Assignment2

Linghui Zhou

April 2019

In this assignment, we train and test a two layer network.

1 Analytic Check

Firstly, I compare the analytic and numerical gradients to verify the gradient computation is bug free. To do so, I tested the relative error performance of the weight matrices and the bias. Four groups of the results are listed in Table 1. As suggested in the instruction, I fixed `h=1e-5` in the numerical calculations.

Lambda	0	0	0	0.001	0.001	0.001
Batch Size	1	50	100	1	50	100
gradient b_1	5.749e-11	6.936e-10	1.016e-09	5.749e-11	6.936e-10	1.061e-09
gradient b_2	2.459e-11	2.919e-10	4.378e-10	2.459e-11	2.919e-10	4.378e-10
gradient W_1	9.804e-11	8.753e-10	1.680e-09	1.218e-10	8.928e-10	1.702e-09
gradient W_2	4.948e-11	1.479e-10	2.373e-10	5.405e-11	1.504e-10	2.399e-10

Table 1: Relative error between analytically and numerically computed gradient values.

As we see from Table 1, with different parameters settings, the relative errors are all sufficiently small. Therefore, I can assume the gradient computations are correct.

Secondly, I did the sanity check. Here, I used 100 samples of the training data and turned the regularization off, i.e., `lambda=0`. Moreover, let `eta=0.01`. The training loss is plotted in Figure 1.

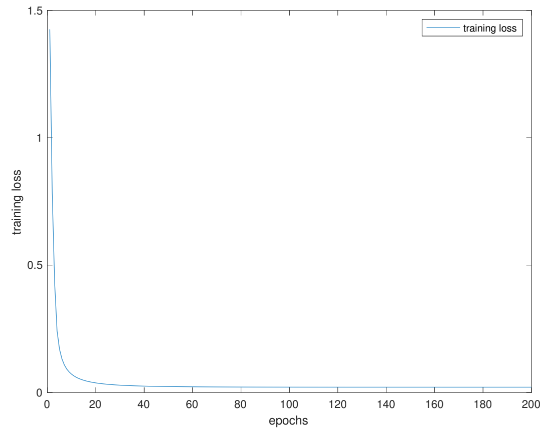
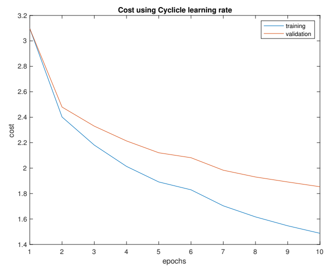


Figure 1: Plot of training loss. As the number of epochs increase, the training loss is decreases and it becomes sufficiently small after some epochs.

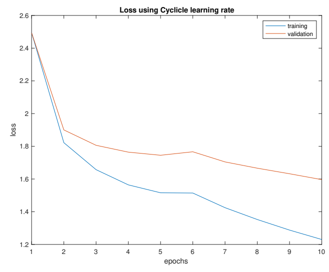
As we see from Figure 1, after training for sufficient number of epochs, the training loss is reduced to sufficient small, e.g. epoch 200 gives training loss . Therefore, this indicates the gradient computations and mini-batch gradient descent algorithm are okay.

2 Cyclical Learning Rate

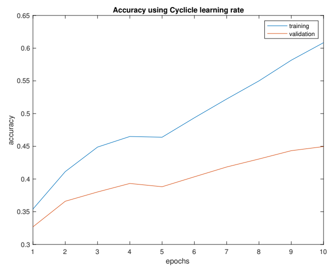
In this section, we present the result of using cyclical learning rate as given in the instruction. To verify the cyclical learning rate is correctly implemented, I did the following test. The results are given in Figures 2 and3.



