DD2424 - Deep Learning in Data Science

Assignment 2

Eloi Dieme

March 2024

Analytical Computation of the Gradient

We computed the gradients using the formulas:

$$\begin{aligned}
&\mathbf{G} = -(\mathbf{Y} - \mathbf{P}) \\
\Rightarrow &\begin{cases} \frac{\partial L}{\partial W_2} = \frac{1}{n_b}\mathbf{G}.\mathbf{H}^T + 2\lambda W_2 \\ \frac{\partial L}{\partial \mathbf{b_2}} = \frac{1}{n_b}\mathbf{G}.\mathbf{1} \end{cases} \\
&\mathbf{G} = (W_2^T.\mathbf{G}) \odot \text{Ind}(\mathbf{H} > 0) \\
\Rightarrow &\begin{cases} \frac{\partial L}{\partial W_1} = \frac{1}{n_b}\mathbf{G}.\mathbf{X}^T + 2\lambda W \\ \frac{\partial L}{\partial \mathbf{b}_1} = \frac{1}{n_b}\mathbf{G}.\mathbf{1} \end{cases}
\end{aligned}$$

We used Numpy arrays and vectorized operations for this. We tested the correctness of the gradients with a function that compares each coefficient of the numerical and analytical gradients and outputs the maximum relative difference, computed according to the formula:

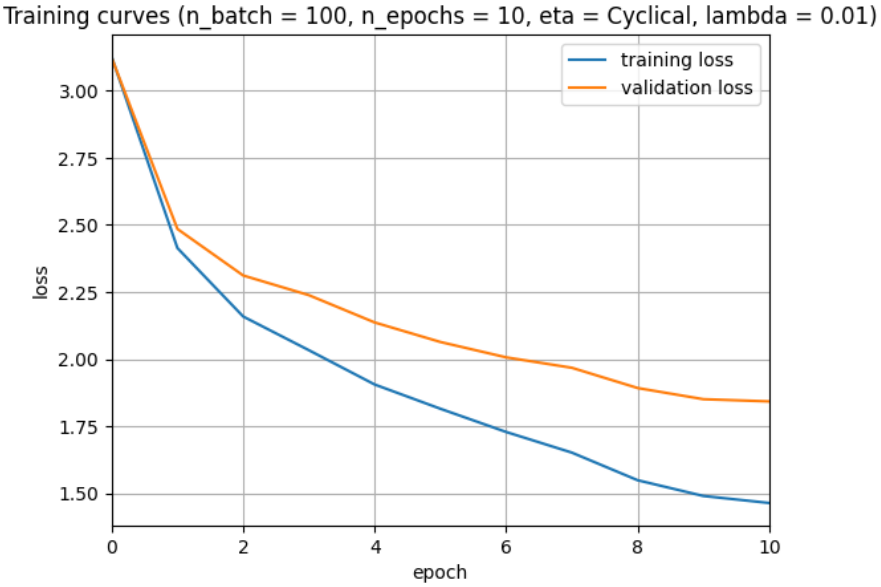
$$\frac{|g_a - g_n|}{\max(\epsilon, |g_a| + |g_n|)}$$

This difference doesn't go above `1e-5` for all the different cases, indicating that the analytical gradient is correct. To speed things up, we tested this with 20 features and 10 samples.

Training curves for default cyclical learning

- eta_min = 1e-5, eta_max = 1e-1, lambda = .01, n_s = 500 ; One cycle

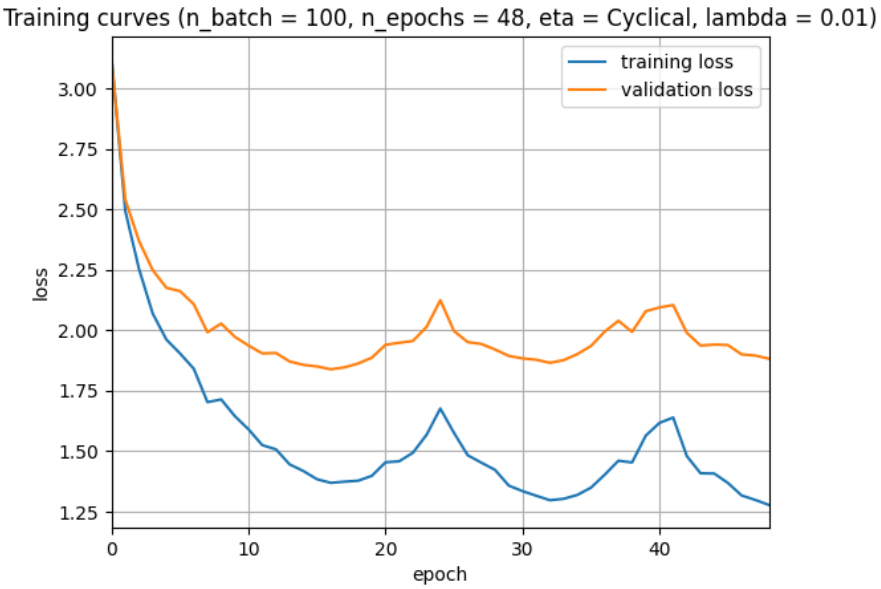
Training curves:



