Deep Learning: Assignment 2

Anton Stråhle

April 7, 2020

Introduction

In this assignment we have constructed, and trained, a 2-layer fully connected network using data from CIFAR-10.

Gradient Check

In order to check that my analytical gradients were correct, which I believe they are, I translated the given Matlab function `ComputeGradsNumSlow` to Python and then compared my analytcial gradients for W_1, W_2, b_1, b_2 with the numerical ones using the function `np.testing.assert_array_almost_equal` at 6 decimal places. The outcome was that both matrices and both vectors were equal within the first 6 digits.

The function `np.testing.assert_array_almost_equal` returns `None` in the case that two arrays are identical up to a set number of decimals. In the case that they are not different the function returns the differences per cell as well as some other information. When comparing all analytical gradients to the numerical ones obtained via centered difference the function returned `None` for all arrays with a threshold of 6 decimals, indicating that all analytcial arrays were identical to their numerical counterparts up to 6 decimals.

Cyclical Learning Rates

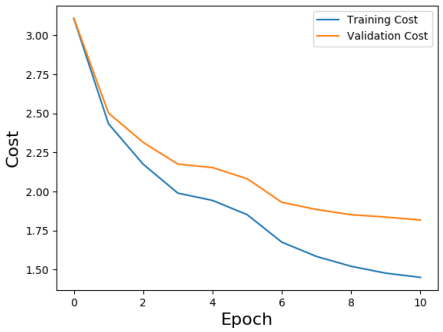

Figure 1: With $\eta_{\min} = 0.00001$, $\eta_{\max} = 0.1$, and $n_s = 500$
Trained for 1 cycle

During epoch 4-6 in Figure 1 we notice some deviations from the otherwise somewhat smooth decrease in cost. This is due to η reaching it's peak halfway through the cycle which leads to larger step sizes.

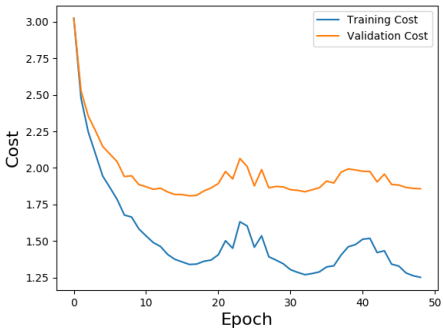

Figure 2: With $\eta_{\min} = 0.00001$, $\eta_{\max} = 0.1$, and $n_s = 800$
Trained for 3 cycles

After the initial decrease in Figure 2 we note that the cost function looks to have three local minima. By using cyclical learning rates it seems as if we do not get stuck in the first local minima that we ecounter but instead allows us to further explore many different local minima. As mentioned in the lectures, the networks at these different local minimas will be different and can therefore be combined by for example Bagging.

Coarse-to-fine Search

During the coarse search I used the following range for λ.

$$U \sim \text{Unif}(-5, -1)$$
$$\lambda = 10^U$$

During this search I found that the values of λ that generated the highest accuracies were all within, or close to, $(10^{-3}, 10^{-2})$. After another run most values seemed to be closer to 10^{-3} than 10^{-2} and as such I narrowed down the search even further to $(10^{-2.75}, 10^{-2.5})$.

During the coarse search I used the suggested $n_S = 900$ obtained through the mentioned formula and trained for two cycles. Secondly, I did not change the max or min learning rates and continued to use $\eta_{\min} = 1e-5$ and $\eta_{\max} = 1e-1$ throughout the search. The best networks generated did therefore only differ in the choice of λ and were

λ	Accuracy
0.005	51.46
0.0029	51.10
0.0012	50.94

During the fine search as previously mentioned I decreased the size of the interval further. As suggested I also increased the number of cycles as well as the cycle lengths. My first attempt was with 4 cycles and a cycle length $n_s = 1800$. For these settings, as well as limiting λ to the interval $(10^{-2.75}, 10^{-2.5})$ I was able to achieve the following results.

λ	Accuracy
0.00196	54.00
0.00199	53.72
0.0026	53.56

Best Accuracy

We now want to examine our network when using the best λ found in the Coarse-to-fine search. Using $\lambda = 0.00196$ and almost all data for training we get the following cost function over the epochs when training for the four cycles used in the fine search above. Note that when using 49000 instead of 45000 we increase n_s from 1800 to 1960. Training this network for 3 cycles gives us an accuracy of 54.5% as well as the following training and validation costs.

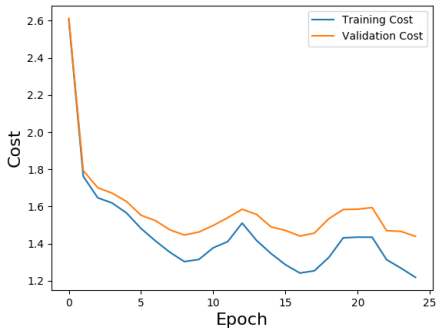

Figure 3: With $\eta_{\min} = 0.00001$, $\lambda = 0.00196$, $\eta_{\max} = 0.1$, and $n_s = 1960$