CSE 591 - Introduction to Deep Learning
Mini Project 3

Bijan Fakhri
School of Computing, Informatics and
Decision Systems Engineering
Arizona State University
Tempe, Arizona 85044
Email: bfakhri@asu.edu

Abstract—**Hyperparameter tuning is an important aspect of deep learning (or any machine learning in general). The goal of Mini Project 3 is to become familiar with the Theano toolbox Yann by tuning the hyperparameters of the Yann MLNN tutorial. Hyperparameters were heuristically ordered from greatest to least impactful (in terms of their impact on error rate). The error rate of my tuned network was 98.53%.**

1. Introduction

The parameters in question were regularization, optimization technique, momentum technique, and the learning rate. Below is a table of tunable hyperparameters, their options and values, under the scope of this mini project.

Hyperparameter	Options	Values
Regularization	ON OFF	L1 Coeff L2 Coeff
Optimization	RMSProp AdaGrad	–
Momentum	None Polyak Nesterov	StartVal EndVal EndEpoch
Learning Rate	–	AnnealingFactor FirstEraRate SecondEraRate

After succesfully installing Yann, the tutorial was run using the default values. The default values gave very good results at **98.39%**. The network was then purged of tuned hyperparameters, making the network as simple as possible, creating a good baseline to work with. Training/testing the clean network resulted in an accuracy of **97.98%**. This became the baseline. A table of the state of this network is below.

Hyperparameter	Options	Values
Regularization	OFF	–
Optimization	RMSProp	–
Momentum	None	–
Learning Rate	–	(0.05, 0.01, 0.001)

2. Tuning

Based on intuition, I ranked the hyperparameters in order of what I though would have the largest effect: *Regularization, Optimization, Momentum, LearningRate*. Begining with regularization, turning it on and with the default parameters (0.001, 0.001), accuracy was affected very slightly negatively to **97.97%**. Raising the L1 and L2 coefficient to 0.01 and 0.02 raised accuracy to **98.18%**. Deciding to keep this, I moved onto *Optimization*. Changing it to textitAdagrad had the negative effect of reducing accuracy to **98.11%**. No further exploration of optimization technique was deemed necessary. Next hyperparameter to tune was *Momentum*. Adding *Momentum* with values (0.5, 0.95, 30) resulted in poor gains. Raising the startVal to 0.9 ((0.9, 0.95, 30) resulted in the greatest improvement in accuracy: Polyak *Momentum* resulted in an accuracy of **98.53%**. Trying Nesterov *Momentum* gave comparable but less stable (changed each iteration) results. For this reason, Polyak was chosen going forward. Unfortunately, with *LearningRate*, deviating from the default values of (0.1,0.001,0.005) only resulted in a reduction in accuracy. The final state of the network (best local minima) is listed below:

Hyperparameter	Options	Values
Regularization	ON	(0.01, 0.02)
Optimization	RMSProp	–
Momentum	Polyak	(0.9, 0.95, 30)–
Learning Rate	–	(0.05, 0.01, 0.001)

3. Conclusion

In conclusion, the hyperparameters on the Yann MLNN tutorial were highly tuned from the start: the error rate of the tuned network was **98.39%**. Because of this, there was not a lot of room for improvement. Starting from a more reasonable baseline (stripping the network of tuned parameters to learn the tuning manually) resulted in a tuning gain of **98.53% - 97.98% = 0.55 %**.