DD2424 - Assignment 3

Timo Nicolai

April 29, 2019

1 Gradient Checks

To verify that my analytical gradient calculations for networks with batch normalization are correct, I calculated the element-wise maximum relative difference between analytical and numerical parameter gradients for a three layer network with both hidden layers of size 50 and $\lambda = 0.5$. I used a training batch of size 20 with samples reduced to their first 10 dimensions to compute both types of gradients. I ignored the bias gradients since these tend to be very close to zero when batch normalization is enabled and thus typically exhibit high relative error in any case. Figure 1 shows the results I obtained, the relative errors are small enough to conclude that my gradient implementation is very likely correct.

Parameter	$\epsilon_{rel,max}$
∇W_1	$2.39e-4$
∇W_2	$1.43e-3$
∇W_3	$1.22e-4$
$\nabla \gamma_1$	$1.41e-6$
$\nabla \gamma_2$	$2.32e-6$
$\nabla \beta_1$	$5.17e-6$
$\nabla \beta_2$	$3.35e-5$

Figure 1: Maximum relative gradient error between analytical and numerical gradient. Numerical gradients obtained via simple forward difference with $h = 1e-6$.

2 Initial Training

Figure 2 shows learning rate schedule as well as loss, cost and accuracy development for both training and validation set for a three layer network trained without batch normalization. Figure 3 shows the same curves for a three layer network training **with** batch normalization. Similarly, Figures 4 and 5 show these curves for nine layer networks trained without and with batch normalization respectively[1]

All networks were trained with the same hyperparameter settings ($\lambda = 0.005$, $n_{batch} = 100$, $\eta_{min} = 1e-5$, $\eta_{max} = 1e-1$, $n_s = 2250$) for two cycles. The hidden layers of these networks were sized in accordance with the assignment description and all weights were initialized with the He method.

The figures show that batch normalization slightly improves the final validation error for both network architectures. Despite this, the nine layer network still performs considerably worse. The three layer network performs approximately achieves the $\approx 53.8\%$ test accuracy reported in the assignment.

[1] The assignment text describes a nine layer network while the report requirements mention a six layer network, I'll assume that the former is correct.

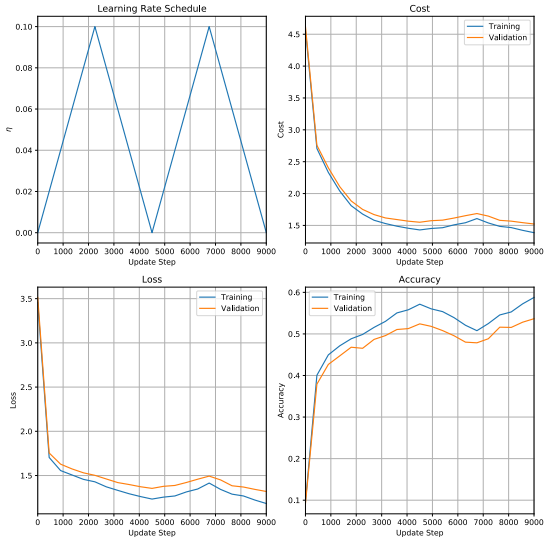

Figure 2: Learning curves for three layer network trained without batch normalization.

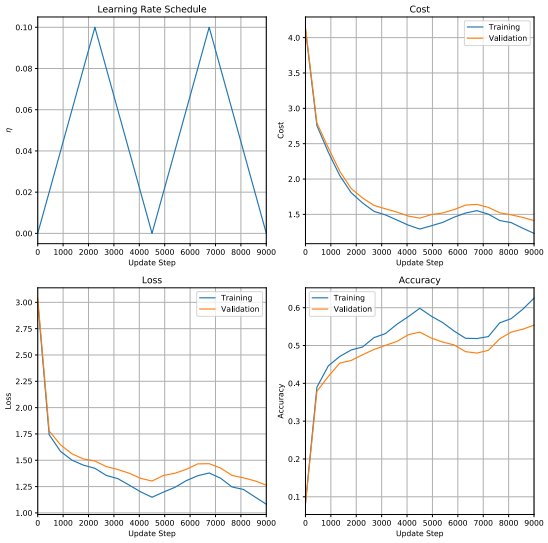

Figure 3: Learning curves for three layer network trained with batch normalization.

