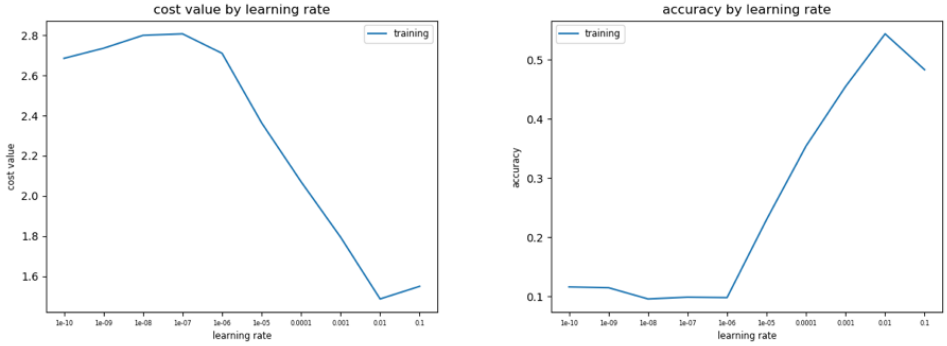

Figure 3. The cost value (left) and accuracy (right) on training data using different learning rates within [1e-10, 1e-1].

Based on those two figures, I could slice into a smaller range of [1e-3, 1e-1] as the two thresholds are where the cost value decreases toward the smallest value and start to increase again, as depicted in Figure 4. Thus, the range could be the best start for implementing the cyclical learning rate.

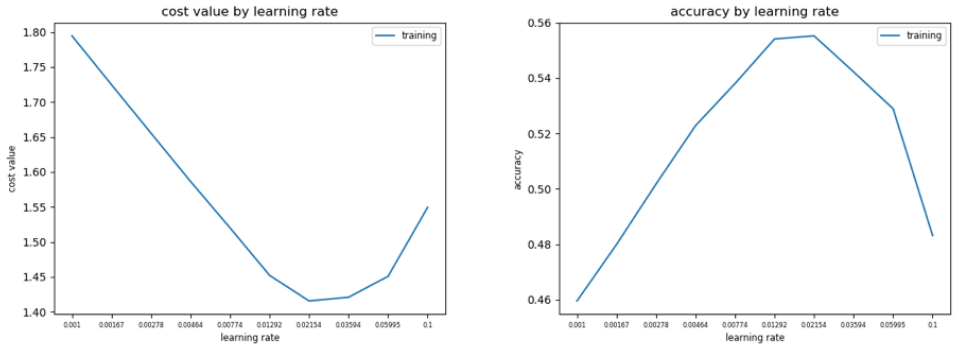

Figure 4. The cost value (left) and accuracy (right) on training data using different learning rates within [1e-3, 1e-1].

Using similar hyperparameter settings as done in 2.3 but with new minimum and maximum value of learning rate, [1e-3, 1e-1], I got 63.46% accuracy on training data, 56.02% accuracy on validation data, and 54.10% accuracy on testing data. It is concluded that there is a little improvement in the testing accuracy compared to the results in 2.3. Thus, selecting the best minimum and maximum value of the learning rate before performing a cyclical learning rate is quite critical.

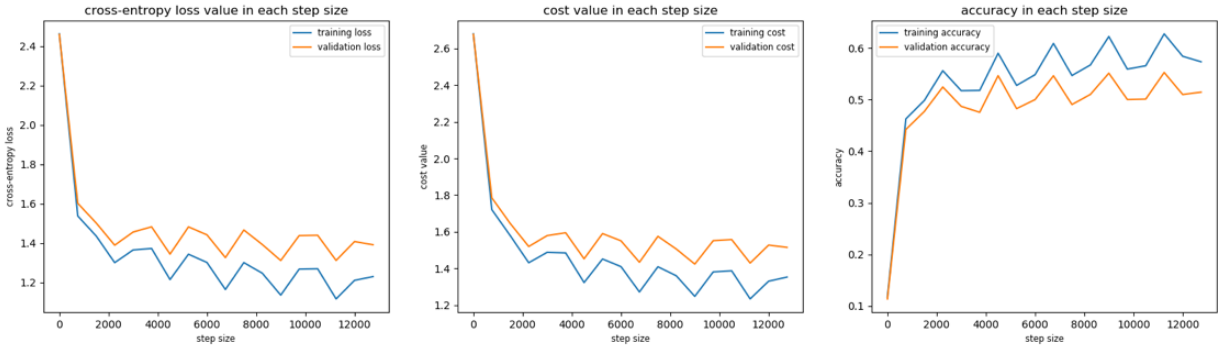

Figure 5. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from a model using a new range of learning rate

3. Final remarks

From this assignment, it could be learned that random search could be used for hyperparameter tuning as an alternative to the grid search. Besides, adding more hidden nodes could also potentially increase the accuracy with the cost of regularization drop due to overfitting. However, the dropout technique could come as a solution to the problem. Lastly, it is essential to pick the best minimum and maximum value of the learning rate before performing a cyclical learning rate since it could guarantee better performance (54.10%).