(a) Cost plot

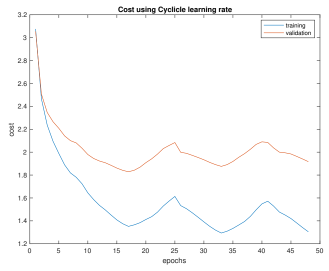


(b) Loss plot

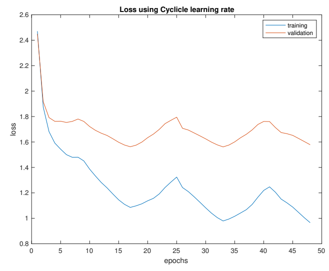


(c) Accuracy plot

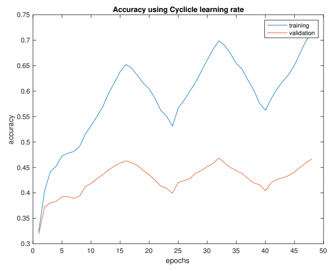
Figure 2: **Training curves (cost, loss, accuracy) for one cycle of training.** Fix the hyper-parameters setting to `eta_min=1e-5`, `eta_max=1e-1`, `lambda=0.01` and `n_s=500`. Furthermore, let the batch size be 100 and one batch of the training data is used. In this case, one cycle corresponds to 10 epochs. In the end of training, a test accuracy of `45.65%` is achieved.



(a) Cost plot



(b) Loss plot



(c) Accuracy plot

Figure 3: **Training curves (cost, loss, accuracy) for three cycles of training.** Fix the hyper-parameters setting to `eta_min=1e-5`, `eta_max=1e-1`, `lambda=0.01` and `n_s=800`. Furthermore, let the batch size be 100 and one batch of the training data is used. In this case, three cycle corresponds to 48 epochs. In the end of training, a test accuracy of `46.68%` is achieved.

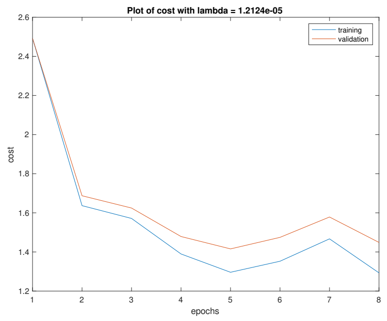
As seen from Figures 2 and 3, the results are identical to those given in the instruction. By implementing cyclical learning rate, the performance is better than the results in assignment 1. Therefore, the use of cyclical function provides substantial improvements. Additionally, observe the curves, we can see there are obvious peaks, whose number equals the number of cycles. They evolve in the zig-zag form and gradually achieves better performance as update steps increase. As pointed out in [1], the cyclic nature allows dropping the learning rates after some cycles and reduces the guesswork in setting the learning rates.

3 Coarse-to-Fine Random Search

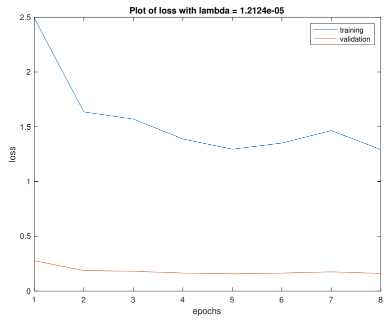
In this section, the results of coarse-to-fine random search. To do so, a random initialization is implemented and then the performance is evaluated. In this example, 5 training batches and use all for training except for 5000 samples for validation. Furthermore, 2 cycles are used in the coarse search. The range of `lambda` is from `1e-5` to `1e-1`. A uniform sampling is implemented on a log scale and `10` different values. The best 3 networks are as listed in Table 2 and their performance are plotted in Figures 4, 5 and 6.

`lambda`	5.9945e-5	1.2124e-5	1.2821e-4
Validation Accuracy	51.62%	52.46%	51.98%
Train Accuracy	58.98%	58.66%	59.05%

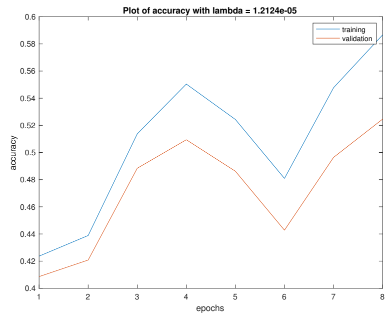
Table 2: **Performance evaluation of coarse search.**



