Report bonus assignment 3

Shiva Besharat Pour
shivabp@kth.se

For the bonus exercise, I decided to optimize the batch-normalized implementation of the k-layer network. The optimization methods I chose are:

- Exhaustive search for lambda
- Investigation of the best performing network architecture in terms of number of hidden layers
- Adding noise to the training data

I will go through the results of each of the above used optimization methods.
NOTE: In all figures below, red line represents the training data and blue line represents the validation data.

Exhaustive search for lambda:

In the default assignment, I iterated 15 times within the range of [-6, -2]. However, the best lambda found was not really effective in terms of improving performance and did not make much of a difference. Therefore, I increased the number of iterations to 25 and decreased the range to [-4, -2]. The reason was that the best lambda found seemed to be very close to 0.005 which was suggested by the instructions initially. Therefore, I thought that a lambda that could potentially improve the performance should not be that far from 0.005. Rest of the parameters were kept the same as follows:

```
d = 3072
k = 10
eta_min = 1e-5
eta_max = 1e-1
l_min = -4
l_max = -2
n_layers = 3
```

The best lambda value as suggested by the search results is:

```
0.005080
```

The results are as follows:
Lambda: 0.005 **test accuracy:** 53.77
Lambda: 0.005080 **test accuracy:** 53.82

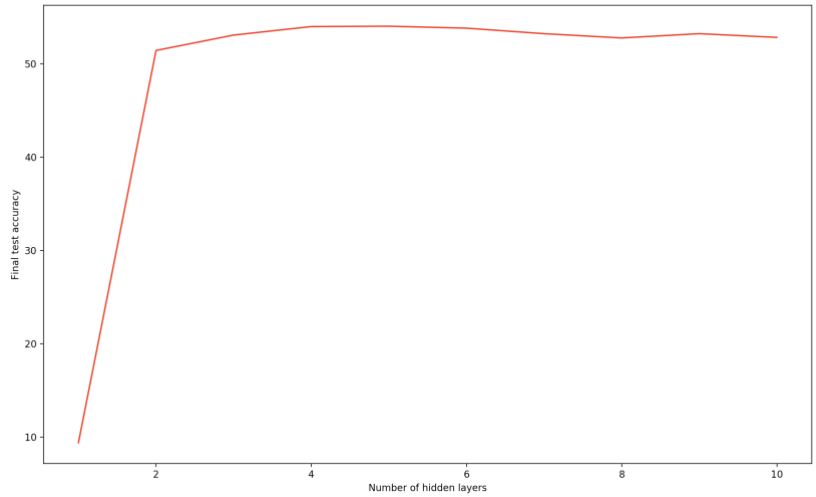
The improvement was not very significant which is expected since the lambda as suggested by the search is very close to the initial lambda used, 0.005.

Network architecture:

I experimented with increasing the number of hidden layers one by one from 1 to 10 in order to investigate when will the network be at its best performance and when will it start to over fit due to network's over-complexity. Of course, parameters such as hidden nodes per layer were kept fixed at 50. Rest of the parameters were the same as the parameters suggested by the instructions for the default assignment. Results are presented below:

Number of hidden layers	Final test accuracy
1	39.38
2	51.43
3	53.08
4	53.99
5	54.03
6	53.82
7	53.23
8	52.77
9	53.23
10	52.83

The relevance of the final test accuracy is due to the fact that too simple network will cause under training and too complex network will cause over training. In the latter case, the network will over fit to the training data and will not be able to generalize and perform well on unseen data which implies low test accuracy. The complexity of the network is of course measured with respect to the particular data being learned and the level of complexity that it requires. The diagram below shows changes of final test accuracy with respect to the number of hidden layers.

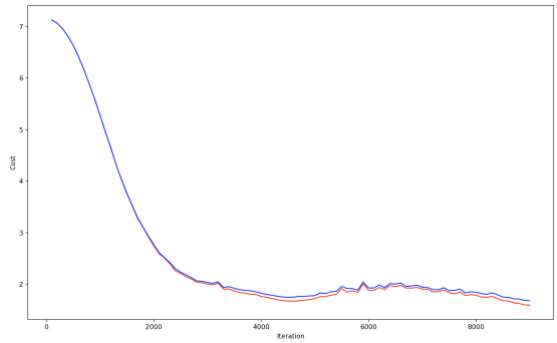


According to the diagram as well as the table above, the network generalization capability improves up until 5 hidden layers and then the test accuracy starts to decrease. Thus, implying that the network becomes too complex for the particular data used with more than 5 hidden layers and starts to over fit to the training data. Thus, the optimal number of hidden layers to use is 5 with the particular set of parameters I used and the data given.

Adding noise to the training data:

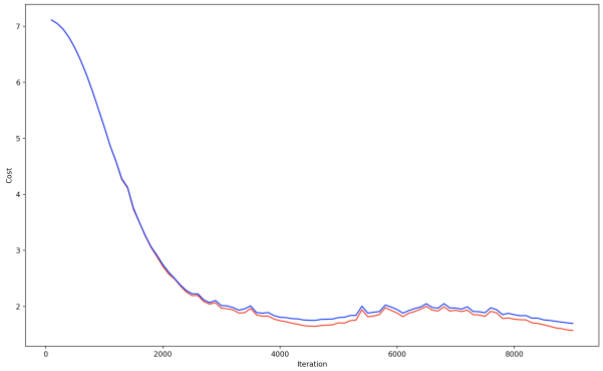
Noise is usually used to avoid network overfitting to training data and is expected to improve final test accuracy. Thus, noise would be particularly practical when the network architecture may be too complex for the data. Therefore, I experimented with adding noise to data directly by using a quite complex network with 20 hidden layers. The number of hidden nodes per layer was 50. The results are as follows:

Before adding noise:




```
Test accuracy:  46.69
```

After adding noise:




```
Test accuracy:  47.08
```

the results do confirm that the noise added did improve final test accuracy and thereby the network's generalization capability.