

Deep Learning in Data Science

Assignment 1

Isak Gamnes Sneltvedt

April 4th, 2022

DD2424

Assignment 2

Gradient comparison

For the gradient comparison I checked the absolute differences between the numerical and analytical computed vectors. This method check that the relative difference between the numerical and the analytical is less than a certain threshold (10^{-5} in my case).

The output from the terminal after the comparison is as follows:

```
1 Output:
2     Grads test on W1 returned: True, with nominator: 4.724876676739617
   e-05 and denominator: 47.678748314893035
3     Grads test on W2 returned: True, with nominator: 4.779799141510413
   e-07 and denominator: 2.925975628886696
4     Grads test on b1 returned: True, with nominator:
   2.2119807889667623e-06 and denominator: 1.970798722821977
5     Grads test on b2 returned: True, with nominator: 4.499112193947965
   e-06 and denominator: 0.8364133132592686
6     9.90981693884725e-07
7     1.6335744885643762e-07
8     1.1223778275030734e-06
9     5.3790537795437325e-06
```

This shows both the nominator and denominator. The nominator is calculated from the absolute different between the two gradients. The denominator is calculated from the absolute sum of the two gradients. The reason why I use the denominator is because it will yield the *relative error* rather than just the error. This is important as an error of, for instance, 0.0005 is considered small if we have a gradient of 3. Yet, if we have a gradient of 0.000001, then the error of 0.0005 is considered huge and we are not able to use it. So by using the *relative error*, we find the error in regards to the total gradient. The two gradient checks that I chose arbitrary earlier will now be: $\frac{0.0005}{3}$ which is a very small number, and $\frac{0.0005}{0.000001}$, which is a very large number. Based on this, I believe my gradient check and calculation to be correct.

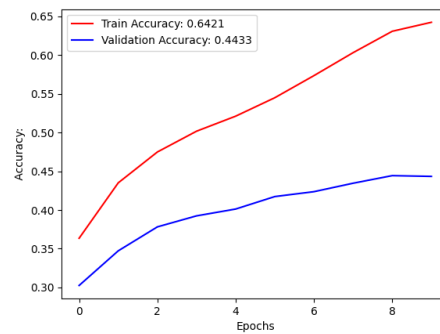


Figure 1: Accuracy with CLR

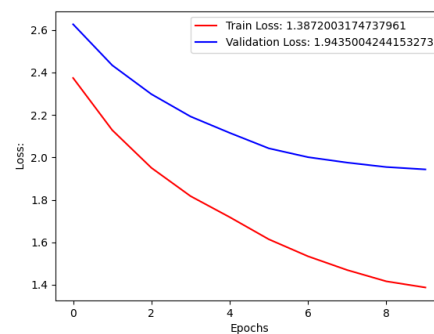


Figure 2: Loss with CLR

CLR

As can be seen on the plots, the model is being a bit over fitted. This can be seen by the accuracy. The training yields 20% higher accuracy compared to the validation set. Also on the loss we are able to see that the loss on the training data is lower than the loss on the validation data. Figure 3 was added to show correct implementation of the CLR.

Coarse lambda search

For the coarse search for a lambda value I used the recommended numbers given in the assignment. I generated a total of 10 random numbers that was used to find a good search area for a lambda value.

```
1 l_min = -5
2 l_max = -1
3 l = l_min + (l_max-l_min)*np.random.rand(10)
```

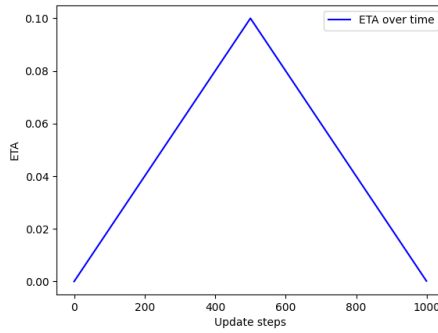


Figure 3: The change of ETA in one cycle

```
4 list_of_lambdas = 10**1
```

`np.random.rand(10)` will generate a list of 10 elements ranging from 0-1. From the expression $(l_{max} - l_{min})$ we get -4 . The extreme cases are when the random generated number is 0 or 1. When the number is 0 we will end up with $\lambda = 10^{-1}$. Whereas if the random generated number is 1 we get that $\lambda = 10^{-5}$. This implies that the initial search range for my lambda values ranges from 10^{-1} to 10^{-5} .

