Adaptive Learning Rates in Non-Convex Optimization for Machine Learning Models

Mr.Balaji S[1], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
balaji.s@shanmugha.edu.in
Mrs. K.P.Vijayalakshmi[2], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
vijayalakshmi@shanmugha.edu.in
Ms. Pavithra P M[3], Assistant Professor
Department of AI&DS, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
pavithra.pm@shanmugha.edu.in
Mrs.S.Mahalakshmi[4], Assistant Professor
Department of AI&DS, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
mahalakshmi.s@shanmugha.edu.in

Abstract- adaptive gradient methods that rely on scaling gradients down by the square root of exponential moving averages of past squared gradients, such RMSPROP, ADAM, ADADELTA have found wide application in optimizing the non convex problems that arise in deep learning. However, it has been recently demonstrated that such methods can fail to converge even in simple convex optimization settings. In this work, we provide a new analysis of such methods applied to non convex stochastic optimization problems, characterizing the effect of increasing mini batch size. Our analysis shows that under this scenario such methods do converge to stationary up to the statistical limit of variance in the stochastic gradients (scaled by a constant factor). In particular, our result implies that increasing mini batch sizes enables convergence, thus providing a way to circumvent the non convergence issues. Fur- thermoses, we provide a new adaptive optimization algorithm, YOGI, which controls the increase in effective learning rate, leading to even better performance with similar theoretical guarantees on convergence.

Keywords – RMSPROP, ADAM, ADADELTA, non-convex problems, deep learning, adaptive gradient methods.

INTRODUCTION

Stochastic gradient descent (SGD) is the dominant method for solving such optimization problems, especially in non convex settings. SGD iteratively updates the parameters of the model by moving them in the direction of the negative gradient computed on a mini batch scaled by step length, typically referred to as learning rate[1]. One has to decay this learning rate as the algorithm proceeds in order to control the variance in the stochastic gradients computed over a mini batch and thereby, ensure convergence. Hand tuning the learning rate decay in SGD is often painstakingly hard. To tackle this issue, several methods that automatically decay the learning rate have been proposed[2]. The first prominent algorithms in this line of research is ADAGRAD,

which uses a per-dimension learning rate based on squared past gradients. ADAGRAD achieved significant performance gains in comparison to SGD when the gradients are sparse. Although ADAGRAD has been demonstrated to work well in sparse settings, it has been observed that its performance[3], unfortunately, degrades in dense and nonconvex settings. This degraded performance is often attributed to the rapid decay in the learning rate when gradients are dense, which is often the case in many machine learning applications. Several methods have been proposed in the deep learning literature to alleviate this issue. One such popular approach is to use gradients scaled down by square roots of exponential moving averages of squared past gradients instead of cumulative sum of squared gradients in ADAGRAD[4]. The basic intuition behind these approaches is to adaptively tune the learning rate based on only the recent gradients; thereby, limiting the reliance of the update on only the past few gradients. RMSPROP, ADAM, ADADELTA are just few of many methods based on this update mechanism. Exponential moving average (EMA) based adaptive methods are very popular in the deep learning community. These methods have been successfully employed in plethora of applications. ADAM and RMSPROP, in particular[5], have been instrumental in achieving state-of-the-art results in many applications. At the same time, there have also been concerns about their convergence and generalization properties, indicating that despite their widespread use, our understanding of these algorithms is still very limited. Recently, showed that EMA based adaptive methods may not converge to the optimal solution even in simple convex settings when a constant mini batch size is used[6]. Their analysis relies on the fact that the effective learning rate (i.e. the learning rate parameter divided by square root of an exponential moving average of squared past gradients) of EMA methods can potentially increase over time in a fairly quick manner, and for convergence it is important to have the learning rate decrease over iterations, or at least have controlled increase. This issue persists even if the learning rate parameter is decreased over iterations. Despite the problem of non-convergence demonstrated by, their work does not rule out convergence in case the mini batch size is increased with time, thereby decreasing the variance of the stochastic gradients. Increasing mini batch size has been shown to help convergence in a few optimization algorithms that are not based on EMA methods[7].

