DL Assignment 2

Meinan Gou meinang@kth.se

1 Introduction

In this assignment, I trained and tested a two layer network with multiple outputs to classify images from the CIFAR-10 dataset. Mini-batch gradient descent was applied and cross-entropy loss calculated to the labelled training data and an L_2 regularization term on the weight matrix was used.

2 Functions

Compared with Assignment 1, I added the following functions to Assignment 2:

- cyclical_learning_rate(): Takes $\eta_{max}, \eta_{min}, n_s$ and t as arguments and returns η at the current update step.
- coarse_search_lamda(): Takes l_{min}, l_{max} and $best\_lamba$ as arguments and returns new $lamda$ value.

3 Results

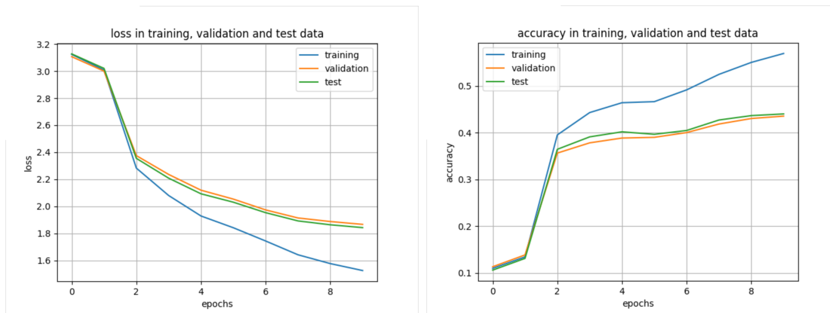
3.1 Comparison of computing gradients analytically and numerically

The function relative_error() was defined to calculate the relative error between analytical and numerical results. I used the first 20 samples in the training data. The formula is

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)}.$$

The maximum relative error of $W1, b1, W2$ and $b2$ are at level of e-6 or less, which satisfies the criterion.

3.2 Curves for cyclical learning rates with the default values



(a) loss (b) Accuracy

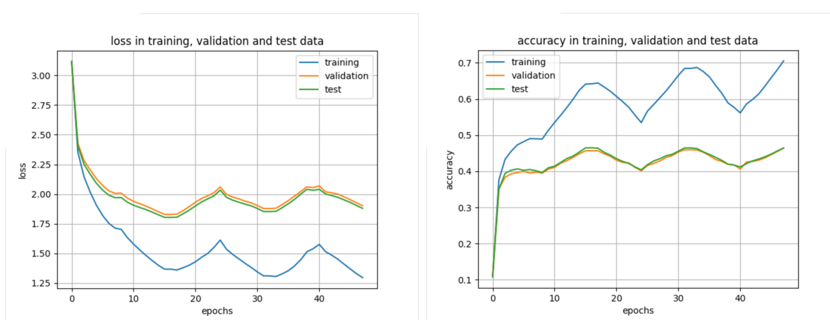
Figure 1: 1 cycle curves

Figure 1 shows the curves of cost and accuracy for one cycle of training with the following hyper-parameter setting: $eta\_min = 1e{-5}, eta\_max = 1e{-1}, lamda = 0.01$ and $n_s = 500$. Only one batch data was used for training. I plotted the curves at every epoch.

From Figure 1 (a), the final loss of the training data is 1.52, and the final loss of the validation data is 1.88.

From Figure 1 (b), the final accuracy of the training data is 0.577, and the final accuracy of the validation data is 0.435.

The best test accuracy is 0.441.



(a) loss (b) Accuracy

Figure 2: 3 cycle curves

Figure 2 shows the curves of cost and accuracy for three cycles of training with the following hyper-parameter setting: $eta\_min = 1e-5, eta\_max = 1e-1, lamda = 0.01$ and $n_s = 800$. Only one batch data was used for training. I plotted the curves at every epoch.

From Figure 2 (a), the final loss of the training data is 1.31, and the final loss of the validation data is 1.82.

From Figure 2 (b), the final accuracy of the training data more than 0.7, and the final accuracy of the validation data is 0.465.

The best test accuracy is 0.464.

3.3 Coarse Search: Range of lamda, number of cycles and 3 best hyper-parameter settings

During the Coarse search, I followed the instructions on the pdf file with the function coarse_search_lamda(). n_s and n_cycles were set to 900 and 2, respectively. The search range was e-5 to e-1. I generated 10 random lamda within the range and computed validation accuracy. Last, I used the command in the terminal " > a.txt" to save the lamda-accuracy pair into a .txt file, which would be used in fine search.