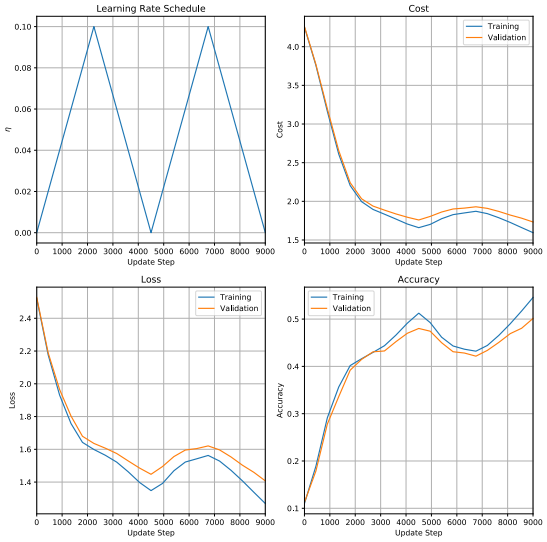

Figure 4: Learning curves for nine layer network trained without batch normalization.

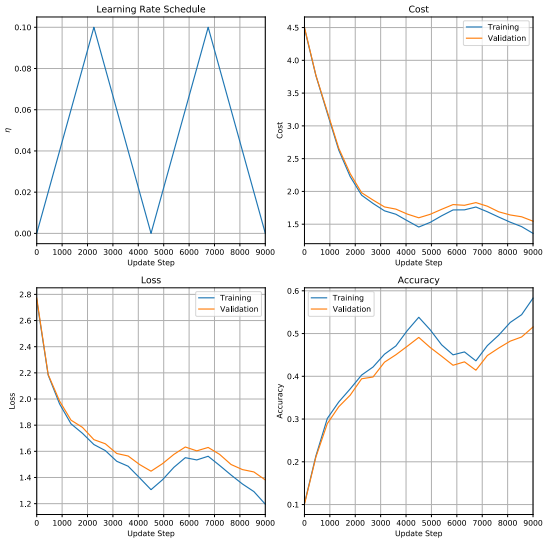

Figure 5: Learning curves for nine layer network trained with batch normalization.

3 Parameter Search

I searched for a suitable value of λ by first drawing ten random samples from the interval $[1e-5, 1e-1]$ (logarithmically) and then "homing in" on the interval $[1e-3, 1e-2]$ from which I drew another ten random samples.

Figure 6 visualizes the results of the initial coarse search, clearly showing that the best validation set accuracies are achieved in when λ lies in the range $[1e-3, 1e-2]$. Figure 7 visualized the fine search results, the best found values of λ and the corresponding achieved validation set accuracies are tabulated in Figure 8.

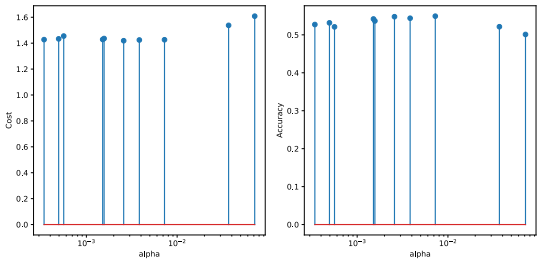

Figure 6: Coarse parameter search validation set costs and accuracies.

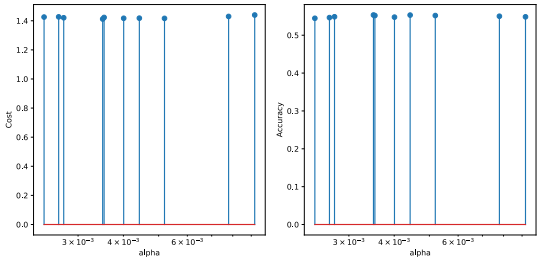

Figure 7: Fine parameter search validation set costs and accuracies.

