OPT-AMSGrad: An Optimistic Acceleration of AMSGrad for Nonconvex Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider a new variant of AMSGrad. AMSGrad Reddi et al. [2018] is a popular adaptive gradient based optimization algorithm that is widely used in training deep neural networks. The new variant assumes that mini-batch gradients in consecutive iterations have some underlying structure, which makes the gradients sequentially predictable. By exploiting the predictability and some ideas from Optimistic Online Learning, the proposed algorithm can accelerate the convergence, increase sample efficiency and also enjoys a tighter non-asymptotic bound under some conditions. We conduct experiments on several deep learning models, which show that by improving sample efficiency, the proposed method achieves a convergence speedup in practice.

1 Introduction

Nowadays deep learning has been very successful in several applications, from robotics (e.g. Levine et al. [2017]), computer vision (e.g. He et al. [2016], Goodfellow et al. [2014]), reinforcement learning (e.g. Mnih et al. [2013]), to natural language processing (e.g. Graves et al. [2013]). A common goal in these applications is learning quickly. It becomes a desired goal due to the presence of big data and/or the use of large neural nets. To accelerate the process, there are number of algorithms proposed in recent years, such as AMSGRAD Reddi et al. [2018], ADAM Kingma and Ba [2015], RMSPROP Tieleman and Hinton [2012], ADADELTA Zeiler [2012], and NADAM Dozat [2016], etc.

All the prevalent algorithms for training deep nets mentioned above combine two ideas: the idea of adaptivity from ADAGRAD Duchi et al. [2011], McMahan and Streeter [2010] and the idea of momentum from NESTEROV'S METHOD Nesterov [2004] or HEAVY BALL method Polyak [1964]. ADAGRAD is an online learning algorithm that works well compared to the standard online gradient descent when the gradient is sparse. Its update has a notable feature: the learning rate is different for each dimension, depending on the magnitude of gradient in each dimension, which might help in exploiting the geometry of data and leading to a better update. On the other hand, NESTEROV'S METHOD or HEAVY BALL Method Polyak [1964] is an accelerated optimization algorithm whose update not only depends on the current iterate and current gradient but also depends on the past gradients (i.e. momentum). State-of-the-art algorithms like AMSGRAD Reddi et al. [2018] and ADAM Kingma and Ba [2015] leverage this ideas to accelerate the training process of neural nets.

In this paper, we propose an algorithm that goes further than the hybrid of the adaptivity and momentum approach. Our algorithm is inspired by OPTIMISTIC ONLINE LEARNING Chiang et al. [2012], Rakhlin and Sridharan [2013], Syrgkanis et al. [2015], Abernethy et al. [2018], which assumes that a good guess of the loss function in each round of online learning is available, and plays an action by exploiting the guess. By exploiting the guess, algorithms in OPTIMISTIC ONLINE LEARNING can enjoy smaller regret than the ones without exploiting the guess. We combine the OPTIMISTIC

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

ONLINE LEARNING idea with the adaptivity and the momentum ideas to design a new algorithm — OPTIMISTIC-AMSGRAD. To the best of our knowledge, this is the first work exploring towards this direction. The proposed algorithm not only adapts to the informative dimensions, exhibits momentum, but also exploits a good guess of the next gradient to facilitate acceleration. Besides theoretical analysis of OPTIMISTIC-AMSGRAD, we also conduct experiments and show that the proposed algorithm not only accelerates convergence of loss function, but also leads to better generalization performance in some cases.

2 Preliminaries

We begin by providing some background in online learning, as we use some tools from it to design and analyze the proposed algorithm. We follow the notations in related adaptive optimization papers Kingma and Ba [2015], Reddi et al. [2018]. For any vector u, $v \in \mathbb{R}^d$, u/v represents element-wise division, u^2 represents element-wise square, $\sqrt{u}$ represents element-wise square-root. We denote $g_{1:T}[i]$ as the sum of the i_{th} element of T vectors $g_1, g_2, \ldots, g_T \in \mathbb{R}^d$.

2.1 Optimistic Online learning

The standard setup of ONLINE LEARNING is that, in each round t, an online learner selects an action $w_t \in \mathcal{K} \subseteq \mathbb{R}^d$, then the learner observes $\ell_t(\cdot)$ and suffers loss $\ell_t(w_t)$ after the learner commits the action. The goal of the learner is minimizing the regret,

$$\text{Regret}_T(\{w_t\}) := \sum_{t=1}^{T} \ell_t(w_t) - \sum_{t=1}^{T} \ell_t(w^*),$$

which is the cumulative loss of the learner minus the cumulative loss of some benchmark $w^* \in \mathcal{K}$.

