1
Learning Problems

The success of neural networks spawned a great interest in the community in the fields of learnability of the various models. This involves testing different models on the same problem and observing the results. It is beneficial to understand the learning dynamics of the models, which helps to find out the limitations. These studies achieved striking success in understanding the neural networks.

1.1 Parity Learning Problem

In [1], authors questioned how far neural networks could go beyond the linear models. They did this by focusing on parities that have a complex family of target functions. They demonstrated that this family could be approximated by a two-layer network trained with SGD, but not by lazy methods. This study brings an explanation of why neural networks' performance is better than linear methods, and it proves neural networks' learning capacities are beyond lazy methods.
Experiments are performed on the MNIST dataset by imitating the parity problem. The task is: given the parameter k (defines the number of digits to be stacked together that is chosen uniformly from the dataset), determine if the sum of the digits is odd or even. When $k = 1$, it is a simplified version of the standard MNIST task to find if a digit is even or odd. Experiment results showed that all models,

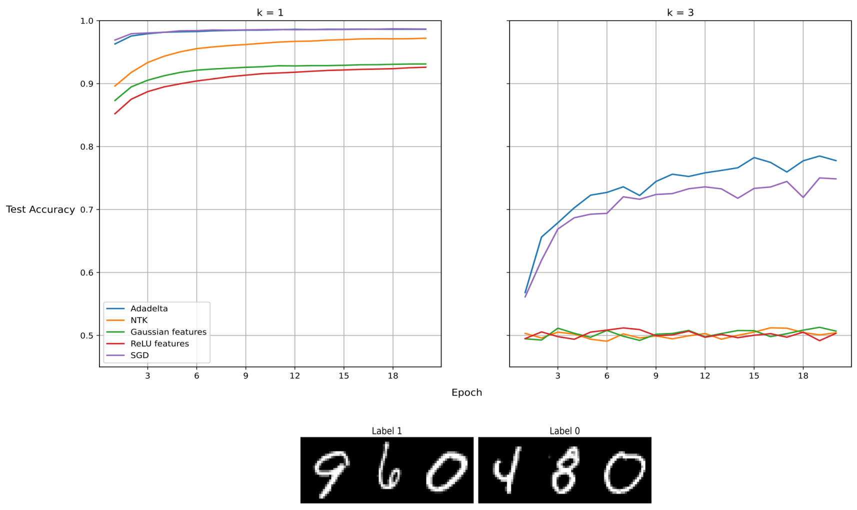

Figure 1.1: Reproduce MNIST parity experiment result from [1]

including the lazy ones, reached a similar performance where the neural network slightly outperformed others. On the other hand, the problem becomes more difficult for the case $k = 3$ because models need to compute the parity of the digits' sum. In this case, there is a drastic gap between the neural network and lazy methods. Since our goal is comparing DFA and BP on this particular problem, reproducing the results from [1] is unavoidable. The same configurations are used with minor differences. The network has only a hidden layer with 512 neurons. For the last layer, sigmoid is used as a non-linear activation function. For the hidden layer, reLU is used. BCE is preferred as a loss function, and 10^{-3} is set to weight decay. We also used SGD to observe how much Adadelta improves, and for the case $k = 3$, we performed a simple hyper-parameter tuning process to get a decent learning rate for each method. Same values are used for the case $k = 1$. The hyperparameter tuning process is the following. First, we define the parameter space, later we run with all different learning rates, and we compare these runs by the average of test accuracy of the last ten epochs, and we choose the highest one. It is also crucial to mention that, at each epoch, train data is recreated to boost the available data for the models. The same is also performed for test data to have an unbiased estimation of test accuracy. The reproduced result can be observed in figure 1.1. Similar to results from [1] all the methods succeed learning for $k = 1$ case. However, adadelta and SGD outperformed lazy methods in this setting. In the case of $k = 3$, adadelta and SGD almost reach 80%, but the performance of lazy methods doesn't go beyond a random guess.

1.2 Random Data

References

[1] Amit Daniely and Eran Malach. “Learning Parities with Neural Networks”. In: *CoRR* abs/2002.07400 (2020). arXiv: `2002.07400`. URL: `https://arxiv.org/abs/2002.07400`.