Henrik Holm henholm@kth.se 2020-04-08

DD2424 Deep Learning in Data Science

Assignment 2 – base version

The toolset used for solving the assignment included default, built-in Python as well as the two external Python libraries numpy (for linear algebra purposes) and matplotlib (for plotting needs). The data set used was the CIFAR-10 collection of labelled images. A random seed of 12345 was used throughout the assignment.

Gradients

For ensuring that the implemented compute_gradients function (which computes gradients analytically) yielded the correct values, a test was devised. The test compares the results of compute_gradients with the results of compute_gradients_num, which calculates the same gradients, however using a numerical approach. The test was conducted using numpy's allclose-function, which compares the values of two equally shaped matrices and returns either True or False depending on whether the compared elements are all close in value or not. With an absolute tolerance of 1e-05, the function returned True when comparing computed gradients for a subset consisting of 10 images from the CIFAR-10 data, meaning that all elements of the gradients were similar. With an absolute tolerance of 1e-06, the function returned False. However, upon closer inspection, one could see that apart from a few exceptions, most values were similar.

A further test was conducted for ensuring the correctness of the gradient calculations. With the regularization term set to 0 and the number of training examples reduced to 100 images, it was investigated whether the network was able to overfit after a sufficient number of epochs. With the number of epochs set to 200, this was very much the case, as the loss was close to 0 by the time training finished. For these reasons, the analytical compute_gradients-function implemented was deemed as having been implemented correctly.

Replicate figures 3 and 4

Replications of the curves for the training and validation loss and cost (from the assignment sheet) are displayed below in Figures 1-2. As one can see when comparing the replications to the original figures, a high degree of similarity is the case.

A main reason behind using a cyclical learning rate, is that it (hopefully) allows for an optimization search which both searches the wider space and "zooms in" on local windows. By analyzing our figures, one can argue that this is the case, since the cost and loss functions do not

monotonously decrease within a cycle. Hence, one can also argue that our implementation of a cyclical learning rate in our mini-batch gradient descent is correct.

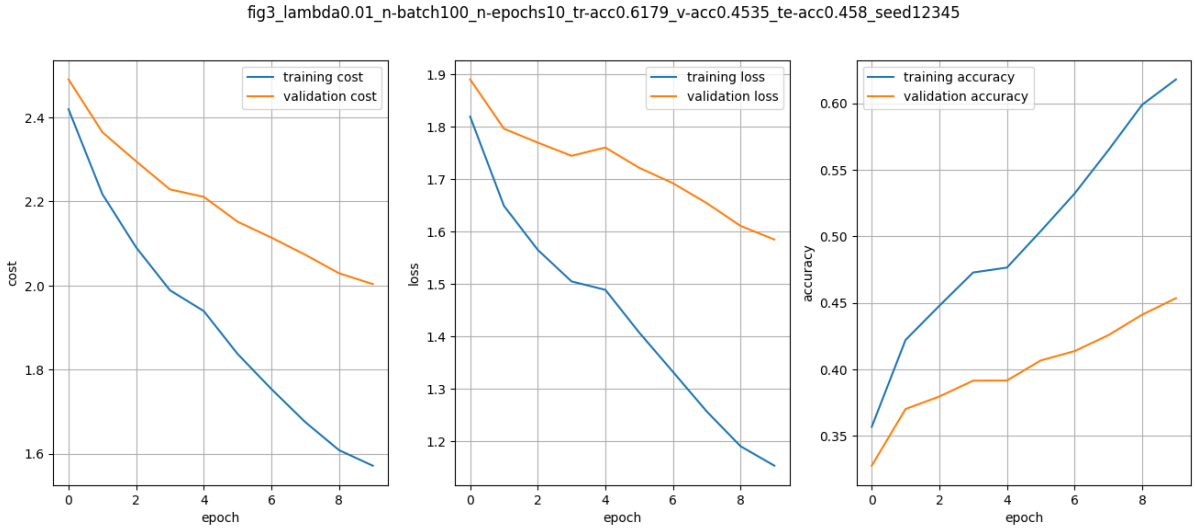

Figure 1: Cost plot, loss plot and accuracy plot respectively, effectively replicating Figure 3 in the assignment sheet. Here, lambda=0.01, n_batch=100, n_epochs=10, n_s=500, eta_min=1e-5 and eta_max=1e-1 were used. The final validation accuracy equaled 45.35% and the final training accuracy equaled 61.79%.