The idea of OPTIMISTIC ONLINE LEARNING (e.g. Chiang et al. [2012], Rakhlin and Sridharan [2013], Syrgkanis et al. [2015], Abernethy et al. [2018]) is as follows. Suppose that, in each round t, the learner has a good guess $m_t(\cdot)$ of the loss function $\ell_t(\cdot)$ before playing an action w_t. Then, the learner should exploit the guess $m_t(\cdot)$ to choose an action w_t since $m_t(\cdot)$ is close to the true loss function $\ell_t(\cdot)$. [1] For example, Syrgkanis et al. [2015] proposes an optimistic-variant of FOLLOW-THE-REGULARIZED-LEADER (FTRL). FTRL (see e.g. Hazan [2016]) is an online learning algorithm whose update is

$$w_t = \arg\min_{w \in \mathcal{K}} \langle w, L_{t-1} \rangle + \tfrac{1}{\eta} R(w), \tag{1}$$

where η is a parameter, $R(\cdot)$ is a 1-strongly convex function with respect to a norm ($\|\cdot\|$) on the constraint set $\mathcal{K}$, and $L_{t-1} := \sum_{s=1}^{t-1} g_s$ is the cumulative sum of gradient vectors of the loss functions (i.e. $g_s := \nabla \ell_s(w_s)$) up to but not including t. FTRL has regret at most $O(\sqrt{\sum_{t=1}^{T} \|g_t\|_*})$. On the other hand, OPTIMISTIC-FTRL Syrgkanis et al. [2015] has update

$$w_t = \arg\min_{w \in \mathcal{K}} \langle w, L_{t-1} + m_t \rangle + \tfrac{1}{\eta} R(w), \tag{2}$$

where m_t is the learner's guess of the gradient vector $g_t := \nabla \ell_t(w_t)$. Under the assumption that loss functions are convex, the regret of OPTIMISTIC-FTRL is at most $O(\sqrt{\sum_{t=1}^{T} \|g_t - m_t\|_*})$, which can be much smaller than the regret of FTRL if m_t is close to g_t. Consequently, OPTIMISTIC-FTRL can achieve better performance than FTRL. On the other hand, if m_t is far from g_t, then the regret of OPTIMISTIC-FTRL would be only a constant factor worse than that of its counterpart FTRL. In Section 4, we will provide a way to get m_t. Now we just want to emphasize the importance of leveraging a good guess m_t for updating w_t in order to get a fast convergence rate (or equivalently, small regret). We will have a similar argument when we compare OPTIMISTIC-AMSGRAD and AMSGRAD.

2.2 Adaptive optimization methods

Recently, adaptive optimization has been popular in various deep learning applications due to their superior empirical performance. ADAM Kingma and Ba [2015] is a very popular adaptive algorithm

[1] Imagine that if the learner would had been known $\ell_t(\cdot)$ before committing its action, then it would exploit the knowledge to determine its action and consequently minimizes the regret.

for training deep nets. It combines the momentum idea Polyak [1964] with the idea of ADAGRAD Duchi et al. [2011], which has different learning rates for different dimensions, adaptive to the learning process. More specifically, the learning rate of ADAGRAD in iteration t for a dimension j is proportional to the inverse of $\sqrt{\Sigma_{s=1}^{t} g_s[j]^2}$, where $g_s[j]$ is the j-th element of the gradient vector g_s at time s. This adaptive learning rate might help for accelerating the convergence when the gradient vector is sparse Duchi et al. [2011]. However, when applying ADAGRAD to train deep nets, it is observed that the learning rate might decay too fast Kingma and Ba [2015]. Therefore, Kingma and Ba [2015] proposes using a moving average of gradients divided by the square root of the second moment of the moving average (element-wise fashion), for updating the model parameter w (i.e. line 5,6 and line 8 of Algorithm 1). Yet, ADAM Kingma and Ba [2015] fails at some online convex optimization problems. AMSGRAD Reddi et al. [2018] fixes the issue. The algorithm of AMSGRAD is shown in Algorithm 1. The difference between ADAM and AMSGRAD lies on line 7 of Algorithm 1. ADAM does not have the max operation on line 7 (i.e. $\hat{v}_t = v_t$ for ADAM) while Reddi et al. [2018] adds the operation to guarantee a non-increasing learning rate, $\frac{\eta_t}{\sqrt{\hat{v}_t}}$, which helps for the convergence (i.e. average regret $\frac{\text{Regret}_T}{T} \to 0$). For the hyper-parameters of AMSGRAD, it is suggested in Reddi et al. [2018] that $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

Algorithm 1 AMSGRAD Reddi et al. [2018]

