DD2424: Assignment 2

Sam Shahriari

April 18, 2024

1 Gradient

To check that my analytical calculation of the gradient was correct, I compared it to a numerical estimation for the same points and features. The numerical estimation was calculated by the given function`ComputeGradsNumSlow()` which uses centered difference formula. The batches tested were all possible combinations of featureSize $= 20$, batchSize $\in \{1, 10, 100\}$ and $\lambda \in \{0, .1, 1\}$. None of the absolute or relative errors were above 10^{-6} and therefore I draw the conclusion that my implementation is correct. More detailed results of the testing can be found in Appendix A.

Without any regularization, it is very easy to overfit the model. This can be seen in Figure 1.

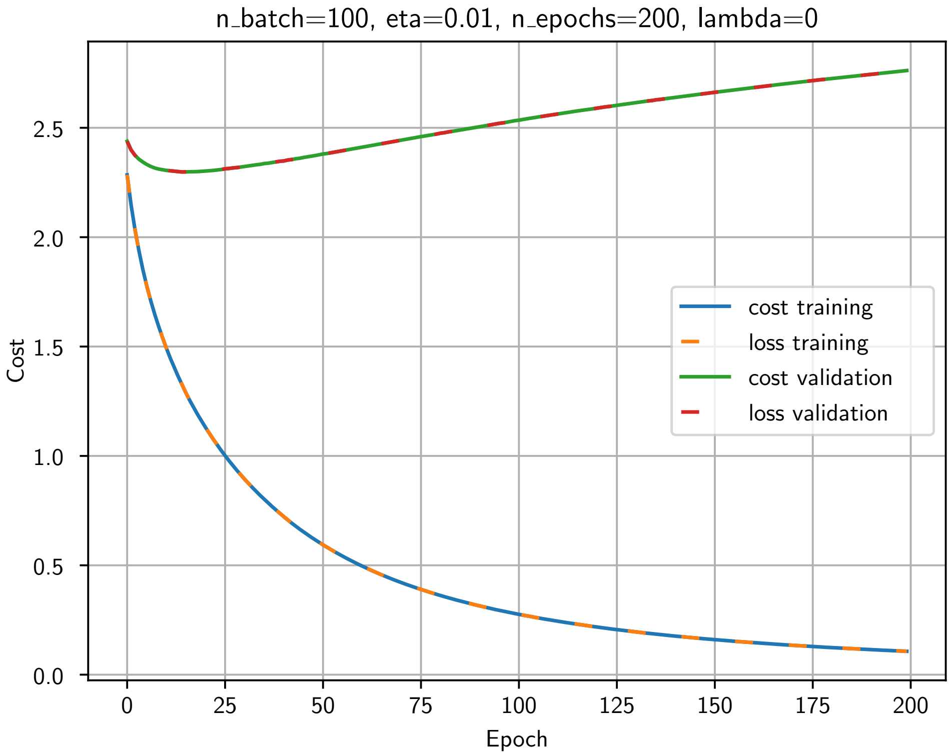

Figure 1: Overfitting the model

2 Cyclical learning

A cyclical learning scheme means that we continuously change one of the hyperparameters from a min value to a max value and then back to a min value. Here we used the cyclical approach on the learning rate which changed between $\eta_{min} = 10^{-5}$ and $\eta_{max} = 10^{-1}$. The effect of varying the learning rate can be seen in Figure 2 and even more clearly in Figure 3.

When examining the graph it can be seen that the y-values mostly goes in the right direction but in an area close the highest learning rate they get worse. This is probably because when using a high lambda, the step might be too big so the minima is jumped over. This could be a good attribute if the loss function contains saddle points or local minimas as they then hopefully will be skipped.

