DD2424 - Assignment 1

Summary

I have written code for and completed all of the problems in the assignment. However, I believe there is some bug in the training which I cannot seem to resolve. The gradients have been checked and verified numerically, but for the first three sets of suggested parameter settings, my cost and loss is somewhat higher compared to the plot in the assignment description. At the same time, my obtained accuracy is lower. Looking at the weights, there are clear differences only when $\lambda = 1.0$. Other than this, I have implemented all of the features suggested in assignment description, including epoch-wise shuffling of the training data and both of the bonus problems.

I have completed the code for the assignment in `python`. I have implemented the classificer as a class. For the hand-in, all of the code has been put toghether in a main file with all the functions and the class declared at the top. For the hand-in, I have also commented out the numerical gradient checking and the saving of generated figures.

Oskar Stigland
DD2424
Spring 2023

Training the classifier

Checking the gradients

I have checked the correctness of the analytically obtained gradients by comparing them to numerically computed gradients. For the latter, I have perturbed the full set of model parameters using a perturbation of $\epsilon = 10^{-5}$ and calculating the cost for a small subset of the training data. The maximum absolute errors for the weights and bias vectors are

$$\max_{\boldsymbol{w}} \left|\left|\nabla_{\boldsymbol{W}} J^{\text{analytical}} - \nabla_{\boldsymbol{W}} J^{\text{num}}\right|\right| < 10^{-9}$$
$$\max_{\boldsymbol{b}} \left|\left|\nabla_{\boldsymbol{b}} J^{\text{analytical}} - \nabla_{\boldsymbol{b}} J^{\text{num}}\right|\right| < 10^{-10}$$

Results

I initially tested the with the same parameter settings as the reported initial test in the assignment, i.e. `batchN` $= 100$, `epochsN` $= 40$, $\eta = 0.001$, and $\lambda = 0$. The results are shown in the figure below. Since $\lambda = 0$, I have only included the loss for this particular example, which is shown together with the accuracy and the weights. As mentioned in the summary, the loss is surprisingly high. The accuracy obtained after 40 epochs was 28.62%. There is also no discernable difference in the plotted weights.

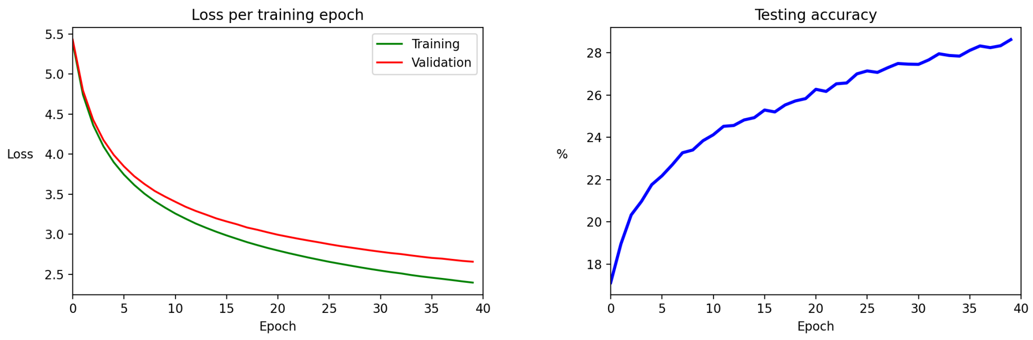

Figure 1: Loss and accuracy for initial set of parameters

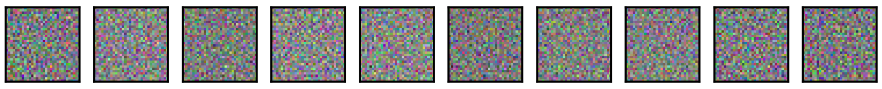

Figure 2: Class-specific weights for initial set of parameters

As mentioned in the summary, there is likely some error. However, I have not been able to identify it. The gradients have been checked numerically. The results look a lot better and more convincing for the last set of parameters, i.e. where $\eta = 0.001$ and $\lambda = 1.0$. The training converges quickly and the training accuracy reaches 37.68% after 40 epochs. Plotting the weights also show that the classes are somewhat discernable. The results are plotted below in the same fashion as for the initial parameter settings.