RELATED WORK

The literature in stochastic optimization is vast; so we summarize a few very closely related works. SGD and its accelerated variants for smooth nonconvex problems are analyzed in. Stochastic methods have also been employed in nonconvex finite-sum problems where stronger results can be shown. However, none of these methods are adaptive and can exhibit poor performance in ill-conditioned problems. All the aforementioned works show convergence to a stationary point[8]. Recently, several first-order and second-order methods have been proposed that are guaranteed to converge to local minima under certain conditions. However, these methods are computationally expensive or exhibit slow convergence in practice, making them unsuitable for large-scale settings. Adaptive methods, that include ADAGRAD, RMSPROP, ADAM, ADADELTA have been mostly studied in the convex setting but their analysis in the nonconvex setting is largely missin. Normalized variants of SGD have been studied recently in nonconvex settings[9]. We develop convergence analysis for ADAM under certain useful parameter settings, showing convergence to a stationary point up to the statistical limit of variance in the stochastic gradients (scaled by a constant factor) even for nonconvex problems. Our analysis implies that increasing batch size will lead to convergence, as increasing batch size decreases variance linearly. Our work thus provides a principled means to circumvent the non-convergence results of.

- Inspired by our analysis of ADAM and the intuition that controlled increase of effective learning rate is essential for good convergence, we also propose a new algorithm (YOGI) for achieving adaptivity in SGD. Similar to the results in ADAM, we show convergence results with increasing minibatch size. Our analysis also highlights the interplay between level of "adaptivity" and convergence of the algorithm[10].
- We provide extensive empirical experiments for YOGI and show that it performs better than ADAM in many state-of-the-art machine learning models. We also demonstrate that YOGI achieves similar, or better, results to best performance reported on these models without much hyperparameter tuning[11].

ANALYZE ALGORITHMS

we discuss adaptive methods and analyze their convergence behavior in the nonconvex setting. In particular, we focus on two algorithms: ADAM (3.1) and the proposed method, YOGI (3.2).

(i) Adam- ADAM is an adaptive method based on EMA[12], which is popular among the deep learning community. EMA based adaptive methods were initially inspired from ADAGRAD and were proposed to address the problem of rapid decay of learning rate in ADAGRAD. These methods scale down the gradient by the square roots of EMA of past squared gradients. The pseudocode for ADAM is provided in Algorithm 1. The terms mt and vt in Algorithm 1 are EMA of the gradients and squared gradients respectively. Note that here, for the sake of clarity, the debiasing step used in the original paper by removed but our results also apply to the debiased version. A value of β1 = 0.9, β2 = 0.999 and = 10−8 is typically recommended in practice[13]. The parameter, which was initially designed to avoid precision issues in practical implementations, is often overlooked. However, it has been observed that very small in some applications has also resulted in performance issues, indicating that it has a role to play in convergence of the algorithm. Intuitively captures the amount of "adaptivity" in ADAM: larger values of imply weaker adaptivity

since dominates vt in this case.

Algorithm 1 ADAM

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, decay parameters $0 \le \beta_1, \beta_2 \le 1, \epsilon > 0$
Set $m_0 = 0, v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t = v_{t-1} - (1-\beta_2)(v_{t-1} - g_t^2)$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

Algorithm 2 YOGI

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1, \epsilon > 0$
Set $m_0 = 0, v_0 = 0$
for $t = 1$ **to** T **do**
  Draw a sample s_t from $\mathbb{P}$.
  Compute $g_t = \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t = v_{t-1} - (1-\beta_2)\text{sign}(v_{t-1} - g_t^2) g_t^2$
  $x_{t+1} = x_t - \eta_t m_t / (\sqrt{v_t} + \epsilon)$
end for

Very recently, has shown non-convergence of ADAM in simple online convex settings, assuming constant minibatch sizes. These results naturally apply to the nonconvex setting too. It is, however, interesting to consider the case of ADAM in nonconvex setting with increasing batch sizes. To this end, we prove the following convergence result for nonconvex setting. In this paper, for the sake of simplicity, we analyze the case where β1 = 0, which is typically referred to as RMSPROP. However, our analysis should extend to the general case as well.

RESULTS

We would like to emphasize that the SFO complexity obtained here for ADAM or YOGI with large batch size is similar to that of SGD (see Corollary 1). While we stated our theoretical results with batch size b = Θ(T) for the sake of simplicity, similar results can be obtained for increasing minibatches bt = Θ(t). In practice, we believe a much weaker increase in batch size is sufficient. In fact, when the variance is not large, our analysis shows that a reasonably large batch size can work well. Before we proceed further, note that these are upper bounds and may not be completely reflective of the performance in practice. It is, however, instructive to note the relationship between different quantities of these algorithms in our results. In particular, the amount of adaptivity that can be tolerated depends on the parameter β2. This convergence analysis s useful when ϵ G is large when compared to 1 − β2 i.e., the adaptivity level is moderate3. Recall that ϵ here is only a parameter of the algorithm and is not associated with accuracy of the solution. Typically, it is often desirable to have small in adaptive methods; however, as we shall see in the experiment section, limiting the adaptivity level to a certain extent almost always improves the performance (e.g. see in Figure 1). For this reason, we also set the adaptivity level to a moderate value of ϵ = 10−3 for YOGI across all our experiments.