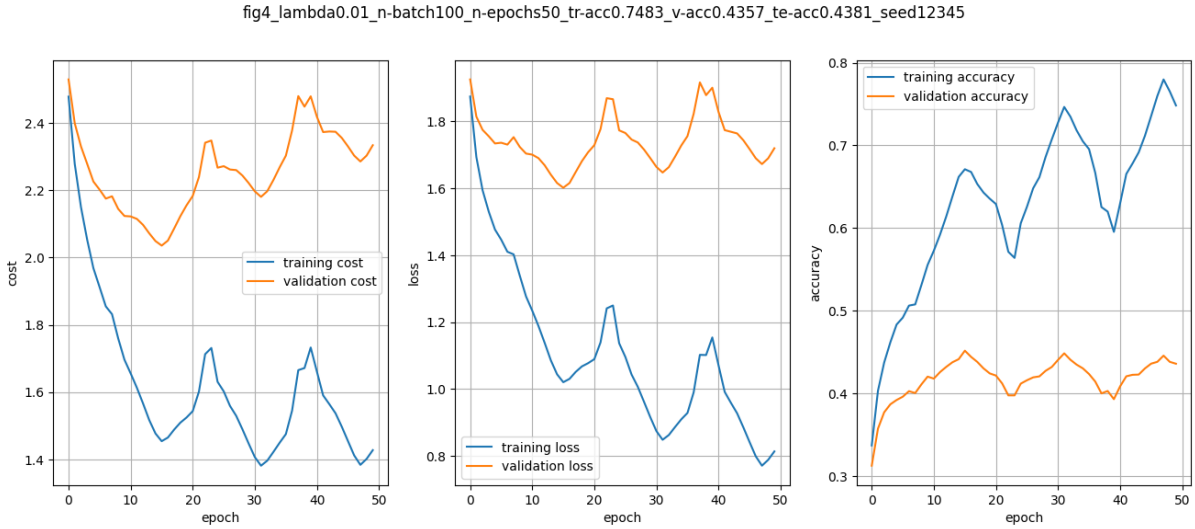

Figure 2: Cost plot, loss plot and accuracy plot respectively, effectively replicating Figure 4 in the assignment sheet. Here, lambda=0.01, n_batch=100, n_epochs=50, n_s=800, eta_min=1e-5 and eta_max=1e-1 were used. The final validation accuracy equaled 43.57% and the final training accuracy equaled 74.83%.

Coarse lambda search

8 values of the regularization term lambda were sampled uniformly from the range [1e-5, 1e-1]. Then, for each of these values, a training session was executed using n_batch=100, eta_min=1e-5, eta_max=1e-1, n_s=900 and n_epochs=36. The results and plots from each session was stored locally. The results are displayed in the following table:

Lambda	Final validation acc.	Mean of top 5 val. acc.	Final training acc.
0.00001	50.60%	50.38%	62.60%
0.01429	52.28%	52.06%	56.00%
0.02857	50.32%	49.61%	52.14%
0.04286	47.70%	47.59%	49.25%
0.05714	45.92%	45.69%	46.85%
0.07143	44.50%	44.21%	45.40%
0.08571	43.08%	43.00%	43.84%
0.10000	41.46%	41.50%	42.54%

Based on the highest validation set accuracies obtained, a subset of lambdas was identified for the fine search. **This subset of lambdas were found in the range [0.00001, 0.02857]**, since it was between these values that the trained classifiers yielded the highest classification accuracies.

