SURVEY OF ITERATE AVERAGING METHODS

Hava Chaptoukaev
MSc Data Science and Artificial Intelligence
Université Côte d'Azur
hava.chaptoukaev@etu.univ-cotedazur.fr

ABSTRACT

Deep Neural Networks (DNN) achieve state of the art performance in a wide range of problems. Given their impressive performance, researchers are particularly interested in their optimization, and one of the simplest optimization algorithms contributing in the success of DNN models is the Stochastic Gradient Descent (SGD). While its theoretical properties make it the ideal optimization approach for large scale applications, it has been observed that SGD suffers from the effect of noisy gradient estimates. In this project we are interested in iterate averaging methods as a noise reduction tools to improve SGD results.

1 INTRODUCTION

Theoretical results and extensive computational experience have led many in the machine learning community to view SGD as the ideal optimization approach for large-scale applications. However, since its inception it has been observed SGD suffers from the effect of noisy gradient estimates, and subsequently generates noisy iterate sequences that tend to oscillate around minima during the optimization process and keeps it from converging to the solution when fixed stepsizes are used. To overcome these limitations, noise reduction methods have been developed to reduce the errors of the gradient estimates. In this project, we consider iterate averaging methods, which have proved to be effective in practice, offer attractive theoretical results, and have a low computational cost. These methods accomplish noise reduction not by averaging gradient estimates, but by maintaining an average of iterates computed during the optimization, and it is this sequence that converges to the solution. First proposed by Ruppert (1) and Polyak (2), iterate averaging methods are based on a simple and natural intuition. The idea is to compute a sequence of iterate averages that would automatically possess less noisy behaviour. Precisely, when minimizing a continuously differentiable function F, the solution is approximated by an average of the iterate sequence $\{w_k\}$ given by

$$w_{k+1} = w_k - \alpha_k g(w_k, \xi) \tag{1}$$

where α_k is the stepsize or learning rate at iteration k, and $g(w_k, \xi)$ is the stochastic gradient. The optimized parameters of the model at iteration n are then given by

$$\hat{w}_n = \frac{1}{n} \sum_{k=1}^{n} w_k \tag{2}$$

In this scheme, the averaged sequence $\{\hat{w}_k\}$ has no effect on the computation of the SGD sequence $\{w_k\}$, and therefore the method does not interfere with the optimization process and is fast to compute. In addition, when using a stepsize diminishing with a rate of $\mathcal{O}(1/k)$ the average sequence has better convergence properties than the SGD iterates, and in particular averaging scenarios can make the effects of ill-conditioning disappear (3). Due to the method's simplicity coupled with the previous theoretical advantages, averaging has been incorporated into various optimization schemes. Examples include tail averaging (4), geometric averaging (5) or stochastic averaging (6), that we describe in next section.

2 ITERATE AVERAGING METHODS

2.1 TAIL AVERAGING

Motivated by the intuition that the first iterations of the gradient descent are usually far from the solution, Jain et al. (4) proposed the *Tail averaging* (or suffix averaging) scheme. This method performs an average on the weights of the model using only the most recent iterations. To do so, the authors introduce a hyperparameter τ, allowing to control the *memory* of the method. The averaged iterates are given by

$$\hat{w}_n = \frac{1}{n-\tau} \sum_{k=\tau}^{n} w_k \tag{3}$$

where $\tau \in [0, n]$. This method demonstrates attractive convergence properties for various learning rate scenarios, and effectively improves SGD variance properties.

2.2 GEOMETRIC AVERAGING

Based on the idea to consider a weighted average rather than an uniform average of the iterations, Neu and Rosasco (5) proposed a generalized version of Polyak-Ruppert averaging, they call *Geometric Polyak-Ruppert averaging*. As its name suggests, this method performs a geometric average of the model's parameters and outputs

$$\hat{w}_n = \frac{1}{\sum_{k=1}^{n}(1-\alpha\lambda)^k} \cdot \sum_{k=1}^{n}(1-\alpha\lambda)^k w_k \tag{4}$$

after iteration n, where α is the learning rate, and $\lambda \in [0, 1/\alpha)$ is an hyperparameter of the model. Unlike the tail averaging method, we can see from the equation above that geometric averaging puts a higher weight on initial iterations, and more precisely attributes weights decaying in a geometric pattern to the model parameters. We can also notice that setting $\lambda = 0$ is exactly equivalent to using the Polyak-Ruppert averaging scheme. In addition, (5) shows geometric averaging is equivalent in convergence to adding a Tikhonov regularisation term to the loss function, but is less computationally costly.

2.3 STOCHASTIC AVERAGING

Izmailov et al. (5) proposed Stochastic Weights Averaging (SWA) as a simple procedure that improves generalization in deep learning over SGD at no additional cost, and has been shown to significantly improve generalization in computer vision tasks and notably on ImageNet and CIFAR benchmarks. SWA performs an average of SGD iterates with a modified cyclical or a high constant learning rate, and exploits the flatness of loss functions specific to deep learning for improved generalization. The idea is the following: at each iteration k, we check if the optimization process has reached the end of the learning rate cycle c by computing $mod(k, c)$. If $mod(k, c) = 0$, then we set $n_{models} = k/c$, and the averaged weight is given by

$$\hat{w}_{k+1} = \frac{\hat{w}_k \cdot n_{models} + w_k}{n_{models} + 1} \tag{5}$$

This suggests the averaged sequences are only computed at each end of one learning rate cycle. In this project we focus on the case of the constant learning rate only. In this scenario, as $c = 1$, the averaged sequences $\{\hat{w}_k\}$ are computed at each step and simply become

$$\hat{w}_{k+1} = \frac{\hat{w}_k \cdot k + w_k}{k + 1} \tag{6}$$

While SGD converges to a solution within a wide flat region of loss, SWA should overcome this issue by averaging multiple SGD solutions, allowing it to move towards the center of the flat region.

