Course: DD2424 - Assignment 2
Optional for Bonus Points

Álvaro Orgaz Expósito

April 14, 2019

1 Optimize the performance of the network

I will study what is the best possible performance achievable by a 2-layer fully connected network on CIFAR 10. There are some tricks/avenues I will explore to help bump up performance, and I will compare the performance results of them with the last parametrization case of the assignment (with the optimal λ=0.001 found), which achieves 1.4356 loss and 52.5% accuracy in the validation set. In the test data, it achieves a 51.14% accuracy. This corresponds to the parametrization case: batch size 100, 3 cycles, η_{min} 1e-5, η_{max} 1e-1, step size 2 epochs or 2 times number of images divided by batch size (in our case 2·45000/100=900). Note that it is equivalent to 12 epochs equal to 3 cycles · 2 · step size 900 / (45000 images / batch size 100).

1.1 Do a more exhaustive random search

Firstly, with the optimal found parametrization I will extend the number of cycles to 10 and see where is the optimal epoch for validation set by plotting the learning curves. Note that it is equivalent to 40 epochs equal to 10 cycles · 2 · step size 900 / (45000 images / batch size 100). In figure 1 we can find the learning curves and we can see how the accuracy in validation can not be much improved even increasing the number of epochs to 40, neither the loss.

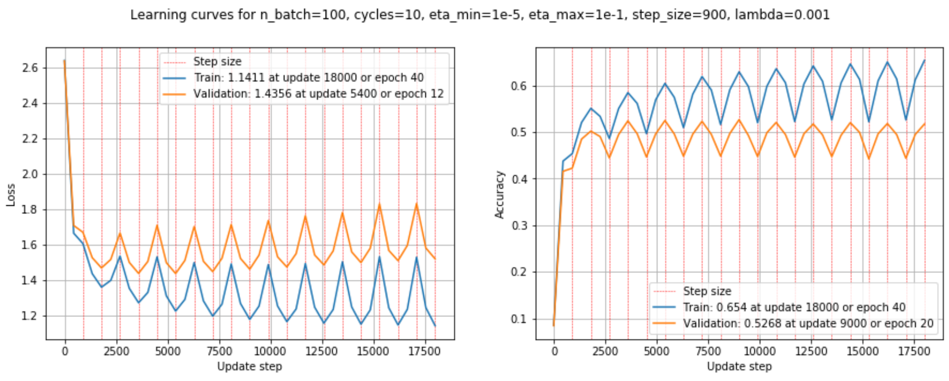

Figure 1: Learning curves for the best parametrization training for 10 cycles.

Secondly, with the same parametrization than before I will increase the step size to 1350 (add 1 more epoch by step size), since in the previous learning curves we can see how the loss/accuracy is lower/higher at the end and beginning of cycles (near minimum learning rate).

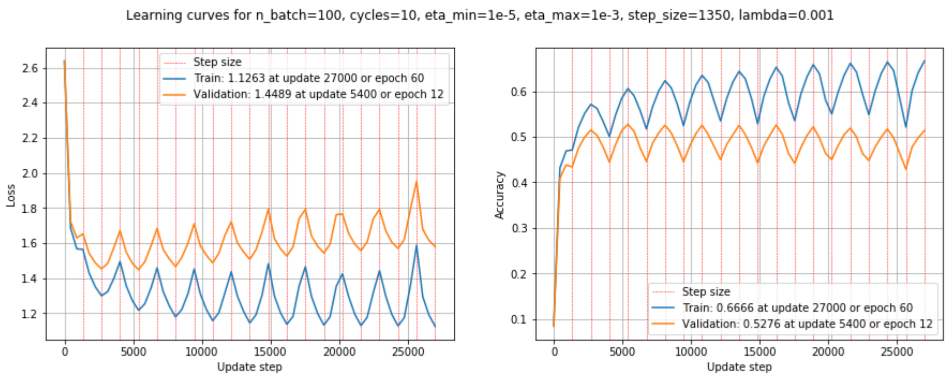

Figure 2: Learning curves for the best parametrization training for 10 cycles and step size 1350 (3 epochs).

1.2 Explore more hidden nodes

I explore whether having more hidden nodes improves the final classification rate. Since one would expect that with more hidden nodes then the amount of regularization would have to increase, I set the λ proportional to the number of hidden nodes respect to the initial case with 50. In figure 3 we can see the learning curves for the best parametrization training by number of hidden nodes and there are interesting results although the performance is not radically enhanced. Firstly, in the loss learning curves we can see how 100 hidden nodes is optimal for training set but 50 is for validation set. However, in the accuracy learning curve 150 and 200 have the same highest accuracy but in the validation set 200 seems to outperform the other some points.