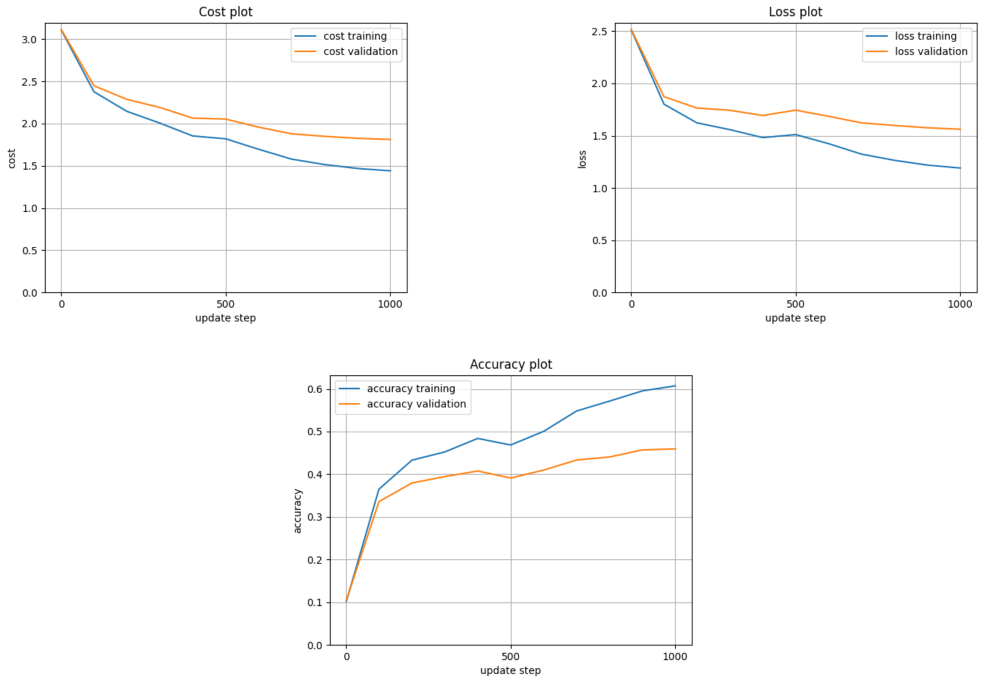

Figure 2: Plots for one cycle with $n_s = 500$

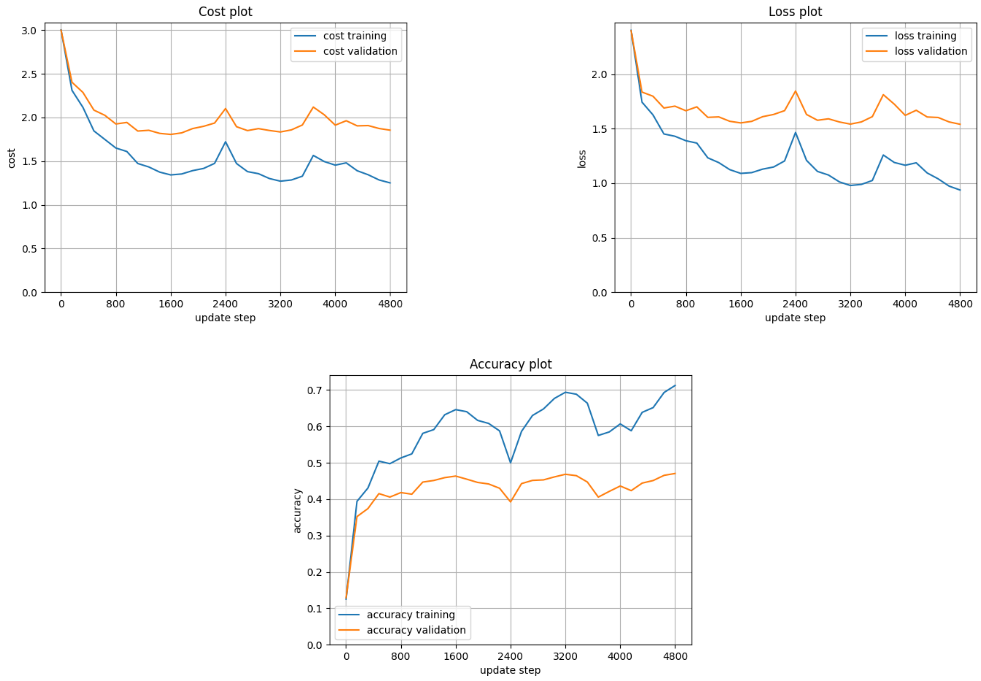

Figure 3: Plots for three cycles with $n_s = 800$

3 Search for lambda

To search for a good lambda value, the training set was extended. It now contained all training batches except for 5000 images that were used as the validation set.

The search was a coarse search with a wide possible range of lambda. 8 random lambda values where chosen in the $\log_{10}$ range from -1 to -5. Other parameter settings were cycles$= 2$, batch size $= 100$, $n_s = 900$, $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$. The three best performing lambdas were:

- Accuracy 0.532 lambda 7.807727344534329e-05 log lambda -4.107475361175347
- Accuracy 0.5296 lambda 5.066346576621312e-05 log lambda -4.295305104485099
- Accuracy 0.5294 lambda 7.193692167309888e-05 log lambda -4.143048150462331

The best lambda was then used for a finer search. Now 20 values were generated in the log range from -4.107 ± 1. The other hyperparameters were the same as in the coarse search. The three best performing lambdas were:

- Accuracy 0.531 lambda 9.867193573189926e-06 log lambda -5.005806351787182
- Accuracy 0.529 lambda 1.6403528938636582e-05 log lambda -4.785062710872495
- Accuracy 0.5272 lambda 2.3557420875260744e-05 log lambda -4.627872258917016

Lastly, the best lambda $= 9.867e{-}06$ was used to train the model. When testing the model on the test data an accuracy of 51.37% was achieved. A plot of the loss function can be seen in Figure 4 and all the lambdas and their respective accuracy can be seen in Appendix B.

