Deep Learning: Bonus Assignment 3

Anton Stråhle

April 27, 2020

Bonus

The improvements that I have chosen are the following

(i) Increasing the number of hidden layers, and/or the number of nodes in the hidden layers

(ii) Implementing dropout

(iii) Exhaustive search for λ

In the cases below we set η_{min} to 0.00001 and η_{max} to 0.1. The batch size is also set to 100. The cycle length was set to 2250 (450×5) and the number of cycles to three.

In order to improve the accuracy of the network I started out by attempting to increase the number of hidden layers an/or the number of nodes in each layer. In order to combat this increase in nodes I also implemented dropout in order to further regularize. After my initial attempts at changing both the number of layers as well as the number of nodes I realised that the effect of additional layers started to stagnate as we reached certain depths for the network. Instead of just piling on more layers I chose to simply scale up the already existing layers by adding more nodes as this seemed to generate similar improvements compared to simply adding more layers.

A lot of the difference in the accuracies obtained in the above search for a good network structure could be attributed to the lack of a good search for the proper regularization. When examining the different architectures I examined some different drop out probabilities in combination with the scaling of λ in accordance to the total number of weight parameters. This is of course quite a sub-optimal way to go about things and as such I have chosen to do a coarse-to-fine search for λ when examining my final architecture.

One of my best performing networks, which I could train in a reasonable time was the quite arbitrarily chosen 5-layer network with $(500, 300, 200, 100, 10)$ nodes which turned out to generate a good accuracy on the validation set, about 59.96%, without a proper search for λ.

Coarse-to-fine Search

Since the network took quite some time to train I refrained from doing a random search and did instead opt for quite a quite wide and sparse grid search that I then narrowed down into more finer grids.

In the coarse search I examined the following grid

$$\begin{aligned} U &\in \{-5, -4, -3, -2, -1\} \\ \lambda &= 10^U \end{aligned}$$

For this grid the best accuracy of about 60% was obtained for $U = -2$ and as such I narrowed down my search further to

$$\begin{aligned} U &\in \{-2.75, -2.5, -2.25, -2, -1.75, -1.5\} \\ \lambda &= 10^U \end{aligned}$$

Here the best accuracies generated for the validation set increased slightly to about 60.06%. For good measure I examined an even narrower, now random, grid in search for a slightly better better accuracy

$$\begin{aligned} U &\sim \text{Unif}(-2.25, -2) \\ \lambda &= 10^U \end{aligned}$$

Realistically I should most likely have stopped here since any difference in accuracy can more than likely be attributed to the difference in initialization. In this random search the best accuracy that I found was about 60.16% for $\lambda = 0.008$.

The final network with good regularization achieves a final accuracy of about 59% on the test set which is a significant increase from the 54% obtained in the base assignment. This accuracy could of course be increased by adding more layers and nodes. Another thing that could be attempted is a 2D grid search for λ and the dropout probability. I fixed the dropout probability in this case to an arbitrary, but adequately performing, 80% in order to reduce the number of networks I would have to train.