1: Required: parameter β_1, β_2, and η_t.
2: Init: $w_1 \in \mathcal{K} \subseteq \mathbb{R}^d$ and $v_0 = \epsilon 1 \in \mathbb{R}^d$.
3: **for** $t = 1$ to T **do**
4: Get mini-batch stochastic gradient vector g_t at w_t.
5: $\theta_t = \beta_1 \theta_{t-1} + (1-\beta_1) g_t$.
6: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$.
7: $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$.
8: $w_{t+1} = w_t - \eta_t \frac{\theta_t}{\sqrt{\hat{v}_t}}$. (element-wise division)
9: **end for**

3 OPTIMISTIC-AMSGRAD

We propose a new optimization algorithm, OPTIMISTIC-AMSGRAD, shown in Algorithm 2. It combines the idea of adaptive optimization with optimistic learning. In each iteration, the learner computes a gradient vector $g_t := \nabla \ell_t(w_t)$ at w_t (line 4), then it maintains an exponential moving average of $\theta_t \in \mathbb{R}^d$ (line 5) and $v_t \in \mathbb{R}^d$ (line 6), which is followed by the max operation to get $\hat{v}_t \in \mathbb{R}^d$ (line 7). The learner also updates an auxiliary variable $w_{t+\frac{1}{2}} \in \mathcal{K}$ (line 8). It uses the auxiliary variable (hidden model) to update and commit w_{t+1} (line 9), which exploits the guess m_{t+1} of g_{t+1} to get w_{t+1}. As the learner's action set is $\mathcal{K} \subseteq \mathbb{R}^d$, we adopt the notation $\Pi_{\mathcal{K}}[\cdot]$ for the projection to $\mathcal{K}$ if needed. The scheme of AMSGRAD is summarized in Figure 1.

Algorithm 2 OPTIMISTIC-AMSGRAD

1: Required: parameter β_1, β_2, ϵ, and η_t.
2: Init: $w_1 = w_{-1/2} \in \mathcal{K} \subseteq \mathbb{R}^d$ and $v_0 = \epsilon 1 \in \mathbb{R}^d$.
3: **for** $t = 1$ to T **do**
4: Get mini-batch stochastic gradient g_t at w_t.
5: $\theta_t = \beta_1 \theta_{t-1} + (1-\beta_1) g_t$.
6: $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t - m_t)^2$.
7: $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$.
8: $w_{t+\frac{1}{2}} = \Pi_{\mathcal{K}}\left[w_{t-\frac{1}{2}} - \eta_t \frac{\theta_t}{\sqrt{\hat{v}_t}}\right]$.
9: $w_{t+1} = \Pi_{\mathcal{K}}\left[w_{t+\frac{1}{2}} - \eta_{t+1} \frac{h_{t+1}}{\sqrt{\hat{v}_t}}\right]$,
where $h_{t+1} := \beta_1 \theta_{t-1} + (1-\beta_1) m_{t+1}$
and m_{t+1} is the guess of g_{t+1}.
10: **end for**

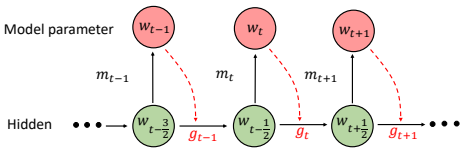

Figure 1: Scheme of OPTIMISTIC-AMSGRAD.

We see that the proposed OPTIMISTIC-AMSGRAD inherits three properties:

- Adaptive learning rate of each dimension as ADAGRAD Duchi et al. [2011]. (line 6, line 8 and line 9)

- Exponential moving average of the past gradients as NESTEROV'S METHOD Nesterov [2004] and the HEAVY-BALL method Polyak [1964]. (line 5)
- Optimistic update that exploits a good guess of the next gradient vector as optimistic online learning algorithms Chiang et al. [2012], Rakhlin and Sridharan [2013], Syrgkanis et al. [2015]. (line 9)

The first property helps for acceleration when the gradient has a sparse structure. The second one is from the well-recognized idea of momentum which can also help for acceleration. The last one, perhaps less known outside the ONLINE LEARNING community, can actually lead to acceleration when the prediction of the next gradient is good. This property will be elaborated in the following subsection in which we provide the theoretical analysis of OPTIMISTIC-AMSGRAD.

Observe that the proposed algorithm does not reduce to AMSGRAD when $m_t = 0$. Furthermore, if $\mathcal{K} = \mathbb{R}^d$ (unconstrained case), one might want to combine line 8 and line 9 and get a single line as $w_{t+1} = w_{t-\frac{1}{2}} - \eta_t \frac{\theta_t}{\sqrt{\hat{v}_t}} - \eta_{t+1}\frac{h_{t+1}}{\sqrt{\hat{v}_t}}$. Yet, based on this expression, we see that w_{t+1} is updated from $w_{t-\frac{1}{2}}$ instead of w_t. Therefore, while OPTIMISTIC-AMSGRAD looks like just doing an additional update compared to AMSGRAD, the difference of the updates is subtle. In the following analysis, we show that the interleaving actually leads to some cancellation in the regret bound.

