KTH Royal Institute of Technology

DD2424 Deep Learning in Data Science

Assignment Report 1

One layer network with multiple outputs

Written by

Yutong Jiang

yutongj@kth.se

Date: Mar 29, 2022

1 introduction

In this assignment, I will train and test a one layer network with multiple outputs to classify images from the CIFAR-10 dataset by using mini-batch gradient descent applied to the cost function.

2 Principles of gradients computation

The basic structure of one layer network in assignment 1 could be concluded as linear scoring function + SoftMax + cross-entropy loss + Regularization.

The corresponding cost function could be shown as

$$J = l + \lambda r \tag{1}$$

where $l = -y^T log(p)$. By taking derivative, we could find that the gradients for each sample could be calculated as follows.

$$g = \frac{\partial J}{\partial b} = -y^T diag(p)^{-1}(diag(p) - pp^T) = -(y-p)^T \tag{2}$$

$$\frac{\partial J}{\partial W} = g^T x^T + 2\lambda W \tag{3}$$

After getting the gradients, we could update weight and bias iteratively.

In order to test the gradient is calculated correctly, the results is compared to the numerical results provided by $ComputeGradsNumSlow$ as it is more accurate. The relative error could be computed as

$$relative - error = \frac{|g_a - g_n|}{max(\epsilon, |g_a| + |g_n|)} \tag{4}$$

By setting different λ and batch size, the calculated relative error could be shown as follows.

When $\lambda = 0$ and size of mini-batch = 100, $eps_b = 2.3731e-11$ and $eps_W = 9.9196e-14$

When $\lambda = 0$ and size of mini-batch = 10000, $eps_b = 6.7657e-11$ and $eps_W = 5.9461e-13$

When $\lambda = 0.1$ and size of mini-batch = 10000, $eps_b = 1.4899e-10$ and $eps_W = 2.6924e-12$

When $\lambda = 0.1$ and size of mini-batch = 100, $eps_b = 3.7593e-11$ and $eps_W = 1.0086e-12$

Hence, based on the comparison of analytical and numerical results shown as above, the conclusion that gradients are calculated correctly is drawn.

3 Results

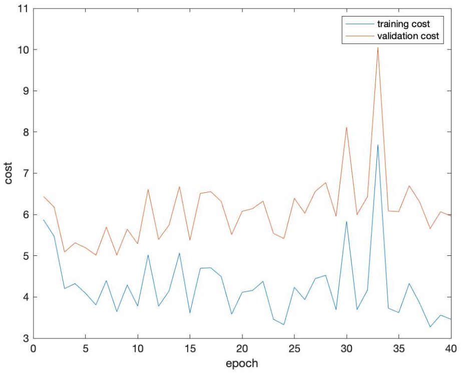

Figure 1: Cost/Loss when lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.1

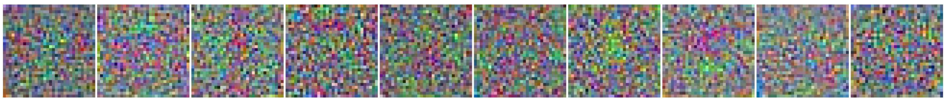

Figure 2: Weight when lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.1

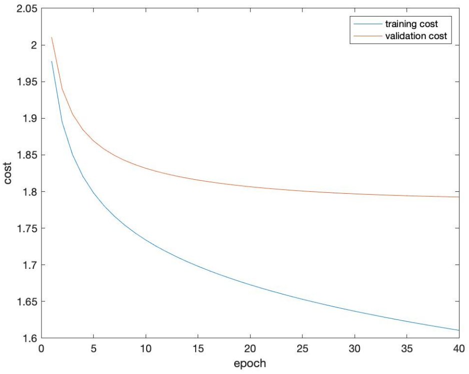

Figure 3: Cost/Loss when lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.001