The hyper-parameter settings for the 3 best-performing networks were:

- lambda 0.00001,
- lambda 0.01429 and
- lambda 0.02857,

all with the remaining hyper-parameters set to the values stated in the beginning of this section (n_batch=100, eta_min=1e-5, eta_max=1e-1, n_s=900 and n_epochs=36).

Fine search

In contrast to the coarse search, values for the fine search were drawn randomly from the range identified in the previous step (as opposed to being uniformly drawn). Again, 8 values were drawn, the best of which were:

- **lambda = 0.00821**, which yielded a final **validation set accuracy of 52.58%**
- **lambda = 0.00409**, which yielded a final **validation set accuracy of 52.14%**
- **lambda = 0.00123**, which yielded a final **validation set accuracy of 51.84%**

All three used the same parameter settings as stated in the previous section, i.e. n_batch=100, eta_min=1e-5, eta_max=1e-1, n_s=900 and n_epochs=36.

Best classifier

In the next step, the full dataset was used for training except for 1000 samples which were used for validation. In other words, the classifier was trained on 49,000 images and validated on 1,000. Using lambda = 0.00821 (our highest-performing candidate from our fine search), the step size *n_s* and the number of epochs *n_epochs* were now increased to 1800 and 144 respectively, which along with n_batch=100, eta_min=1e-5, eta_max=1e-1 and random seed 12345 (same settings as previously), yielded the following final results:

- Validation set accuracy: 53.72%
- Training set accuracy: 62.09%

The cost, loss and accuracy plots of this quasi-optimal classifier is shown in Figure 3 below. From the curves, one can deduce that increasing the number of epochs would probably result in a higher accuracy as the curves show a tendency to further improve in accuracy terms given more training. We leave this for a later computation.

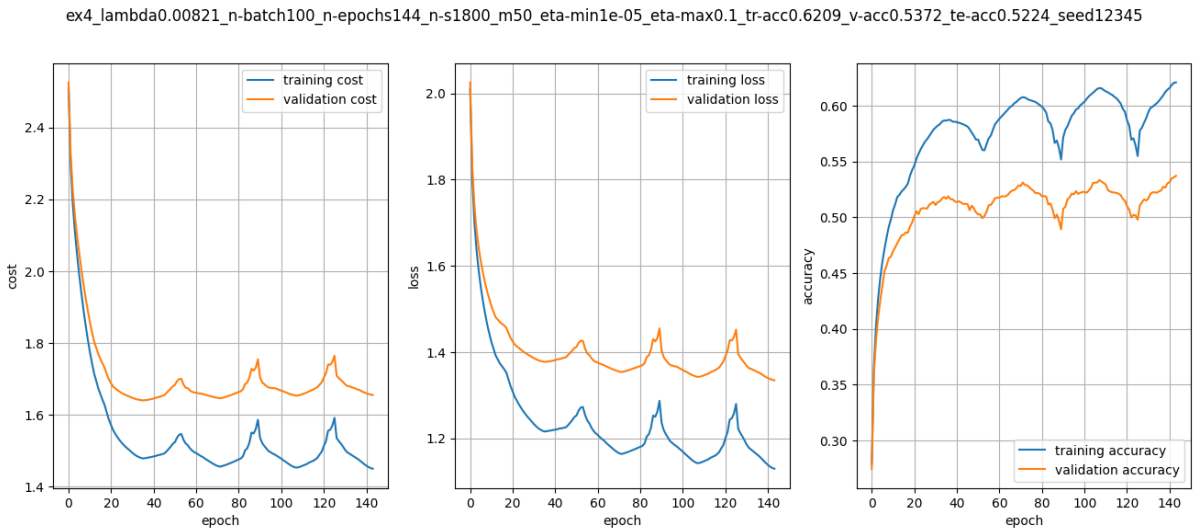

Figure 3: Cost plot, loss plot and accuracy plot respectively of our optimal classifier. Here, lambda=0.00821, n_batch=100, n_epochs=144, n_s=1800, eta_min=1e-5 and eta_max=1e-1 were used. The final validation accuracy equaled 53.72% and the final training accuracy equaled 62.09%.