Optimistic Acceleration for Adaptive Optimization

Anonymous Author(s)
Affiliation
Address
email

1 Experiments

In this section, we provide experiments on classification tasks with various neural network architectures and datasets to demonstrate the effectiveness of OPTIMISTIC-AMSGRAD in practical applications.

Methods. We consider two baselines. The first one is the original AMSGRAD. The hyper-parameters are set to be β_1 and β_2 to be 0.9 and 0.999 respectively, as recommended by [**?**]. We tune the learning rate η over a fine grid and report the best result.

The other competing method is the aforementioned OPTIMISTIC-ADAM+$\hat{v}_t$ method (Algorithm **??**) as in Section 3. The key difference is that it uses previous gradient as the gradient prediction of the next iteration. We also report the best result achieved by tuning the step size η for OPTIMISTIC-ADAM+$\hat{v}_t$.

For OPTIMISTIC-AMSGRAD, we use the same β_1, β_2 and the best step size η of AMSGRAD for a fair evaluation of the improvement brought by the extra optimistic step. Yet, OPTIMISTIC-AMSGRAD has an additional parameter r that controls the number of previous gradients used for gradient prediction. Fortunately, we observe similar performance of OPTIMISTIC-AMSGRAD with different values of r. Hence, we report $r = 5$ for now when comparing with other baselines. We will address on the choice of r at the end of this section.

In all experiments, all the optimization algorithms are initialized at the same point. We report the results averaged over 5 repetitions.

Datasets. Following [**?**] and [**?**], we compare different algorithms on *MNIST*, *CIFAR10*, *CIFAR100*, and *IMDB* datasets. For *MNIST*, we use two noisy variants named as 1.65*MNIST-back-rand* and 1.65*MNIST-back-image* from [**?**]. They both have 12000 training samples and 50000 test samples, where random background is inserted to the original *MNIST* hand written digit images. For *MNIST-back-rand*, each image is inserted with a random background, whose pixel values generated uniformly from 0 to 255, while *MNIST-back-image* takes random patches from a black and white as noisy background. The input dimension is 784 (28×28) and the number of classes is 10. *CIFAR10* and *CIFAR100* are popular computer-vision datasets consisting of 50000 training images and 10000 test images, of size 32×32. The number of classes are 10 and 100, respectively. The *IMDB* movie review dataset is a binary classification dataset with 25000 training and testing samples respectively. It is a popular datasets for text classification.

Network architecture. We adopt a multi-layer fully-connected neural network with input layer followed by a hidden layer with 200 nodes, which is connected to another layer with 100 nodes before the output layer. The activation function is ReLU for hidden layers, and softmax for the output layer. This network is tested on *MNIST* variants. Since convolutional networks are popular for image classification tasks, we consider an ALL-CNN architecture proposed by [**?**], which is constructed with several convolutional blocks and dropout layers. In addition, we also apply residual networks, Resnet-18 and Resnet-50 [**?**], which have achieved many state-of-the-art results. For the texture *IMDB* dataset, we consider training a Long-Short Term Memory (LSTM) network. The

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

network includes a word embedding layer with 5000 input entries representing most frequent words in the dataset, and each word is embedded into a 32 dimensional space. The output of the embedding layer is passed to 100 LSTM units, which is then connected to 100 fully connected ReLu's before the output layer. For all the models, we use cross-entropy loss. A mini-batch size of 128 is used to compute the stochastic gradients.

Results. Firstly, to illustrate the acceleration effect of OPTIMISTIC-AMSGRAD at early stage, we provide the training loss against number of iterations in Figure 1. We clearly observe that on all datasets, the proposed OPTIMISTIC-AMSGRAD converges faster than the other competing methods, right after the training begins. In other words, we need fewer iterations (samples) to achieve the same training loss. This validates one of the main advantages of OPTIMISTIC-AMSGRAD, which is a higher sample efficiency.

