DD2424 Assignment 1 basic

Jacob Malmberg, 198508060558

March 2019

1 Analytical gradient check

The analytical gradients were checked against numerical gradients computed by ComputeGradsNumSlow.m. This was done for a batch containing the first 100 datapoints of data_batch_1.mat. All 3072 dimensions were included. The relative error was computed following the guidelines laid out on page 7 of Assignment 1:

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

The results are summarized in 1. According to `http://cs231n.github.io/neural-networks-3/#gradcheck` these are good values which makes me conclude that my gradient computations are bug free. Adding to this, I implemented backward/forward pass using efficient matrix computations as specified in the lecture ppt's.

Configuration	λ	Max W relative error	Max B relative error
100 datapts	0	5.17e-6	6.12e-8
100 datapts	0.1	9.11 e-6	2.31e-7

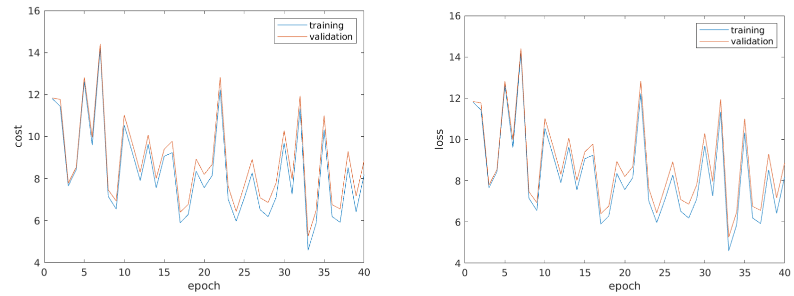
Table 1: Table showing summary of relative errors

2 Results

The following sections contain the results. The last subsection contains a comment on the effect of regularization and the importance of the correct learning rate.

2.1 lambda = 0, n_epochs = 40, n_batch = 100, eta = .1

The accuracy was 27.07%. Loss and cost for this configuration can be found in fig 1. Weights visualization can be found in fig 2.



(a) Cost plot (b) Loss plot

Figure 1: Loss and cost plots for lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1

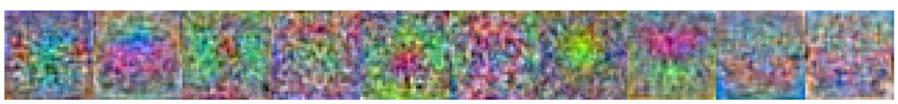
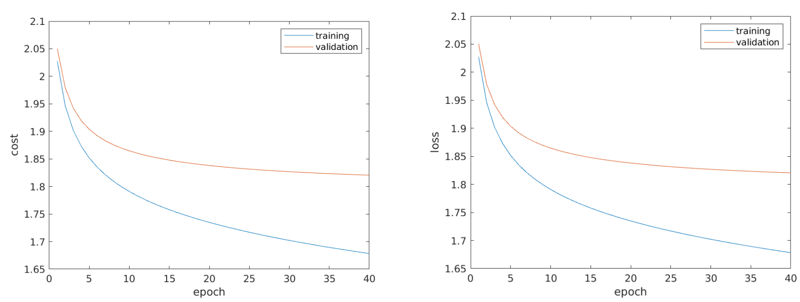


Figure 2: Learnt W matrix for lambda = 0, n_epochs = 40, n_batch = 100, eta = .1



(a) Cost plot (b) Loss plot

Figure 3: Loss and cost plots for lambda = 0, n_epochs = 40, n_batch = 100, eta = .01