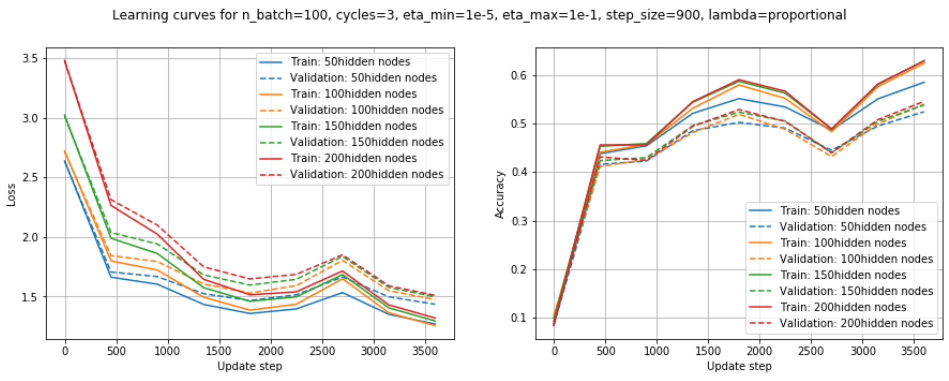

Figure 3: Learning curves for the best parametrization training by number of hidden nodes.

1.3 Ensemble weights after each cycle

I will add an option in my mini-batch gradient to save the weights after each cycle of training. Research in this area seems to indicate that with the cyclical learning rate method the local minima found at the end of each cycle are different and thus I can build an ensemble of classifiers by saving the network parameter values at the end of each cycle. Using these weights I will ensemble all the predictions to classify which may result in better classification than the last weights of the training (one model).

The results with ensembling of 10 cycles and the rest of parameters as the best parametrization mentioned are: the accuracy in the validation set is 52.54%, and in the test set is 51.19% which is not an improvement of the previous best parametrization.

1.4 Apply dropout

I will try with the regularization technique dropout in the training. Note that each training sample in the mini-batch has its own random dropout mask which means that the same activations (hidden nodes) in the hidden layer will be deactivated (turned to 0) for that sample. For selecting which nodes deactivate I will use random uniform values between 0 and 1 and if the value is above the given dropout probability of keeping the value then the hidden node is deactivated. After applying this technique, my performance results are not improved.

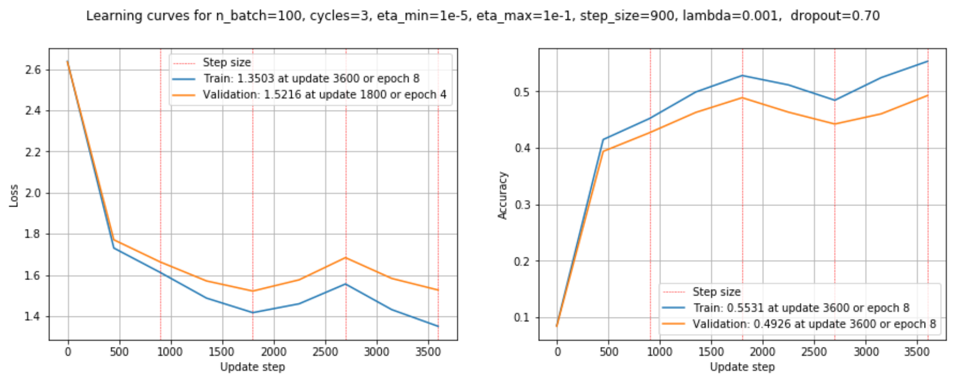

Figure 4: Learning curves for the best parametrization training with dropout 0.7.

2 Read reference *Smith, 2015* [1] and set η_{min} and η_{max} to the guidelines in the paper

From the paper, the guidelines or technique to choose the learning rate range is:

It is a "LR range test"; run your model for several epochs while letting the learning rate increase linearly between low and high LR values.

The triangular learning rate policy provides a simple mechanism to do this. For example, in Caffe, set base learning rate to the minimum value and set max learning rate to the maximum value. Set both the step size and max iter to the same number of iterations. In this case, the learning rate will increase linearly from the minimum value to the maximum value during this short run.