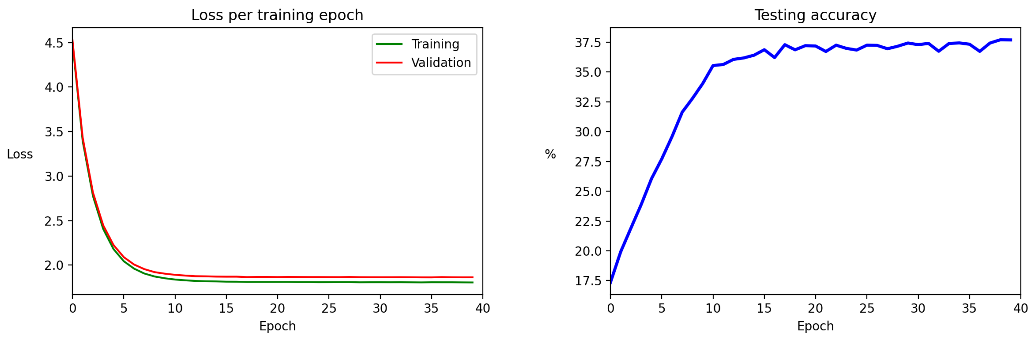

Figure 3: Loss and accuracy for $\eta = 0.001$, $\lambda = 1.0$

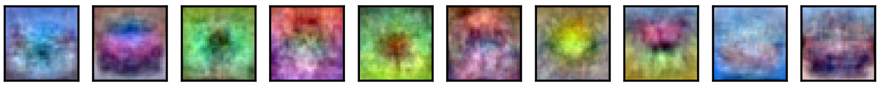

Figure 4: Class-specific weights for $\eta = 0.001$, $\lambda = 1.0$

Results summary

I have trained the classifier for all four suggested sets of parameters. The results are summarized in the table below. Again, it becomes obvious that there is something not entirely correct in the training process. The accuracy for the first two parameter sets are almost identical, even the training is much more volatile in the former case. This is displayed in the set of plots on the last page. However, the performance clearly increases as we tune up the regularization parameter. For my implementation and the suggested parameter settings, I reached the best results with $\lambda = 1.0$. Although, these results should be taken with a grain of salt due to the suspected error. When I attempt to train for a longer period and a smaller batch size, e.g. `batchN` $= 50$ and `epochsN` $= 100$, with $\eta = 0.001$ and $\lambda = 0.0$, the model achieves an accuracy of 33.58%. The corresponding maximum accuracy when using $\eta = 0.001$ and $\lambda = 0.1$ is 39.74%.

In short, the turning up the regularization parameter clearly helps a lot in this particular case. The performance on the testing data gets significantly better with $\lambda > 0$. However, there are also other aspects of the training procedure which are clearly important to take into account, such as the learning rate, the batch size and the number of epochs. That is, performance does not increase in proportion to λ, since a lower λ combined with e.g. a lower batch size actually achieves an even better result. It is also important to consider the learning rate, for which we clearly need a lower value in order to achieve a good result.

`batchN`	`epochsN`	η	λ	Accuracy, %
100	40	0.1	0.0	28.37
100	40	0.001	0.0	28.62
100	40	0.001	0.1	33.37
100	40	0.001	1.0	37.68
50	100	0.001	0.1	39.74
50	100	0.001	0.5	38.75
50	100	0.001	1.0	37.93

