Deep Learning DD2424 - Assignment 1 Bonus

Anton Stagge

March 2019

1 Optimizations

1.1 Shuffle

I first created a model with lambda=0.01, n_epochs=40, n_batch=100 and eta=0.01 without shuffling and got an accuracy of 36.5 percent. The new model that implements the shuffling before every new epoch also had an accuracy of 36.5 percent. The loss graphs are show in figures 5 and 6. As one can see there is no difference. This might be because the number of epochs are too low to be able to see a difference.

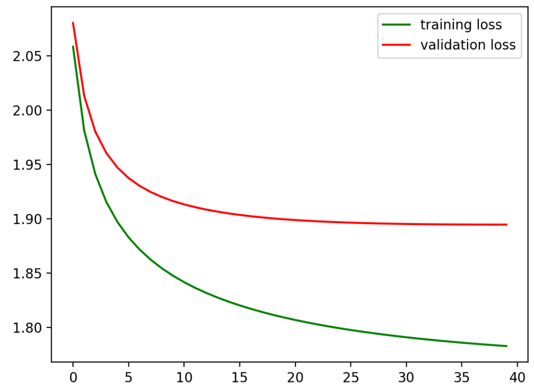

Figure 1: The graph of the training and validation loss computed after every epoch. This model had no shuffling of input data.

1.2 Xavier initialization

I trained the network with the hyper parameters lambda = 0, n_batch = 100, eta = 0.01 n_epochs = 40. First without Xavier initialization, with sigma 0.01

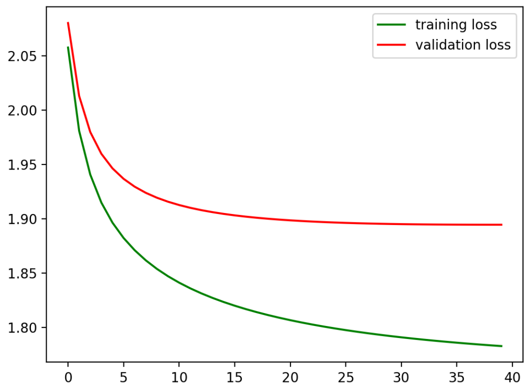

Figure 2: The graph of the training and validation loss computed after every epoch. This model had input shuffling.

as before. This gave a model with accuracy 0.3646. I then trained a model with Xavier initialization $\sigma = \frac{1}{\sqrt{3072}} = 0.018042$ and got a model with accuracy 0.3639. There was not any visible difference in the loss graphs and the weights looked pretty much the same too. The same goes for the value of the gradients as shown in figures 3 and 4. You can not see any big difference. Therefore it is hard to tell if there was any stabilization. The loss in accuracy is very small and I would believe it's due to bad luck in initialization.

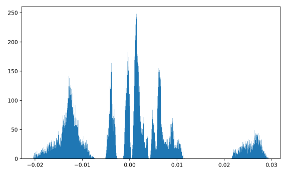

Figure 3: A histogram of the values of the gradients of W computed after model training on the whole training data set. This is with $\sigma = 0.01$.

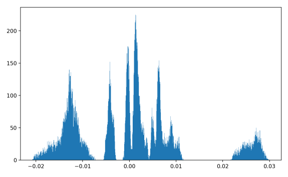

Figure 4: A histogram of the values of the gradients of W computed after model training on the whole training data set. This is with Xavier initialization.

1.3 Decaying learning rate

So far I had not tried anything with the hyper parameters other than σ. Therefore I decided to try implementing a decaying learning rate. A model trained with parameters lambda = 0, n_batch = 100, eta = 0.02 n_epochs = 40 and with an decay of 0.9 got an accuracy of 0.380900, which is the best improvement so far.

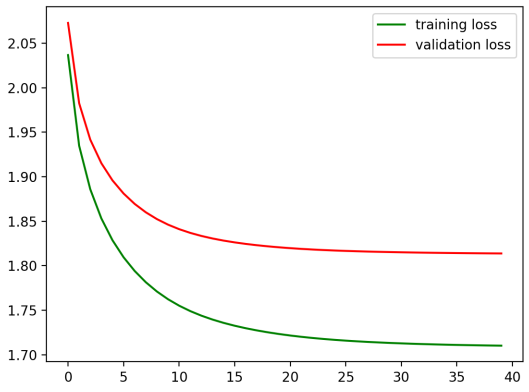

Figure 5: The graph of the training and validation loss computed after every epoch. This model had decaying learning rate with initial eta=0.02 and decay 0.9.