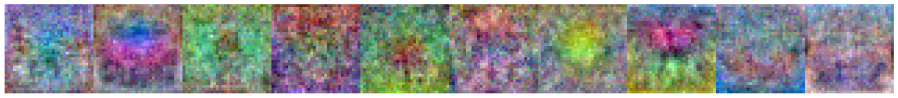
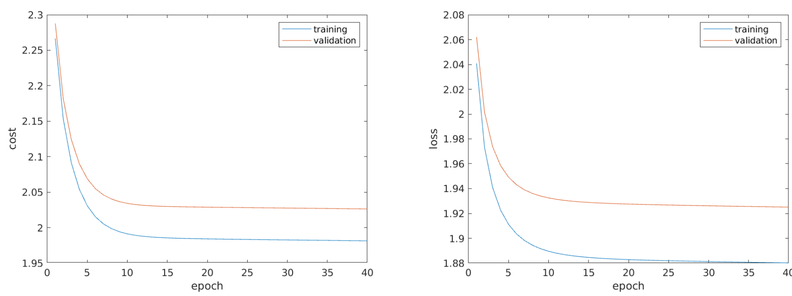


Figure 4: Learnt W matrix for lambda = 0, n_epochs = 40, n_batch = 100, eta = .01



(a) Cost plot

(b) Loss plot

Figure 5: Loss and cost plots for lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.01

2.2 lambda = 0, n_epochs = 40, n_batch = 100, eta = .01

The accuracy was 36.65%. Loss and cost for this configuration can be found in fig 3. Weights visualization can be found in fig 4.

2.3 lambda = 0.1, n_epochs = 40, n_batch = 100, eta = .01

The accuracy was 33.37%. Loss and cost for this configuration can be found in fig 5. Weights visualization can be found in fig 6.

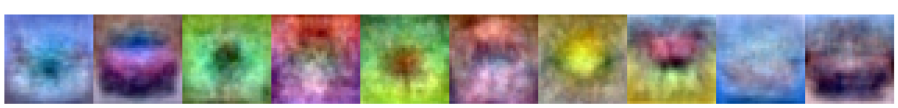
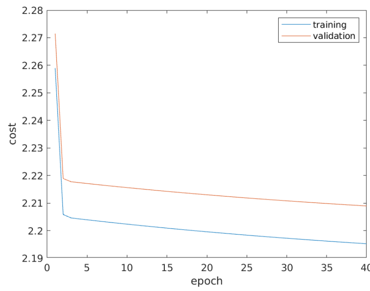
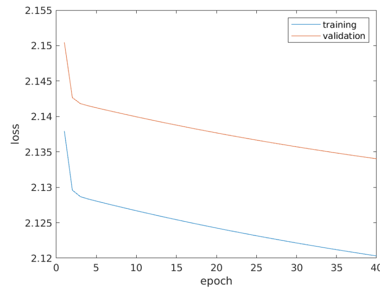


Figure 6: Learnt W matrix for lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.01



(a) Cost plot



(b) Loss plot

Figure 7: Loss and cost plots for lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.01

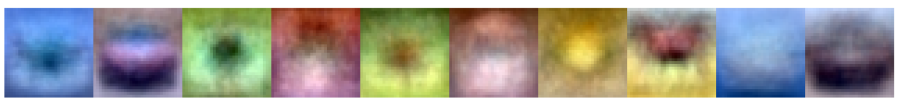


Figure 8: Learnt W matrix for lambda = 1, n_epochs = 40, n_batch = 100, eta = .01

2.4 lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.01

The accuracy was 21.92%. Loss and cost for this configuration can be found in fig 7. Weights visualization can be found in fig 8.

2.5 Effects of regularization and correct learning rate

If the learning rate is too high, we may end up overshooting the minimum point and instead start to climb toward a higher function value when doing gradient descent. I believe this is what is happening in fig. 1. A lower learning rate is slower but should take us toward the minimum. Thus, the learning rate should be well calibrated.

Increasing regularization should prevent overfitting to the training dataset, however a too high lambda may result in poor test performance. In fig. 7 the cost for training and validation set is closer to each other than in plots with lower regularization (the y-axis are different in the plots which makes it a little funny looking). With no regularization we may end up with clear overfitting, shown by training cost and validation loss diverging. This is shown in fig 3.