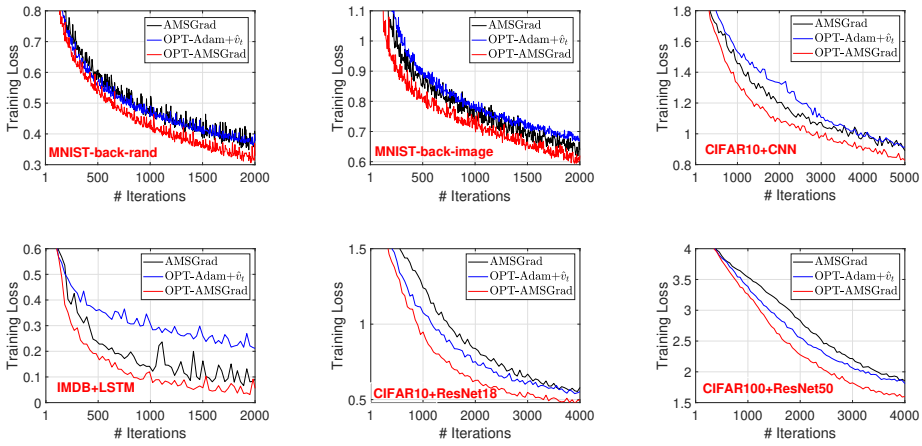

Figure 1: Training loss vs. Number of iterations. The first row are results with fully-connected neural network.

We are also curious about the long-term performance and generalization of the proposed method in test phase. In Figure **??**, Figure 2 and Figure **??**, we plot the corresponding results when the model is trained to the state with stable test accuracy. We observe: 1) In the long term, OPTIMISTIC-AMSGRAD algorithm may converge to a better point with smaller objective function value, and 2) In this three applications, the proposed OPTIMISTIC-AMSGRAD also outperforms the competing methods in terms of test accuracy. These are also important benefits of OPTIMISTIC-AMSGRAD.

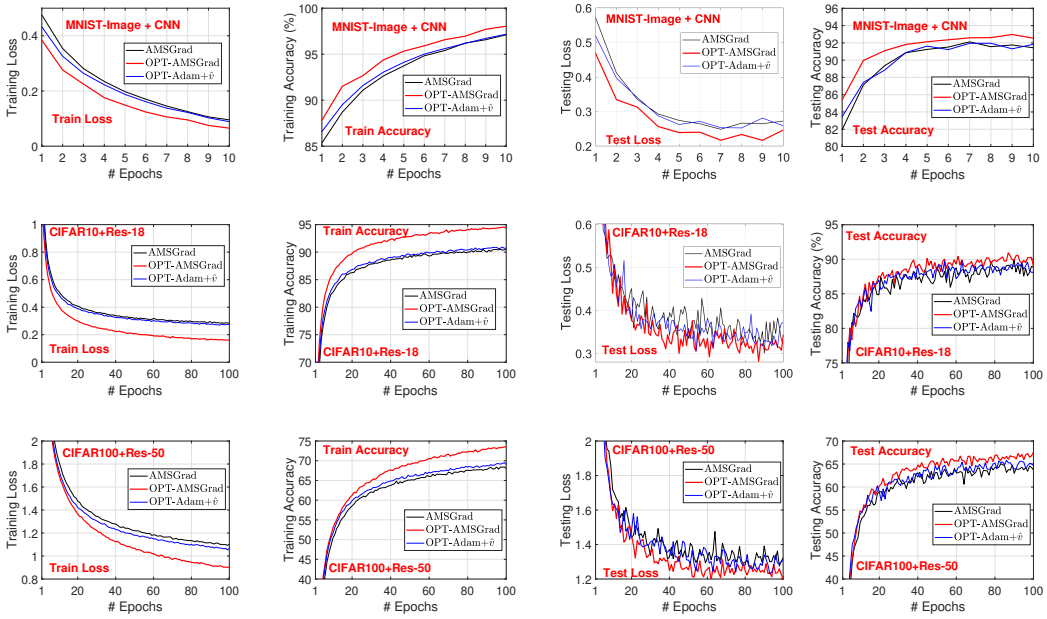

Figure 2: *MNIST-back-image* + CNN, *CIFAR10* + Res-18 and *CIFAR100* + Res-50 . We compare three methods in terms of training (cross-entropy) loss, training accuracy, testing loss, and testing accuracy. We observe that OPTIMISTIC-AMSGRAD consistently improves the two baselines.