(a) Cost plot

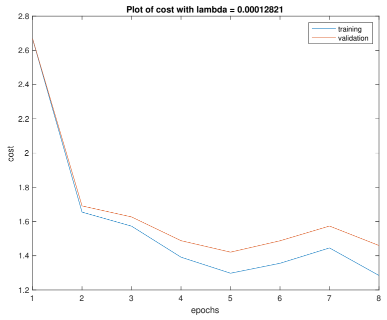


(b) Loss plot

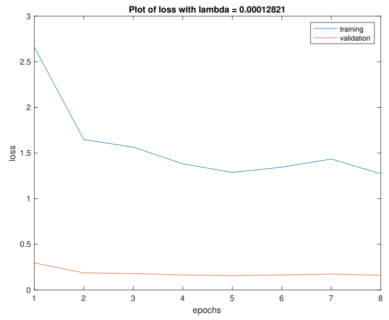


(c) Accuracy plot

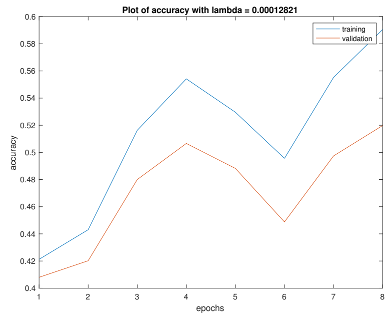
Figure 4: **Training curves (cost, loss, accuracy) for two cycles of training.** The hyper-parameters are `eta_min=1e-5`, `eta_max=1e-1`, `lambda=5.9945e-5` and `n_s=500`. In the end of training, a validation accuracy of `51.62%` is achieved.



(a) Cost plot



(b) Loss plot



(c) Accuracy plot

Figure 5: **Training curves (cost, loss, accuracy) for two cycles of training.** The hyper-parameters are `eta_min=1e-5`, `eta_max=1e-1`, `lambda=1.2124e-5` and `n_s=500`. In the end of training, a validation accuracy of `52.46%` is achieved.

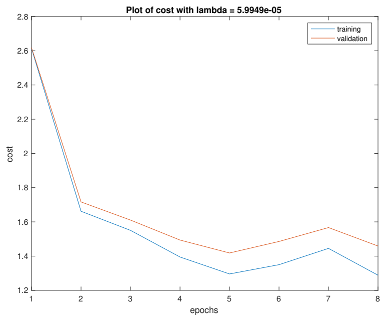
4 Fine Search

After we have the result in coarse search, I narrow down the range of values for `lambda` to `eta_max=1e-3.5` and `eta_min=1e-5`. A uniform sampling is implemented on a log scale and `5` different values. Three cycles are used in the training. Moreover, this time, all the 5 batches are used for training, except for 1000 examples in a validations set. The best 3 networks are as listed in Table 3.

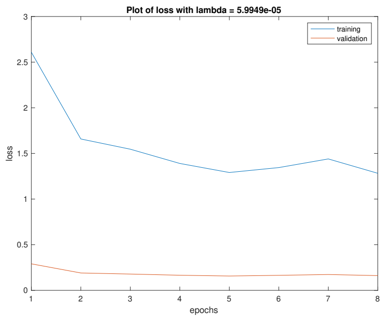
`lambda`	1.0749e-5	1.3396e-4	1.9573e-5
Validation Accuracy	53.40%	52.70%	52.40%
Train Accuracy	61.07%	60.03%	60.56%

Table 3: **Performance evaluation of fine search.**

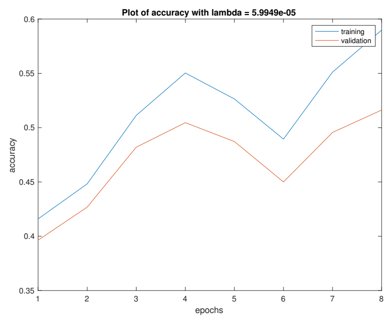
Observe the best 3 results, we can see the best neural network gives the validation accuracy 53.40%.



(a) Cost plot



(b) Loss plot



(c) Accuracy plot

Figure 6: **Training curves (cost, loss, accuracy) for two cycles of training.** The hyper-parameters are `eta_min=1e-5`, `eta_max=1e-1`, `lambda=1.2821e-4` and `n_s=500`. In the end of training, a validation accuracy of `51.98%` is achieved.

References

[1] Smith, L. N. (2015). Cyclical learning rates for training neural networks. *arXiv:1506.01186 [cs.CV].*