

# DD2424 Deep Learning in Data Science

## Assignment 1

Lingxi Xiong  
lingxi@kth.se

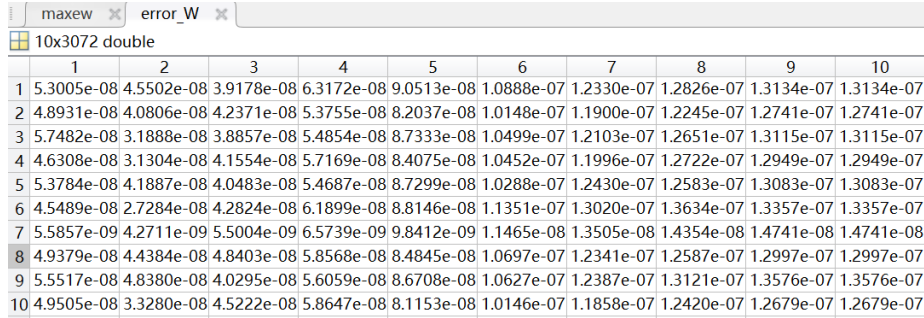
2nd April, 2019

### 1 Exercise 1

In this exercise, the *ComputeDiff* function is used to compute the difference between numerically and analytically computed gradient following the equation (1).

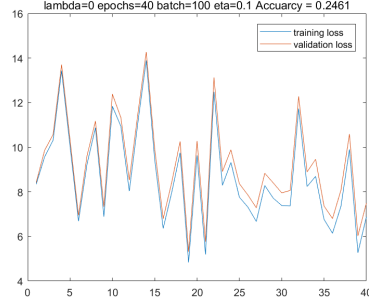
$$err = \frac{|g_a - g_n|}{\max(eps, |g_a| + |g_n|)} \quad (1)$$

The example of error matrix can be seen in Figure 1. By setting the eps to e-5 and using the numerical gradient computation based on the finite difference method. The maximum value in b error vector and W error matrix is 2.3627e-07 for error.

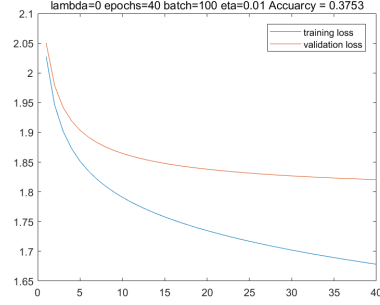


	1	2	3	4	5	6	7	8	9	10
1	5.3005e-08	4.5502e-08	3.9178e-08	6.3172e-08	9.0513e-08	1.0888e-07	1.2330e-07	1.2826e-07	1.3134e-07	1.3134e-07
2	4.8931e-08	4.0806e-08	4.2371e-08	5.3755e-08	8.2037e-08	1.0148e-07	1.1900e-07	1.2245e-07	1.2741e-07	1.2741e-07
3	5.7482e-08	3.1888e-08	3.8857e-08	5.4854e-08	8.7333e-08	1.0499e-07	1.2103e-07	1.2651e-07	1.3115e-07	1.3115e-07
4	4.6308e-08	3.1304e-08	4.1554e-08	5.7169e-08	8.4075e-08	1.0452e-07	1.1996e-07	1.2722e-07	1.2949e-07	1.2949e-07
5	5.3784e-08	4.1887e-08	4.0483e-08	5.4687e-08	8.7299e-08	1.0288e-07	1.2430e-07	1.2583e-07	1.3083e-07	1.3083e-07
6	4.5489e-08	2.7284e-08	4.2824e-08	6.1899e-08	8.8146e-08	1.1351e-07	1.3020e-07	1.3634e-07	1.3357e-07	1.3357e-07
7	5.5857e-09	4.2711e-09	5.5004e-09	6.5739e-09	9.8412e-09	1.1465e-08	1.3505e-08	1.4354e-08	1.4741e-08	1.4741e-08
8	4.9379e-08	4.4384e-08	4.8403e-08	5.8568e-08	8.4845e-08	1.0697e-07	1.2341e-07	1.2587e-07	1.2997e-07	1.2997e-07
9	5.5517e-08	4.8380e-08	4.0295e-08	5.6059e-08	8.6708e-08	1.0627e-07	1.2387e-07	1.3121e-07	1.3576e-07	1.3576e-07
10	4.9505e-08	3.3280e-08	4.5222e-08	5.8647e-08	8.1153e-08	1.0146e-07	1.1858e-07	1.2420e-07	1.2679e-07	1.2679e-07