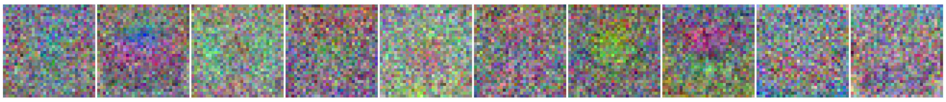

Figure 4: Weight when lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.001

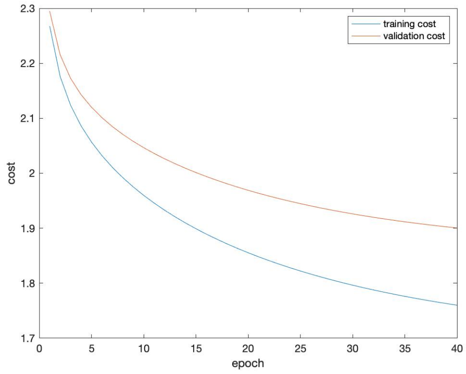

Figure 5: Cost when lambda = 0.1, n-epochs = 40, n-batch = 100, eta = 0.001

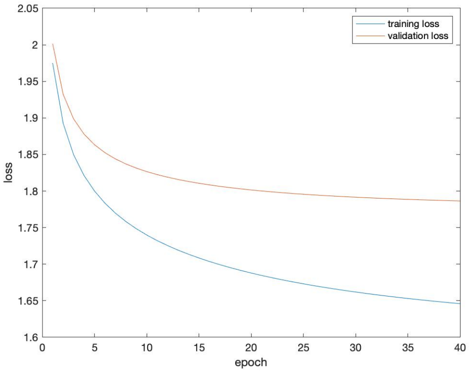

Figure 6: Lost when lambda = 0.1, n-epochs = 40, n-batch = 100, eta = 0.001

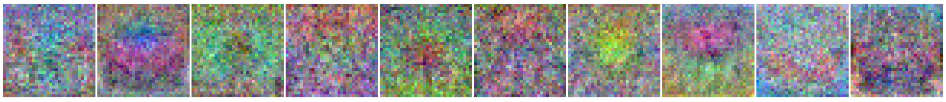

Figure 7: Weight when lambda = 0.1, n-epochs = 40, n-batch = 100, eta = 0.001

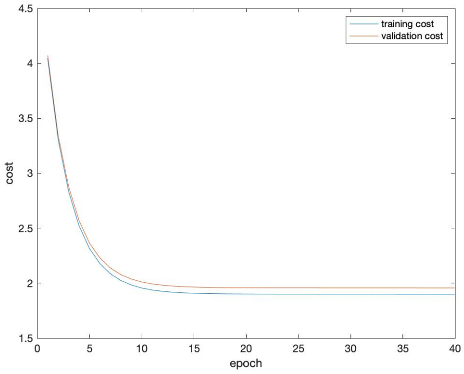

Figure 8: Cost when lambda = 1, n-epochs = 40, n-batch = 100, eta = 0.001

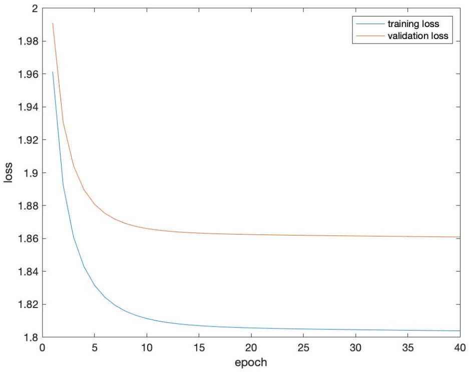

Figure 9: Lost when lambda = 1, n-epochs = 40, n-batch = 100, eta = 0.001

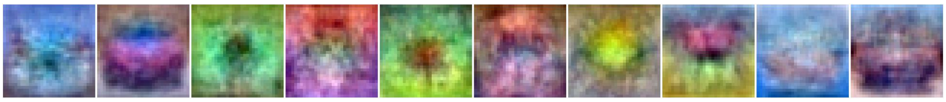

Figure 10: Weight when lambda = 1, n-epochs = 40, n-batch = 100, eta = 0.001

4 conclusion

The final accuracy of lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.1 is 0.2848

The final accuracy of lambda = 0, n-epochs = 40, n-batch = 100, eta = 0.001 is 0.3893

The final accuracy of lambda = 0.1, n-epochs = 40, n-batch = 100, eta = 0.001 is 0.3918

The final accuracy of lambda = 1, n-epochs = 40, n-batch = 100, eta = 0.001 is 0.3755

From the statistics, we could draw the conclusion that the learning rate should not be set too large as it will make the curve fluctuate which can not reach the local minimum point.

Besides, adding regularization could increase the accuracy. However, large λ could also increase bias which decreases the accuracy.