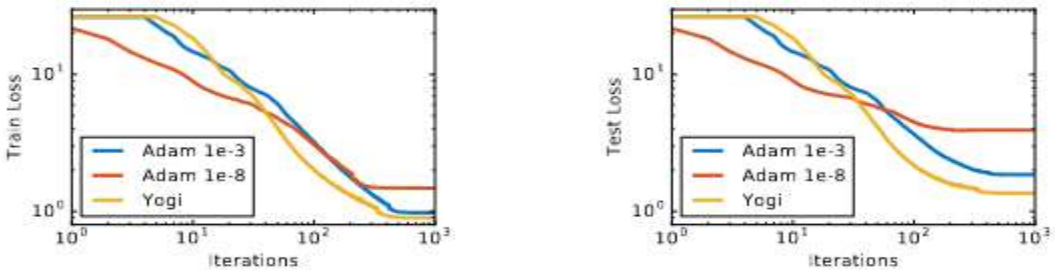

Figure 1- Effect of adaptivity level ϵ on training MNIST deep autoencoder[7]

EXPERIMENTAL SETUP

Based on the insights gained from our theoretical analysis, we now present empirical results showcasing three aspects of our framework: (i) the value gained by controlled variation in learning rate using YOGI, (ii) fast optimization, and (iii) wide applicability on several large-scale problems ranging from natural language processing to computer vision. We run our experiments on a commodity machine with Intel R© Xeon R© CPU E5-2630 v4 CPU, 256GB RAM, and 8 Nvidia R© Titan Xp GPU. We compare YOGI with highly tuned ADAM and the reported state-of-the-art results for the setup. Typically, for obtaining the state-of-the-art results extensive hyper-parameter tuning and carefully designed learning rate schedules are required (see e.g. Transformers. However, for YOGI, we restrict to tuning just two scalar hyper parameters the learning rate and β2. In all our experiments, YOGI and ADAM are always initialized from the same point. Initialization of mt and vt are also important for YOGI and ADAM. These are often initialized with 0 in conjunction with debiasing strategies. Even with additional debiasing, such an initialization can result in very large learning rates at start of the training procedure; thereby, leading to convergence and performance issues. Thus, for YOGI, we propose to initialize the vt based on gradient square evaluated at the initial point averaged over a (reasonably large) mini-batch. Decreasing learning rate is typically necessary for superior performance. To this end, we chose the simple learning rate schedule of reducing the learning rate by a constant factor when performance metric plateaus on the validation/test set (commonly known as ReduceLRonPlateau). Inspired from our theoretical analysis, we set a moderate value of ϵ = 10−3 in YOGI for all the

experiments in order to control the adaptivity. We will see that under such simple setup, YOGI still achieves similar or better performance compared to highly task-specific tuned setups.

(i) Deep AutoEncoder- For our first task, we train two deep autoencoder models called "CURVES" and "MNIST", typically used as standard benchmarks for neural network optimization. The "MNIST" autoencoder consists of an encoder with layers of size (28×28)- 1000-500-250-30 and a symmetric decoder, totaling in 2.8M parameters. The thirty units in the code layer were linear and all the other units were logistic. The data set contains images of handwritten digits 0–9. The pixel intensities were normalized to lie between 0 and 1. The "CURVES" autoencoder consists of an encoder with layers of size (28×28)-400-200-100- 50-25-6 and a symmetric decoder totaling in 0.85M parameters. The six units in the code layer were linear and all the other units were logistic. In "MNIST" autoencoder, we perform significantly better than all prior results including ADAM with specially tuned learning rate. We also observed similar gains in "CURVES" autoencoder with a smaller β2.

(ii) Named Entity Recognition (NER)- Finally, we test YOGI for sequence labeling task in NLP involving recurrent neural networks. We use the popular CNN-LSTM-CRF model for NER task. In this, multiple CNN filters of different widths are used to encode morphological and lexical information observed in characters. A word-level bidirectional LSTM layer models the long-range dependence structure in texts. A linear-chain CRF layer models the sequence-level data likelihood while inference is performed using Viterbi Algorithm. In our experiments, we use the BC5CDR biomedical data. The CNN-LSTM model comprises of 1400 CNN filters of widths, 300 dimensional pre-trained word embeddings, a single layer 256 dimensional bidirectional LSTM, and dropout probability of 0.5 applied to word embeddings and LSTM output layer. We use exact match to evaluate the F1 score of our approach. YOGI, again, performs better than ADAM and achieves better F1 score than the one previously reported.

