Assignment2 - DD2424

Silpa Soni Nallacheruvu

Gradient Check

To verify the correctness of my analytical gradients for the two-layer neural network, I implemented a gradient comparison against numerically computed gradients from PyTorch. The gradients were derived as follows:

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \frac{1}{n}(P - Y)H^\top + 2\lambda W^{[2]}, \quad \frac{\partial \mathcal{L}}{\partial b^{[2]}} = \frac{1}{n}\sum_{i=1}^{n}(P - Y)_i$$

$$\frac{\partial \mathcal{L}}{\partial W^{[1]}} = \frac{1}{n}(G \circ \mathbf{1}_{H>0})X^\top + 2\lambda W^{[1]}, \quad \frac{\partial \mathcal{L}}{\partial b^{[1]}} = \frac{1}{n}\sum_{i=1}^{n}(G \circ \mathbf{1}_{H>0})_i$$

The implementation was tested using a small synthetic input and compared layer-wise with PyTorch-generated gradients. The results were:

```
Relative errors in W:
Layer 0: max relative error = 1.63e-14
Layer 1: max relative error = 2.12e-16

Relative errors in b:
Layer 0: max relative error = 1.60e-15
Layer 1: max relative error = 3.08e-16
```

Listing 1: Relative error of analytical and numerical gradients

The extremely small relative errors confirm that the analytical gradients are implemented correctly and align closely with the numerical gradients.

Cyclical Learning Rate Training Curves

The curves below illustrate the training and validation cost, loss, and accuracy using this learning rate schedule using the default values for the hyper-parameters from the assignment.

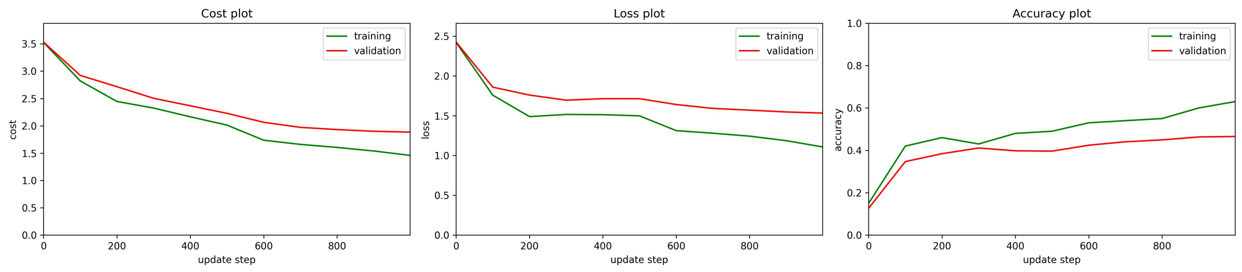

Figure 1: Training curves (cost, loss and accuracy) for one cycle of training.

The following are accuracy results:

```
Accuracy for training : 64.56% with steps 1000
Accuracy for validation : 46.52% with steps 1000
Accuracy for testing : 47.54% with steps 1000
```

Listing 2: Training and validation accuracy for one cycle

As seen in the plots, the training and validation curves in Figure 1 show consistent descent, that resemble `Figure 3` in the assignment. The curves are smooth and reflect the expected rise and fall of learning rate across one cycle.

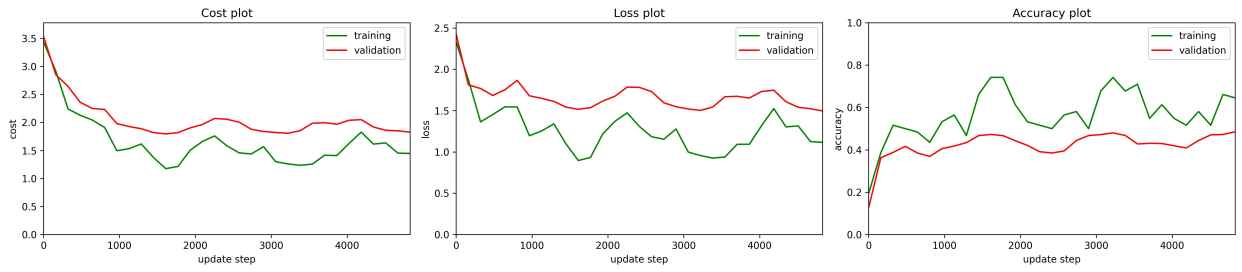

Figure 2: Training curves (cost, loss and accuracy) for three cycles of training.