During the lambda search I used the hyper parameters that was presented during the assignment. I.e. `n_s = 800`, `batch size = 100`, `eta_min = 1e-5`, `eta_max = 1e-1` and two cycles (6.5 epochs, rounded to 7 epochs, calculations explained under the final test).

Lambda	Accuracy
0.053832	0.4558
0.000135	0.4782
0.000037	0.4816
0.019669	0.4912
0.000417	0.4946
0.000269	0.4962
0.000659	0.5018
0.000751	0.5032
0.000047	0.5088
0.001170	0.5182

Table 1: Accuracy for each lambda value sorted after accuracy.

From the initial search I found that $\lambda = 0.003422$ gave the best results.

Fine search

For the fine search I ended up running a total of 3 searches. The first one had the lambda values of:

```
1 list_of_lambdas = [0.0005, 0.0075, 0.001, 0.00125, 0.00150, 0.00175]
```

The results from this search was:

Lambda	Accuracy
0.00050	0.5068
0.00175	0.5084
0.00150	0.5096
0.00750	0.5132
0.00125	0.5132
0.00100	0.5170

As the results came out to be the same as previously, I decided to run the fine search once more with a higher resolution closer to the best accuracy.

```
1 list_of_lambdas = [0.0008, 0.00085, 0.0009, 0.00095, 0.001, 0.00105,  
    0.0011, 0.00115, 0.0012]
```

The results from this search was:

Lambda	Accuracy
0.00120	0.5016
0.00115	0.5018
0.00100	0.5034
0.00095	0.5036
0.00110	0.5036
0.00105	0.5042
0.00080	0.5066
0.00090	0.5090
0.00085	0.5130

After performing all of these tests I realized that the accuracies were very similar, as well as changing from each training. This is, among other things, due to the random values in the

weights and biases and I then concluded that the best value for lambda was around 0.00085, as this gave quite good results with a accuracy consistently above 50%

Final test

For the final test the following specifications are set: $n_s = 2 \cdot \frac{n}{n_{batch}}$, and I have to use the entire data set of 50,000 samples, where 1,000 of these will be used for validation. When using a batch size of 100, I find that the number of updates per epoch is: $Updates = \frac{dataset_size}{batch_size} = \frac{49,000}{100} = 490$, which gives $n_s = 980$. This gives a total of 1960 updates for one cycle. For the desired 3 cycles, it gives a total of 5,880 updates. This means that the total number of epochs is $n_{epochs} = \frac{5880}{490} = 12$.

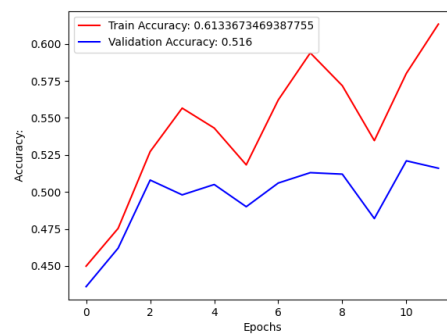


Figure 4: Accuracy in the final test

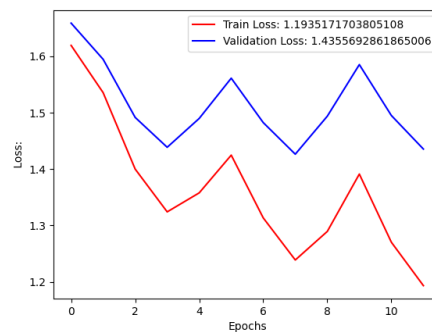


Figure 5: Loss in the final test

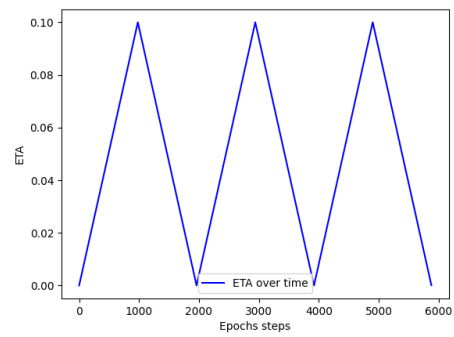


Figure 6: CLR in the final test

The test accuracy came out to be:

```
1 The test accuracy from the final test is: 0.4995 (49.95%)
```