1.1 Choice of parameter r

Recall that our proposed algorithm has the parameter r that governs the use of past information. Figure 3 compares the performance under different values of $r = 3, 5, 10$ on two datasets. From the result we see that the choice of r does not have significant impact on learning performance. Taking into consideration both quality of gradient prediction and computational cost, it appears that $r = 5$ is a good choice. We remark that empirically, the performance comparison among $r = 3, 5, 10$ is not absolutely consistent (i.e. more means better) in all cases. One possible reason is that for deep neural nets which have very complicated and highly non-convex landscape, using gradient information from too long ago may not be helpful in accurate gradient prediction. Nevertheless, $r = 5$ seems to be good for most applications.

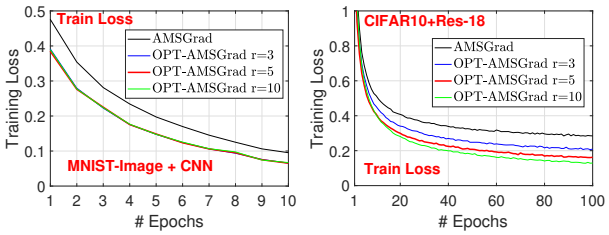

Figure 3: The training loss of OPTIMISTIC-AMSGRAD with different r.

2 Concluding Remarks

2.1 Discussion on the iteration cost

We observe that the iteration cost (i.e., actual running time per iteration) of our implementation of OPTIMISTIC-AMSGRAD with $r = 5$ is roughly two times larger than the standard AMSGRAD. When $r = 3$, the cost is roughly 0.7 times longer. Nevertheless, OPTIMISTIC-AMSGRAD may still be beneficial in terms of training efficiency, since fewer iterations are typically needed. For

example, in Figure 2 and **??**, to reach the training loss of AMSGRAD at 100 epochs, the proposed method only needs roughly 20 and 40 epochs, respectively. That said, OPTIMISTIC-AMSGRAD needs 40% and 80% time to achieve same training loss as AMSGrad, in this two problems.

The computational overhead mostly comes from the gradient extrapolation step. More specifically, recall that the extrapolation step consists of: (a) The step of constructing the linear system $(U^\top U)$. The cost of this step can be optimized and reduced to $\mathcal{O}(d)$, since the matrix U only changes one column at a time. (b) The step of solving the linear system. The cost of this step is $O(r^3)$, which is negligible as the linear system is very small (5-by-5 if $r = 5$). (c) The step that outputs an estimated gradient as a weighted average of previous gradients. The cost of this step is $\mathcal{O}(r \times d)$. Thus, the computational overhead is $\mathcal{O}\left((r+1)d + r^3\right)$. Yet, we notice that step (a) and (c) is parallelizable, so they can be accelerated in practice.

Memory usage: Our algorithm needs a storage of past r gradients for each coordinate, in addition to the estimated second moments and the moving average. Though it seems demanding compared to the standard AMSGrad, it is relatively cheap compared to Natural gradient method (e.g., [**?**]), as Natural gradient method needs to store some matrix inverse.

2.2 Conclusion

In this paper, we propose OPTIMISTIC-AMSGRAD, which combines optimistic learning and AMSGRAD to improve sampling efficiency and accelerate the process of training, in particular for deep neural networks. With a good gradient prediction, the regret can be smaller than that of standard AMSGRAD. Experiments on various deep learning problems demonstrate the effectiveness of the proposed method in improving the training efficiency.