1.4 Grid search

As changing eta had such a big improvement in the decaying learning rate method, I decided to find better values for the other hyper parameters as well. When trying all combinations of:

- lambda: $0, 0.001, 0.0001$

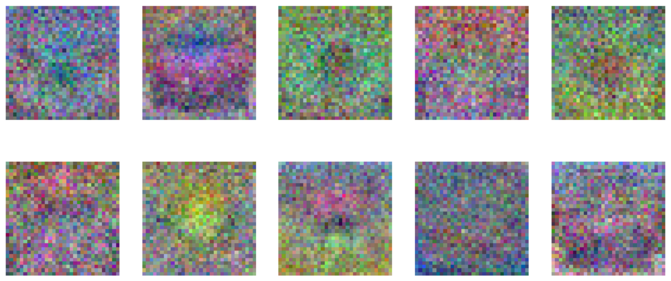

Figure 6: The weights visualized from the model with decaying learning rate.

- n_batch: $64, 100, 128$
- eta: $0.01, 0.02, 0.03$

The winning combination was: lambda: 0.0001, n_batch: 64 and eta: 0.02

This final model had an accuracy of 0.3865 which is the best one I achieved. The validation and traing loss can be seen in Figure 7 and the visulized weigth in Figure 8.

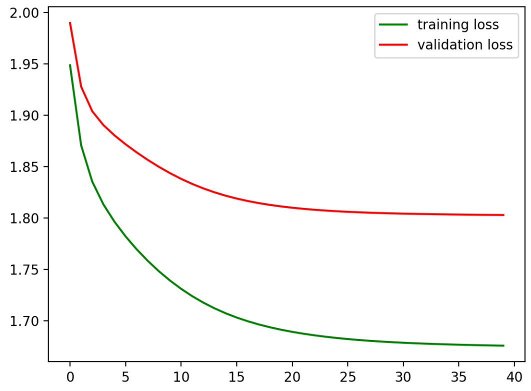

Figure 7: The graph of the training and validation loss computed after every epoch. This is the best model I achieved.

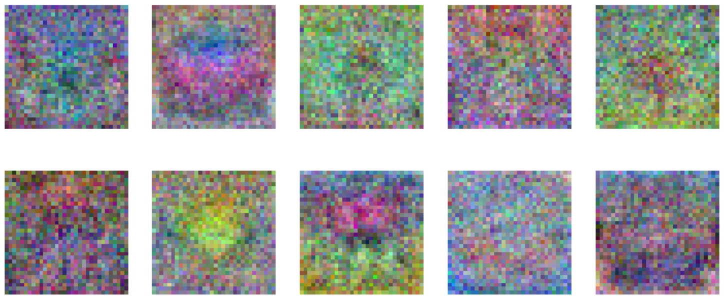
Figure 8: The weights visualized from the model with decaying learning rate. This is of the best model I achieved.

2 Hinge loss and L_2 regularization - SVM

For the model with SVM loss instead of cross entropy i used the same hyper-parameter values as for the final model with cross-entropy. They were: lambda: 0.0001, n_batch: 64 and eta: 0.02.

This yielded a SVM model with accuracy 0.3601, which is slightly worse. This could be due to that maybe the optimal hyper parameters for cross-entropy loss are not optimal for SVM loss. The last value of the loss function was also much higher than for the cross-entropy as one can see in Figure 9. Although, the model works fine!

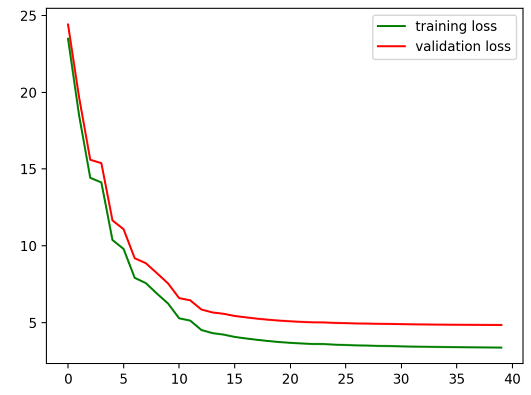

Figure 9: The graph of the training and validation loss computed after every epoch. This model was computed using SVM loss function instead of cross entropy. Hyper parameters were: lambda: 0.0001, n_batch: 64 and eta: 0.02

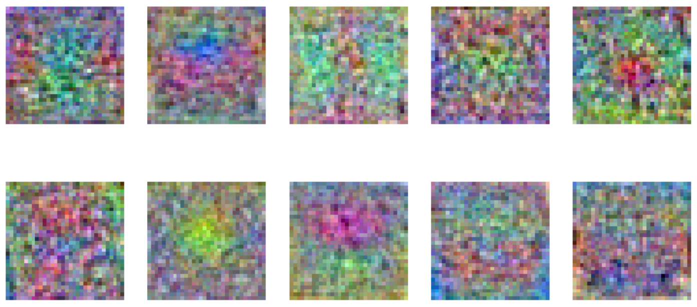

Figure 10: The weights visualized from the model with decaying learning rate. This model was computed using SVM loss function instead of cross entropy.