The following are accuracy results:

```
Accuracy for training : 73.88% with steps 4830
Accuracy for validation : 48.48% with steps 4830
Accuracy for testing : 48.69% with steps 4830
```

Listing 3: Training and validation accuracy for three cycles

As seen in the plots, the training and validation curves in Figure 2 display more fluctuations compared to the first training algorithm, similar to `Figure 4` in the assignment. Despite these oscillations, the validation accuracy improved and converges to a higher value than the first training algorithm. This suggests that longer cycles help escape local minima and explore the loss landscape more effectively, though at the cost of more training instability.

Overall, cyclical learning rates provided effective control over the optimization dynamics, offering a balance between fast convergence and robustness against overfitting.

Coarse search for lambda

I performed a coarse search for lambda by choosing 10 different values of lambda, ranging from 10^{-5} to 10^{-1} using a uniform grid and logarithm scale as mentioned in the assignment. I ran the training for one cycle with hyper-parameters set to eta_min = 1e-5, eta_max = 1e-1, and n_batch = 100. The following are the best three performing networks based on validation accuracy:

```
lambda=0.00033731757047661924, training accuracy=72.04%, validation accuracy=53.92%
lambda=2.0365736267721594e-05, training accuracy=72.40%, validation accuracy=53.32%
lambda=0.00018613396927305185, training accuracy=72.35%, validation accuracy=53.32%
```

Listing 4: Coarse search for lambda

Fine search for lambda

I performed a fine search for lambda by choosing 10 different values of lambda in logarithm scale, by shortening the range of lambda to 10^{-4} to 10^{-2} focusing on the good settings from the coarse search. I ran the training for two cycles with the same hyper-parameters as before. The following are the best three performing networks based on validation accuracy:

```
lambda=0.0022477040596100145, training accuracy=67.24%, validation accuracy=54.76%
lambda=0.00037294936421987206, training accuracy=71.87%, validation accuracy=54.34%
lambda=0.004726183203219606, training accuracy=62.84%, validation accuracy=54.34%
```

Listing 5: Fine search for lambda

As mentioned in the assignment, all the 10 values of lambda from the fine search displayed validation accuracies above 50%, indicating that the model is learning effectively. The best performing lambda from the fine search was 0.0022, which achieved a training accuracy of 67.24% and a validation accuracy of 54.76%.

Best found lambda

I performed a final training run with the best lambda value of 0.0022, using the same hyper-parameters as before. I ran the training for three cycles and excluded 1000 examples from the training set for validation. The test data was the provided `test_batch` dataset.

The plot below indicates the training and validation curves for this run:

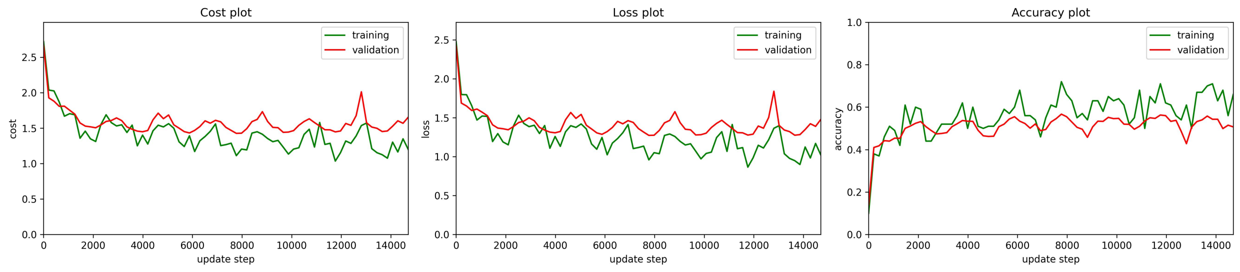

Figure 3: Training curves (cost, loss and accuracy) for three cycles of training with best lambda.

The following are accuracy results:

```
Accuracy for training : 57.01% with steps 14700
Accuracy for validation : 50.70% with steps 14700
Accuracy for testing : 48.38% with steps 14700
```

Listing 6: Training and validation accuracy for best lambda

The training curves for $\lambda = 0.0022$, identified through a coarse-to-fine grid search, show reduced overfitting and improved generalization compared to earlier experiments with $\lambda = 0.01$. The training and validation losses are closer, and accuracy is more stable, indicating better regularization.