Accuracy on test data: **0.4596**

- eta_min = 1e-5, eta_max = 1e-1, lambda = .01, n_s = 800 ; Three cycles

Training curves:



Accuracy on test data: **0.4672**

We notice little bumps when the learning rate is at its maximum, followed by local minima. As expected, the slopes of the learning curves get smaller as we get closer to the local minimum and then rises again.

Lambda - Coarse search

First, we searched for lambda by randomly selecting 8 values between $\lambda = 10^{-5}$ and $\lambda = 10^{-1}$. Then, the network was trained for 2 cycles on the full data (except 5000 samples held out for validation). The 3 best hyperparameters and their associated accuracies were:

1. $\lambda = 9.99 \times 10^{-4}$ - **49.41%**
2. $\lambda = 3.26 \times 10^{-3}$ - **49.37%**
3. $\lambda = 1.13 \times 10^{-3}$ - **49.34%**

Then, we repeated the same process but between $\lambda = 10^{-4}$ and $\lambda = 10^{-3}$, and trained for 3 cycles. The 3 best hyperparameters and their associated accuracies were:

1. $\lambda = 5.9 \times 10^{-4}$ - **50.00%**
2. $\lambda = 6.8 \times 10^{-4}$ - **50.08%**
3. $\lambda = 7.4 \times 10^{-4}$ - **50.04%**

Lambda - Fine search

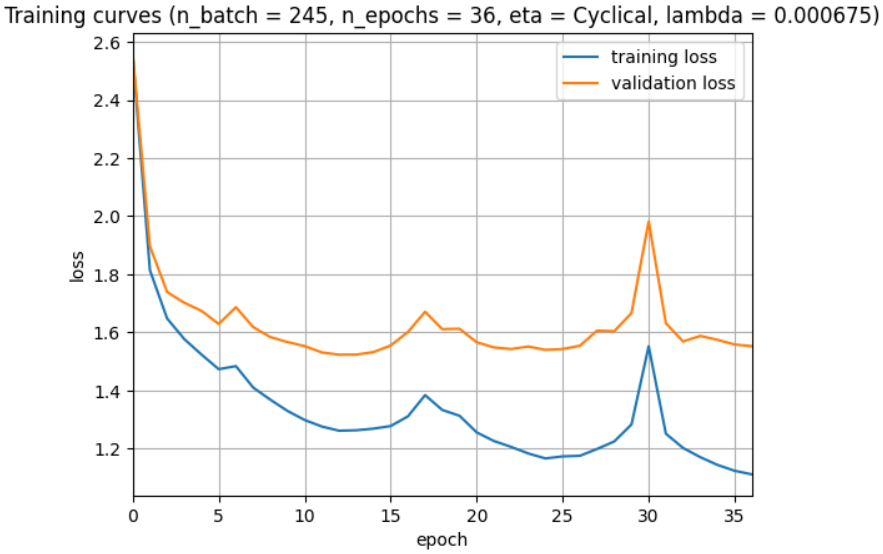
Then, for the fine-search, having reduced our search interval to $[6e-4, 7e-4]$, we selected the following values to train the network for 3 cycles:

$$\text{lambdas} = [6 \times 10^{-4},\ 6.25 \times 10^{-4},\ 6.5 \times 10^{-4},\ 6.75 \times 10^{-4},\ 7 \times 10^{-4}]$$

The best value we found was $\lambda = 6.75e-4$ with an accuracy of **50.22%.**

Final results

Having found a best value for lambda, we trained our network on all training data except for 1000 samples in the validation set, for 3 cycles. The training curves are:



Accuracy on test data: **0.5173**

Bonus Points

Optimize performance of the network

To improve the performance of the network, we implemented three methods, namely using all the available training data, grid search and step decay.

More hidden nodes

Apply dropout

Data Augmentation

Adam optimizer