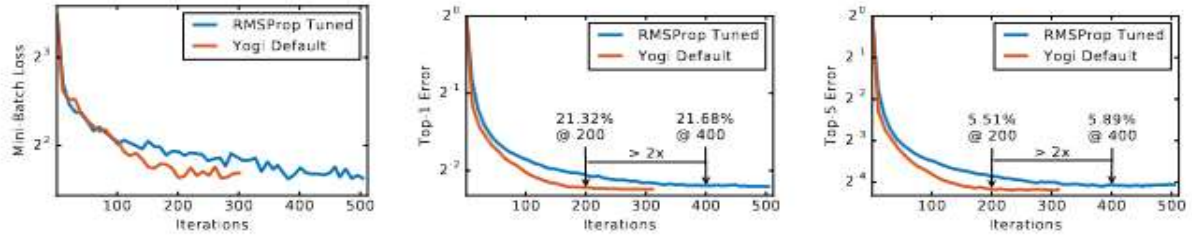

Figure 2- Comparison of highly tuned RMSProp optimizer with YOGI for Inception Resnet-v2 on Imagenet[4]

CONCLUSION

We discuss our observations about empirical results and suggest some practical guidelines for using YOGI. As we have shown, YOGI tends to perform superior to ADAM and highly hand-tuned SGD across various architectures and datasets. It is also interesting to note that typically YOGI has a smaller variance across different runs when compared to ADAM. Furthermore, very little hyperparameter tuning was used for YOGI to remain faithful to our goal. Inspired from our theoretical analysis, we fixed $\epsilon = 10-3$ and $\beta1 = 0.9$ in all experiments. We also observed that initial learning rate η in the range $[10-2, 10-3]$ and $\beta2 = \{0.9, 0.99, 0.999\}$ appears to work well in most settings. As a general remark, the learning rate for YOGI seems to be around $5 - 10$ times higher than that of ADAM. For decreasing the learning rate, we used the standard heuristic of ReduceLROnPlateau. In general, the patience of ReduceLROnPlateau scheduler will depend on data size, but in most experiments we saw a patience of 5,000-10,000 gradient steps works well

with a decay factor of 0.5. Finally, we would like to emphasize that other popular learning rate decay schedules different from ReduceLRonPlatueau can also be used for YOGI. Also, it is easy to deploy YOGI in existing pipelines to provide good performance with minimal tuning. To illustrate these points, we conducted an experiment on Inception-Resnet-v2, a high performing model on ImageNet 2012 classification challenge involving 1.28M images and 1,000 classes. In order to achieve best results, employed a heavily tuned RMSPROP optimizer with learning rate of 0.045, momentum decay of 0.9, $\beta = 0.9$, and $\epsilon = 1$.

REFERENCES

[1]. Naman Agarwal, Zeyuan Allen Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima for nonconvex optimization in linear time. CoRR, abs/1611.01146, 2022.

[2]. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. stat, 1050:21, 2023.

[3]. Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. 2022. arXiv:1802.04434.

[4]. Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization. CoRR, abs/1611.00756, 2023.

[5]. Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308, 2022.

[6]. Kevin Clark. Semi-supervised learning for nlp. http: // web. stanford. edu/ class/ cs224n/ lectures/ lecture17. pdf , 2023.

[7]. Sanjay Kumar Suman, Dhananjay Kumar and L. Bhagyalakshmi, "Non Cooperative Power Control Game with New Pricing for Wireless Ad hoc Networks", International Review on Computers and Software, vol. 9, no. 1, pp. 18-28, 2014. ISSN: 1828-6003,

[8]. S. Porselvi, Sanjay Kumar Suman and L. Bhagyalakshmi, "Harvesting RF energy for mobile charging", Australian Journal of Basic and Applied Science, vol. 9, no. 20, pp. 454-465, June 2015.

[9]. K. Swapna, P. Rajalakshmi and Sanjay Kumar Suman, "Security Enhancement in MANET using Game Theory", Middle East Journal of Scientific Research, vol. 23, pp. 190-195, 2015.

[10]. VinaySrivatsan, Sanjay Kumar Suman, L. Bhagyalakshmi and S. Porselvi, "Non radiative wireless power transfer", Journal of Advances in Natural and Applied Sciences, vol. 10, no. 16, pp. 147-153, Nov. 2016.

[11]. Sujeetha Devi, Bhagyalakshmi L and Sanjay Kumar Suman, "Cluster based energy efficient joint routing algorithm for delay minimization in wireless sensor networks", International Journal of Pure and Applied Mathematics, vol. 119, no. 15, 307-313, 2018

.