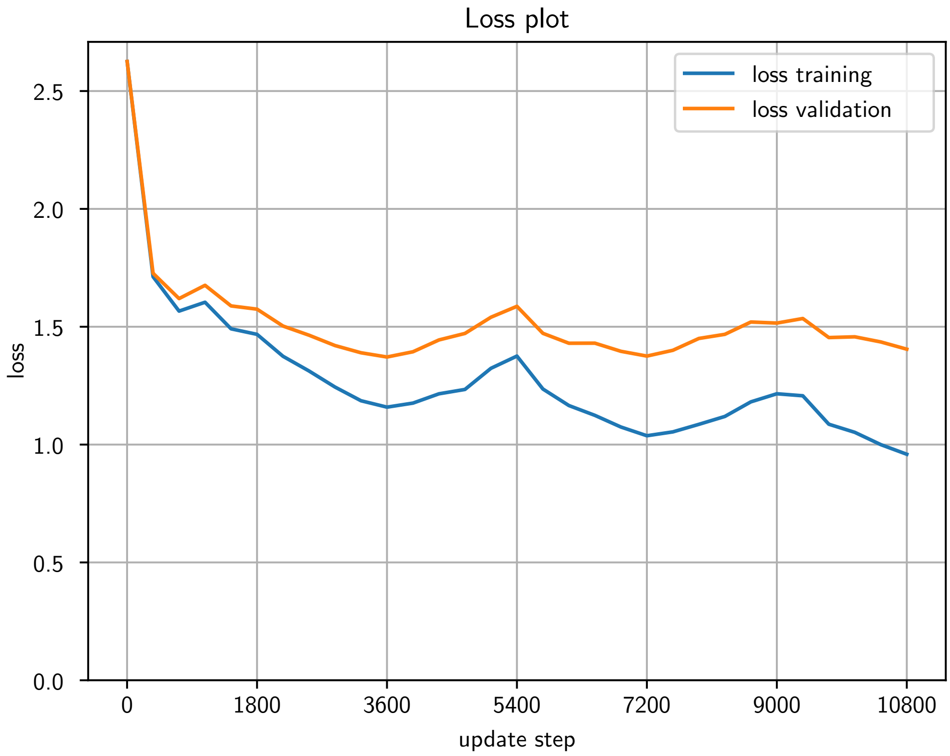

Figure 4: Plot of the loss function for the training and validation set with the best lambda.