Following is the result of the coarse search:

λ	Validation Accuracy
0.00023949797371584832	**0.5066**
0.007086989381592453	0.488
0.0013928467795857747	0.4922
0.0024989152613350413	0.4924
0.01921305639929294	0.4686
0.0012451570260812114	**0.502**
0.0008135633078925343	0.4914
0.0002228836980124634	**0.4988**
0.0003423173447628359	0.4986
1.2417186056667177e-05	0.4926

Table 1: Initial random search of λ

I highlighted the values of λ that produce the highest validation accuracy. I explored them respectively in the fine search. The three best λ values are 0.00023949797371584832 (l_1), 0.0012451570260812114(l_2) and 0.0002228836980124634 (l_3).

3.4 Fine Search: Range of lamda, number of cycles and 3 best hyper-parameter settings

During fine search, I set n_s and n_cycles to 1800 and 3 respectively. Then I generated small numbers around the lamda with highest accuracy founded in coarse search with the following code:

$$l = l\_min + (l\_max - l\_min) * np.random.rand()$$

$$lamda = best\_lamba + np.random.choice([-1, 1], p = [0.5, 0.5]) * (10 * *l)$$

The following table shows the search result based on l_1.

λ	Validation Accuracy
0.00031927857737659634	0.5114
0.00022791990568291404	0.5114
0.00033828419837301826	**0.5182**
0.00028747331123609544	0.5134
0.0002693907843568422	0.5114
0.00022923456428761564	0.5128
0.00021407222656189448	**0.5158**
0.00027624540785932964	0.5128
0.00030729837397370407	0.5122
0.000251089140918953	0.5142

Table 2: Random search based on l_1

The following table shows the search result based on l_2.

λ	Validation Accuracy
0.0018534501607979474	0.5128
0.0014342591930241282	0.506
0.0008079337686589599	0.5008
0.0011129433461329482	0.51
0.0009837132136861291	0.5042
0.001376267937301526	0.5138
0.0010538569836390957	0.5114
0.0015408507162854071	0.5034
0.001058181278451133	**0.5166**
0.0009870817617452425	0.5104

Table 3: Random search based on l_2

The following table shows the search result based on l_3.

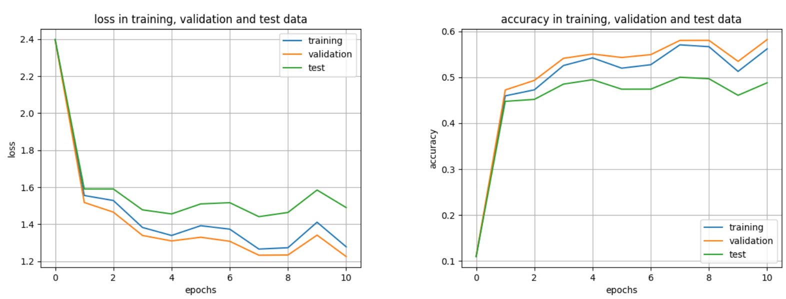
λ	Validation Accuracy
0.00027680401615186285	0.5044
0.00027513778824671194	0.5034
0.0002649883368011326	0.5018
0.00014586355728213288	0.5038
0.0001782899439074353	0.5064
0.0001995976933835882	0.5018
0.00030663958599504306	0.505
0.00018486556862452446	0.5038
0.0002429966067802218	0.51
0.00016698739438653564	0.5042

Table 4: Random search based on l_3

From above three tables, the best three λs are 0.000338, 0.001058 and 0.000214, which are highlighted in bold.

3.5 Final training and validation curves

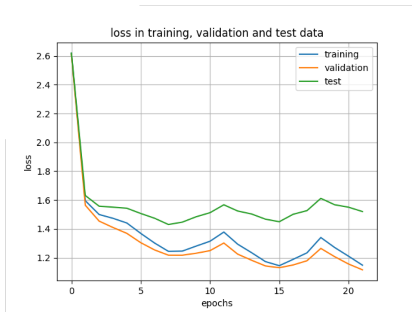
I tested with n_s=900 and 1800 respectively, using $\lambda = 0.000338$.



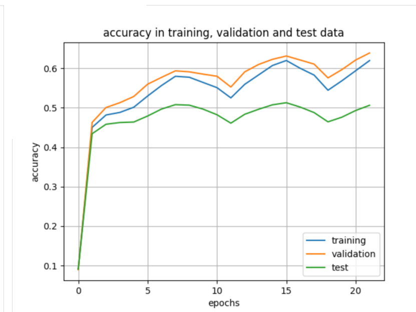
(a) loss (b) Accuracy

Figure 3: n_s=900, n_cycles=3

The best test accuracy is 0.5003.



(a) loss



(b) Accuracy

Figure 4: n_s=1800, n_cycles=3

The best test accuracy is 0.5129.

4 Conclusions

In the process of searching the best lamda values, I found that increasing n_s and n_cycles can improve validation and test accuracy. This is because increasing n_s can increase the number of epochs of training during one cycle, and increasing n_cycles increases the number of training process.

Also, I found that the best values of lamda are in a certain range. Too low or too high lamda would lead to divergence of cost and reduce test accuracy.