Table 1: Comparison of the averaging schemes

Method	**Accuracy**
No averaging	51.2 ± 2.7
Polyak-Ruppert averaging	57.1 ± 0.8
Tail averaging	65.8 ± 2.3
Geometric averaging ($\lambda \neq 0$)	54.9 ± 0.9
Stochastic averaging	$56.\ 5 \pm 0.3$

3 Experiments and Results

3.1 Experiment Proposal

In order to assess the performances of the different averaging schemes, we propose to compare them on an unique problem. We chose to perform classification on the CIFAR-10 dataset. The CIFAR-10 dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research. The dataset contains $60,000$ 32×32 color images in 10 different classes representing airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. To compare the averaging schemes between them, we train the model for only 10 epochs and compare the different accuracies obtained on the test dataset with each of the averaged parameter sets. To do, we simply implement a small Convolutional Neural Network (CNN), with 2 convolutional layers and 3 fully connected layers. We use a cross-entropy loss function in combination with a softmax for the layer of our CNN for classification, we use a SGD optimizer with a fixed learning rate of 0.1, and a batch size of 124. In addition, the hyperparameters of each methods are selected using cross-validation.

3.2 Analysis of the Results

The results of the experiment are reported in Table 1. While we can clearly see all the averaging methods considerably improve the results of SGD, in our application the tail averaging performs remarkably better than other models. Indeed, the accuracy on the test set goes from 51.2% using the SGD estimates to 65.8% using fine tuned tail averaging in only 10 epochs. Geometric averaging on

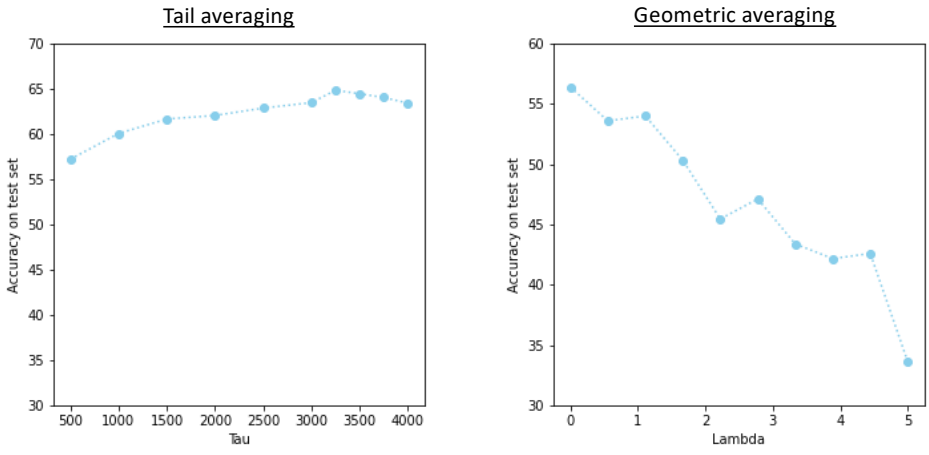

Figure 1: (Left) Accuracy on the test dataset is at its best using tail averaging for $\tau = 3250$, i.e. the method only keeps the 780 last iterations out of 4030. (Right) Accuracy using geometric averaging is at its best for $\lambda = 0$, i.e. when the method comes back to the basic Polyak-Ruppert method. Second best accuracy is reached for $\lambda = 2$.

the other hand, performs rather poorly in comparison. Figure 1 illustrates the performances of tail averaging and geometric averaging with respect to their hyperparameters. Tail averaging performs best for a $\tau = 3250$, meaning the method only keeps in memory the $n - \tau$ last iterations for the averaging. It seems like getting rid of the first iterations, which are usually far from the solution, improves the performance of SGD, and reduces its variability. In this sense, it seems natural that geometric averaging, which inversely gives less importance to recent iterations, does not perform as well. In this particular case, we can see the best value for λ is 0, meaning geometric averaging performs at best as well as the original Polyak-Ruppert averaging method. Both methods behave in complementary ways and from their respective definitions, we could imagine geometric averaging is more suited for a small number of iteration while tail averaging performs better for large values of n. Stochastic averaging also considerably improves the performance of SGD, but we can easily imagine given the nature of the algorithm, the method is not exploited at its full capacity during a training of 10 epochs.

4 CONCLUSION

In this project, we reviewed the main existing iterate averaging schemes. We tested the methods on a unique problem of image classification. In our case, tail averaging demonstrated remarkable results, and brought a significant improvement to the SGD estimates. As the method gets rid of the first flawed iterations, it seems natural that it yields better results than the classical scheme. Although both stochastic and geometric averaging display convincing results, they do not outperform tail averaging. However they might become assets when increasing the number of epochs and once the loss has the time to converge, as both methods are averaging schemes that help for generalisation. However, most of the theoretical convergence results of iterate averaging are achieved in the case of adaptative learning rates. The next step of this analysis should be testing the various schemes with such learning rates.

REFERENCES

Ruppert, D. (1988). Efficient estimations from a slowly convergent Robbins-Monro process. Cornell University Operations Research and Industrial Engineering.

Polyak, B. T. (1990). New stochastic approximation type procedures. *Automat. i Telemekh*, 7(98-107), 2.

Bottou, L., Curtis, F. E., Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2), 223-311.

Jain, P., Kakade, S., Kidambi, R., Netrapalli, P., Sidford, A. (2018). Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18.

Neu, G., Rosasco, L. (2018, July). Iterate averaging as regularization for stochastic gradient descent. In *Conference On Learning Theory* (pp. 3222-3242). PMLR.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *arXiv preprint* arXiv:1803.05407.