Next, plot the accuracy versus learning rate. Note the learning rate value when the accuracy starts to increase and when the accuracy slows, becomes ragged, or starts to fall. These two learning rates are good choices for bounds; that is, set base learning rate to the first value and set maximum learning rate to the latter value.

In figure 5 we can find the accuracy versus learning rate for the first step size of training equivalent to 10 epochs. Setting the learning rate minimum in 0 and maximum in 0.1 we can find that, following the instructions of the paper, an optimal range for my network is [0, 0.05].

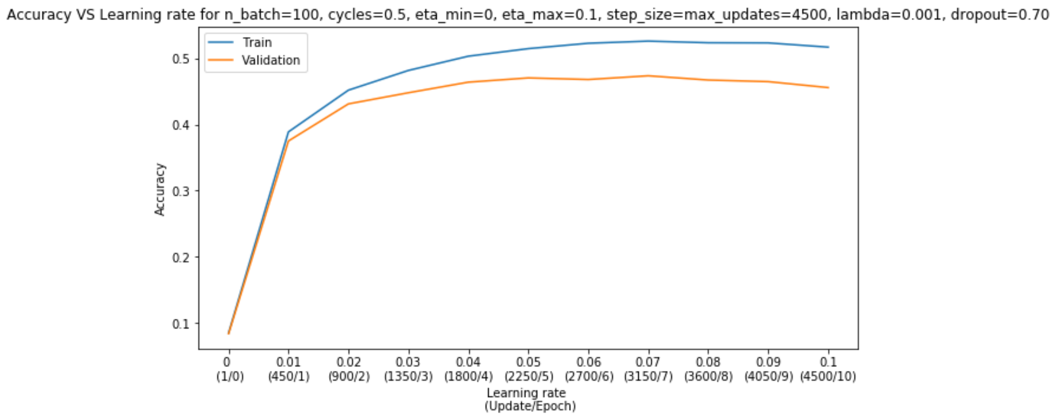

Figure 5: Accuracy versus learning rate as in [Smith, 2015] [1].

In figure 6 we can find the learning curves after finding new learning rate range and using the same initial parametrization than we are comparing with during all the report, but changing the learning rate minimum and maximum. As we can see the performance is not enhanced since it achieves a very similar performance (slightly worst).

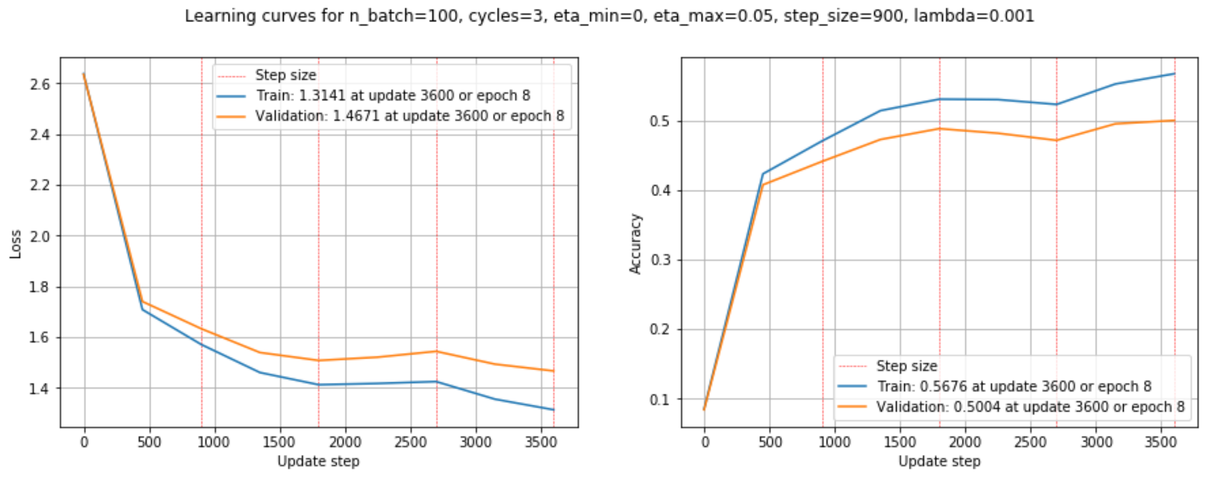

Figure 6: Learning curves after finding new learning rate range with [Smith, 2015] [1].

References

[1] Smith, L. N. (2015). Cyclical learning rates for training neural networks. *arXiv:1506.01186*.