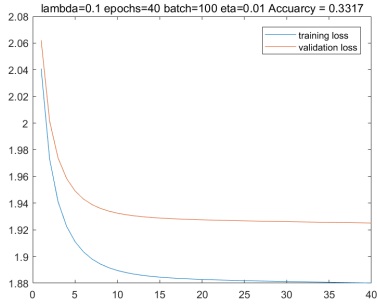
Figure 1: example of difference between W gradient matrix computed analytically and numerically one



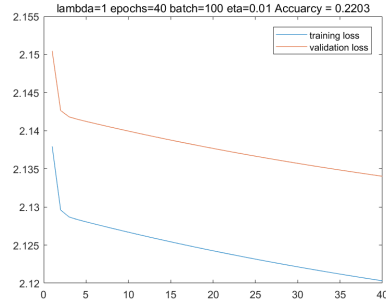
(a)  $\lambda=0, \eta=0.1$



(b)  $\lambda=0, \eta=0.01$



(c)  $\lambda=0.1, \eta=0.01$



(d)  $\lambda=1, \eta=0.01$

Figure 2: Total loss on the training data and validation data (#epochs=40, batch size=100)

Graphs of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm is shown in Figure 2.

Images representing the learned weight matrix after the completion of training is shown in Figure 3.

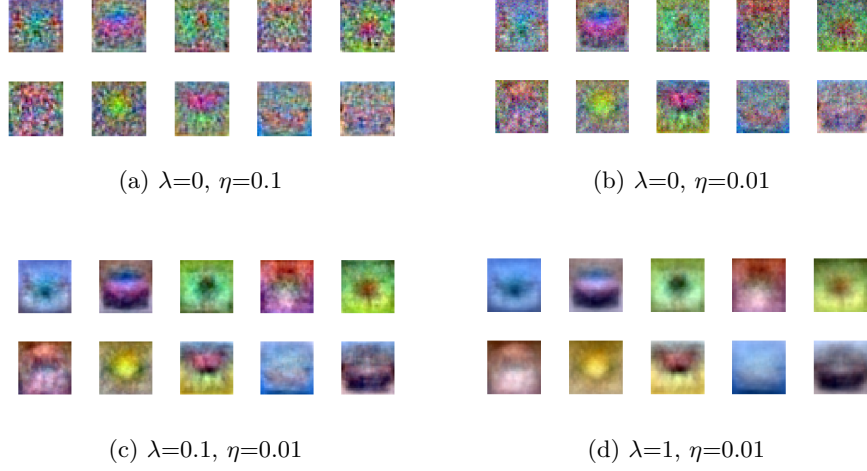


Figure 3: The learnt weight matrix with different settings(#epochs=40, batch size=100)

Table 1: Accuracy on test data after training under different configurations

lambda	eta	epochs	batch	Accuracy
0	0.1	40	100	0.2461
<b>0</b>	<b>0.01</b>	<b>40</b>	<b>100</b>	<b>0.3753</b>
0.1	0.01	40	100	0.3317
1	0.01	40	100	0.2203

From Table 1 we can see the setting of low learning rate and no regularization achieve the highest accuracy. Generally, conducting regularization in the right range may increase the performance of the model, since it will decrease the model complexity, prevent it from overfitting and facilitate the generalization. Whereas in this case, the regularization may bring some overkill to this very simple model. Although the visualized weight matrix with regularization reveals more explicit leaned representation of the data (e.g. we can see more clearly that the second figure resembles a car). Also, too large learning rate will end up having jumping loss and eventually lead to poor performance. On the other hand, too small learning rate will cost too much time and resource. Therefore it is very important to choose the correct learning rate.