A Gradients

```
num features: 20 , batch size: 1 , lambda:  0
SLOW W abs wrong 0 biggest error 2.4892828250078214e-10
SLOW W rel wrong 0 biggest error 1.4533802739282618e-08
SLOW b abs wrong 0 biggest error 6.958188192296433e-11
SLOW b rel wrong 0 biggest error 3.4129318763622067e-10
------
num features: 20 , batch size: 1 , lambda:  0.1
SLOW W abs wrong 0 biggest error 4.316231330681042e-10
SLOW W rel wrong 0 biggest error 2.7179142571849796e-09
SLOW b abs wrong 0 biggest error 6.958188192296433e-11
SLOW b rel wrong 0 biggest error 3.4129318763622067e-10
------
num features: 20 , batch size: 1 , lambda:  1
SLOW W abs wrong 0 biggest error 1.4254555591453055e-09
SLOW W rel wrong 0 biggest error 9.170683667909907e-10
SLOW b abs wrong 0 biggest error 6.415323966502129e-10
SLOW b rel wrong 0 biggest error 3.1466616103752504e-09
------
num features: 20 , batch size: 10 , lambda:  0
SLOW W abs wrong 0 biggest error 4.117283293816887e-10
SLOW W rel wrong 0 biggest error 2.4653246494206414e-08
SLOW b abs wrong 0 biggest error 3.1131446032173216e-10
SLOW b rel wrong 0 biggest error 1.4792496788051695e-09
------
num features: 20 , batch size: 10 , lambda:  0.1
SLOW W abs wrong 0 biggest error 5.233947858451771e-10
SLOW W rel wrong 0 biggest error 3.4908850003620592e-09
SLOW b abs wrong 0 biggest error 3.1131446032173216e-10
SLOW b rel wrong 0 biggest error 1.4792496788051695e-09
------
num features: 20 , batch size: 10 , lambda:  1
SLOW W abs wrong 0 biggest error 1.805463070714275e-09
SLOW W rel wrong 0 biggest error 1.1683430450338027e-09
SLOW b abs wrong 0 biggest error 8.080397739806955e-10
SLOW b rel wrong 0 biggest error 3.83950226689584e-09
------
num features: 20 , batch size: 100 , lambda:  0
SLOW W abs wrong 0 biggest error 5.002005569725715e-10
SLOW W rel wrong 0 biggest error 3.516490864551905e-08
```

```
SLOW b abs wrong 0 biggest error 2.42636927910711e-10
SLOW b rel wrong 0 biggest error 1.9957499952210056e-09
------
num features: 20 , batch size: 100 , lambda:  0.1
SLOW W abs wrong 0 biggest error 5.088013725618179e-10
SLOW W rel wrong 0 biggest error 3.3926608475673607e-09
SLOW b abs wrong 0 biggest error 2.42636927910711e-10
SLOW b rel wrong 0 biggest error 1.9957499952210056e-09
------
num features: 20 , batch size: 100 , lambda:  1
SLOW W abs wrong 0 biggest error 1.6478069464476164e-09
SLOW W rel wrong 0 biggest error 1.0661789264572265e-09
SLOW b abs wrong 0 biggest error 7.109335689592378e-10
SLOW b rel wrong 0 biggest error 5.8476081075959645e-09
```

B Lambda Random Search

```
Accuracy 0.532 lambda 7.807727344534329e-05 log lambda -4.107475361175347
Accuracy 0.5296 lambda 5.066346576621312e-05 log lambda -4.295305104485099
Accuracy 0.5294 lambda 7.193692167309888e-05 log lambda -4.143048150462331
Accuracy 0.5236 lambda 0.00014498005602996882 log lambda -3.8386917367433813
Accuracy 0.5224 lambda 0.00010803538625357892 log lambda -3.966433971017319
Accuracy 0.5224 lambda 3.197602887656392e-05 log lambda -4.495175472549843
Accuracy 0.522 lambda 0.0016228962075828747 log lambda -2.7897092546133395
Accuracy 0.516 lambda 2.931955461229069e-05 log lambda -4.532842631264662
-------
Accuracy 0.531 lambda 9.867193573189926e-06 log lambda -5.005806351787182
Accuracy 0.529 lambda 1.6403528938636582e-05 log lambda -4.785062710872495
Accuracy 0.5272 lambda 2.3557420875260744e-05 log lambda -4.627872258917016
Accuracy 0.5262 lambda 0.00010832213245853509 log lambda -3.965282798937192
Accuracy 0.5256 lambda 9.069945553883424e-06 log lambda -5.042395319965149
Accuracy 0.5244 lambda 7.7774484176332e-05 log lambda -4.109162860580757
Accuracy 0.5234 lambda 1.089283904004913e-05 log lambda -4.962858913748786
Accuracy 0.5218 lambda 0.0004735627803922964 log lambda -3.324622438264533
Accuracy 0.5216 lambda 0.000607347932639988 log lambda -3.2165624424743715
Accuracy 0.52 lambda 1.828514484964466e-05 log lambda -4.737901594956661
Accuracy 0.5194 lambda 3.353723742505652e-05 log lambda -4.474472714552444
Accuracy 0.5194 lambda 2.917295194356437e-05 log lambda -4.535019623400148
Accuracy 0.5186 lambda 0.0004622448632289073 log lambda -3.335127906309916
Accuracy 0.5174 lambda 7.063692063734561e-05 log lambda -4.150968241754363
```

```
Accuracy 0.5168 lambda 0.0005695846990260709 log lambda -3.244441685835034
Accuracy 0.5156 lambda 4.6161234167673446e-05 log lambda -4.335722588391209
Accuracy 0.5146 lambda 0.00010063522872220608 log lambda -3.997249962009483
Accuracy 0.5136 lambda 1.2155852830676228e-05 log lambda -4.915214566506879
Accuracy 0.5114 lambda 5.137162709657223e-05 log lambda -4.289276678623103
Accuracy 0.5072 lambda 0.0007257027866074966 log lambda -3.1392412092790263
-------
Accuracy on test data: 0.5137
```