Loss and cost plots

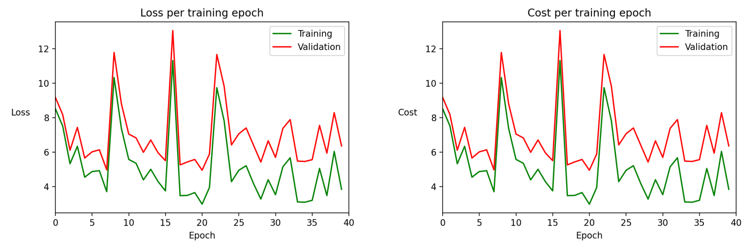

Figure 5: Loss and accuracy for $\eta = 0.1$, $\lambda = 0.0$

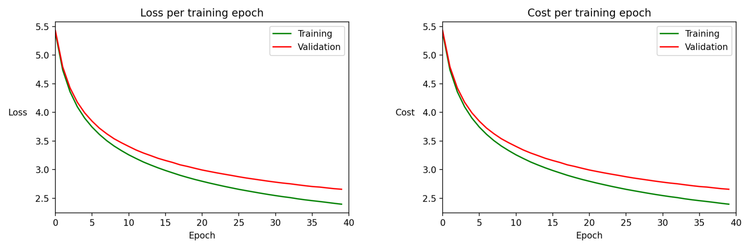

Figure 6: Loss and accuracy for $\eta = 0.001$, $\lambda = 0.0$

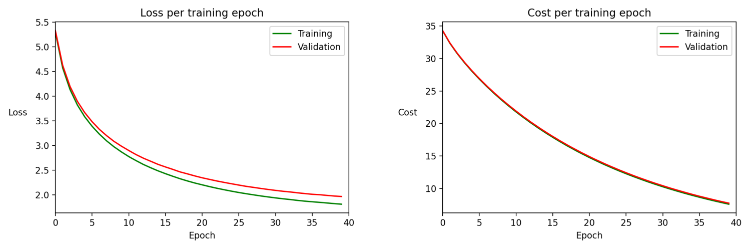

Figure 7: Loss and accuracy for $\eta = 0.001$, $\lambda = 0.1$

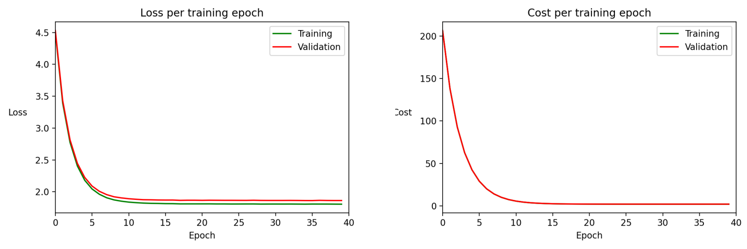

Figure 8: Loss and accuracy for $\eta = 0.001$, $\lambda = 1.0$

Weight plots

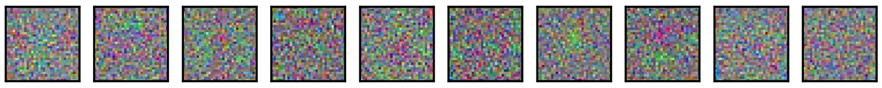

Figure 9: Loss and accuracy for $\eta = 0.1$, $\lambda = 0.0$

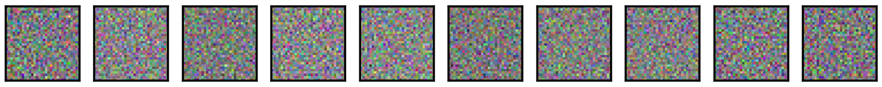

Figure 10: Loss and accuracy for $\eta = 0.001$, $\lambda = 0.0$

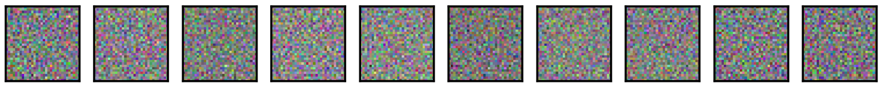

Figure 11: Loss and accuracy for $\eta = 0.001$, $\lambda = 0.1$

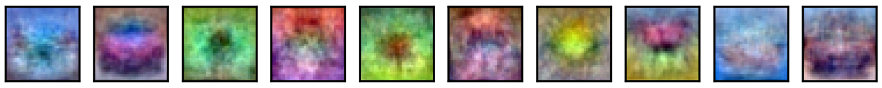

Figure 12: Loss and accuracy for $\eta = 0.001$, $\lambda = 1.0$