Final Project Report

Alex Enze Liu

TTIC31230 - Fundamentals of Deep Learning

January 17 2017

Contents

1. Project Proposal

1.1. Project Topic

This project is trying to compare standard SGD to RMSProp and Adam under rigorous hyperparameter tuning based on project 4. Perplexity and Average Loss are mainly used for evaluating the performance of SGD, RMSProp and Adam.

There are a number of hyperparameters in the above algorithms, such as learning rate, decay rate, batch size, depth, etc. To simplify the experiment, I only consider a few among them. The hyperparameters taken into consideration are listed in table 1.

	SGD	RMSProp	Adam
Hyperparameters	Learning Rate	Learning Rate Decay Rate	Learning Rate First Moment Decay Rate Second Moment Decay Rate

Table 1: Hyperparameters To Tune

1.2. Design of Experiment

The data set will be divided into 3 subsets: train set, test set and validation set (with respectable size). In this project, the learning rate is fixed during training.

The experiment will mainly use random search, which is proposed by [1].

First, as is suggested in [2], the experiment will apply random search based on log scale on learning rate and decay rate. For example, the experiment will sample the hyperparameters on a very large scale first: $\eta = 10^{uniform(-8,3)}, \beta = 1 - 10^{uniform(-4,-1)}$. The search space of decay rate initially is small, as [3] and [4] already gives a few values, such as 0.9, 0.99 and 0.999. Next, based on the results, a few (η, β) pairs with less loss will be selected for further experimentation. The search space will then be narrowed based on selected pairs. Finally, selecting the learning rate repeatedly until the search space is narrowed down to a very small interval. [2] also recommends that the search could also be staged. That is, while the search interval is being narrowed, more epochs are performed during training.

Other possible approaches reported by [5] for tuning hyperparamters, such as Sequential Model-based Global Optimization and Tree-structured Parzen Estimator, might also be used for tuning hyperparameters.

1.3. Possible Results and Explanations

There are many possible results, and some of them are very hard to explain.

- Normally we expect Adam to converges first, then RMSProp, finally SGD. The reason of this can relate to why RMSProp and SGD are proposed. RMSProp fixes the problem in SGD when the optimal point is on the shallow direction (i.e. the gradient is small on that direction). RMSProp rescales the gradient and accelerates the progress. Adam in some sense combines momentum and RMSProp, and compares favorably to Adam or SGD [6].

- One possible result is that Adam converges first, then SGD, then RMSProp. As is mentioned above, RMSProp rescales the gradient. It is possible that rescaling actually slows down the progress of converging. By comparison, Adam updates are estimated using a running average of first and second moment of the gradient [3]. [3] also points out another reason that might lead to slow convergence in RMSProp; RMSProp lacks a bias-correction term, which could result in very large stepsizes and divergence.

- It is also possible that Adam converges first, then RMSProp, and SGD does not converge. This happens when there is a saddle point. [6] shows that SGD actually stick in the saddle point while Adam and RMSProp will finally get out of the saddle point.

- Another possible result is that RMSProp performs better than Adam. This is reported by [7], when ϵ in Adam is not carefully selected. Since ϵ is not part of the experiment, a bad choice of ϵ could lead to this result.

- [8] shows another possible situation, where RMSProp and SGD perform slightly better than Adam in the beginning, but in the end Adam has the best performance. The reason is unknown. It is probably because of the special dataset.

- Finally, when doing the previous experimentation on momentum and SGD, I found that sometimes momentum and SGD produced almost same results in terms of average loss. So I think it is possible that SGD, Adam and RMSProp have same results, on condition that the hyperparameters, such as learning rate, are not well-selected.

References

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[2] A. Karpathy, "Cs231n convolutional neural networks for visual recognition," *CS231 Course Websites http://cs231n.github.io/neural-networks-3/hyper,[Online]*, 2016.

[3] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[4] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a overview of mini–batch gradient descent," *Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture,[Online]*, 2012.

[5] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.

[6] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[7] S. Funk, "Rmsprop loses to smorms3 - beware the epsilon!," *Blog http://sifter.org/ simon/journal/20150420.html,[Online]*, 2015.

[8] M. E. Khan, R. Babanezhad, W. Lin, M. Schmidt, and M. Sugiyama, "Faster stochastic variational inference using proximal-gradient methods with general divergence functions," *arXiv preprint arXiv:1511.00146*, 2015.