λ	Accuracy
$3.51e-3$	≈ 0.553
$4.42e-3$	≈ 0.553
$5.19e-3$	≈ 0.552

Figure 8: Best values of λ and associated validation set accuracies.

Based on this I retrained the three layer network with batch normalization and $\lambda = 3.51e-3$ for three cycles. Figure 9 shows the trained network's test set confusion matrix with an overall accuracy of 54.4%.

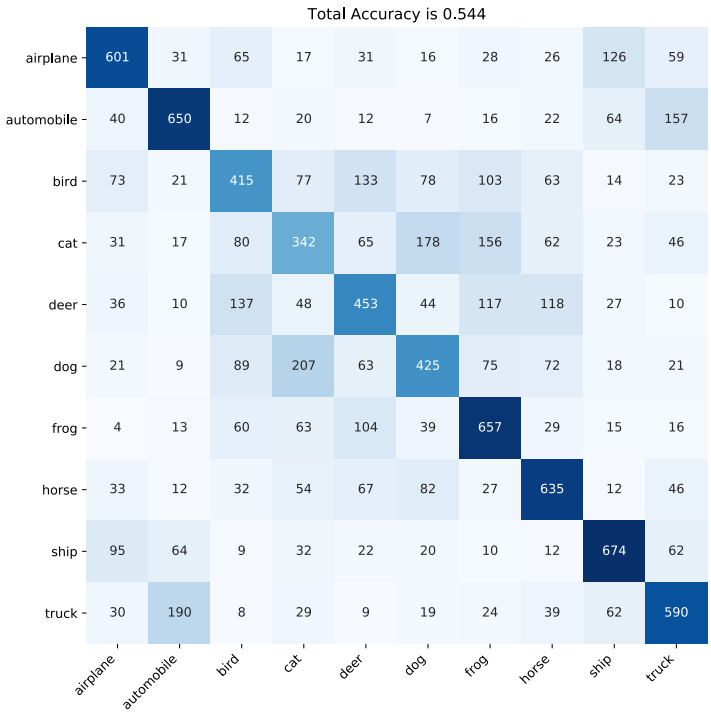

Figure 9: Test set performance of a three layer network trained with $\lambda = 3.51e-3$ and batch normalization for three cycles.

4 Parameter Initialization Sensitivity

I trained the networks as described in the assignment using a smaller $\eta_{ss} = 900$. Figure 10 shows the corresponding (training set) loss curves. It is evident, that batch normalization was able to prevent divergence for small values of σ. I presume that without batch normalization divergence occurs in these cases because the closer the initial weight values are to zero, the more gradient updates will tend to modify weights "in the same direction" thus making training prone to divergence.

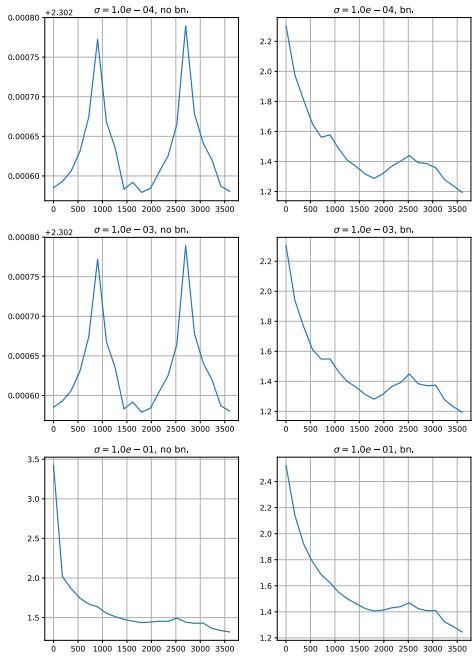

Figure 10: Training set loss curves for a three layer network trained with different weight initialization schemes and either with or without batch normalization.