3.1 Theoretical analysis

In this section, we provide regret analysis of the proposed method and show that it may improve the bound of vanilla AMSGRAD with good guess of gradient.

Notations. To begin with, let us introduce some notations first. We denote the Mahalanobis norm $\|\cdot\|_H := \sqrt{\langle \cdot, H\cdot\rangle}$ for some PSD matrix H. We let $\psi_t(x) := \langle x, \text{diag}\{\hat{v}_t\}^{1/2}x\rangle$ for a PSD matrix $H_t^{1/2} := \text{diag}\{\hat{v}_t\}^{1/2}$, where $\text{diag}\{\hat{v}_t\}$ represents the diagonal matrix whose i_{th} diagonal element is $\hat{v}_t[i]$ in Algorithm 2. We define its corresponding Mahalanobis norm $\|\cdot\|_{\psi_t} := \sqrt{\langle \cdot, \text{diag}\{\hat{v}_t\}^{1/2}\cdot\rangle}$, where we abuse the notation ψ_t to represent the PSD matrix $H_t^{1/2} := \text{diag}\{\hat{v}_t\}^{1/2}$. Consequently, $\psi_t(\cdot)$ is 1-strongly convex with respect to the norm $\|\cdot\|_{\psi_t} := \sqrt{\langle \cdot, \text{diag}\{\hat{v}_t\}^{1/2}\cdot\rangle}$. Namely, $\psi_t(\cdot)$ satisfies $\psi_t(u) \geq \psi_t(v) + \langle \psi_t(v), u - v\rangle + \frac{1}{2}\|u-v\|_{\psi_t}^2$ for any point u, v. A consequence of 1-strongly convexity of $\psi_t(\cdot)$ is that $B_{\psi_t}(u, v) \geq \frac{1}{2}\|u - v\|_{\psi_t}^2$, where the Bregman divergence $B_{\psi_t}(u, v)$ is defined as $B_{\psi_t}(u, v) := \psi_t(u) - \psi_t(v) - \langle \psi_t(v), u - v\rangle$ with $\psi_t(\cdot)$ as the distance generating function. We can also define the corresponding dual norm $\|\cdot\|_{\psi_t^*} := \sqrt{\langle \cdot, \text{diag}\{\hat{v}_t\}^{-1/2}\cdot\rangle}$.

Non-asymptotic analysis. We establish in this section a finite-time upper bound of the expected squared norm of the gradient of the objective function we are optimizing. The objective function for most deep learning task reads as follows:

$$\min_{w\in\Theta} f(w) := \frac{1}{n}\sum_{i=1}^{n} f(w, \xi_i) \tag{3}$$

where $(\xi_i, i \in [1, n])$ is the vector of n input data.

3.2 Comparison to some related methods

Comparison to nonconvex optimization works. Recently, Zaheer et al. [2018], Chen et al. [2019a], Ward et al. [2019], Zhou et al. [2018], Zou and Shen [2018], Li and Orabona. [2019] provide some theoretical analysis of ADAM-type algorithms when applying them to smooth non-convex optimization problems. For example, Chen et al. [2019a] provides a bound, which is $\min_{t\in[T]} \mathbb{E}[\|\nabla f(w_t)\|^2] = O(\log T/\sqrt{T})$. Yet, this data independent bound does not show any advantage over standard stochastic gradient descent. Similar concerns appear in other papers.

To get some adaptive data dependent bound (e.g. bounds like (**??**) or (**??**) that are in terms of the gradient norms observed along the trajectory) when applying OPTIMISTIC-AMSGRAD to nonconvex optimization, one can follow the approach of Agarwal et al. [2019] or Chen et al. [2019b]. They provide ways to convert algorithms with adaptive data dependent regret bound for convex loss functions (e.g. ADAGRAD) to the ones that can find an approximate stationary point of non-convex

loss functions. Their approaches are modular so that simply using OPTIMISTIC-AMSGRAD as the base algorithm in their methods will immediately lead to a variant of OPTIMISTIC-AMSGRAD that enjoys some guarantee on nonconvex optimization. The variant can outperform the ones instantiated by other ADAM-type algorithms when the gradient prediction m_t is close to g_t. The details are omitted since this is a straightforward application.

Comparison to AO-FTRL Mohri and Yang [2016]. In Mohri and Yang [2016], the authors propose AO-FTRL, which has the update of the form $w_{t+1} = \arg\min_{w\in\mathcal{K}}(\sum_{s=1}^{t} g_s)^\top w + m_{t+1}^\top w + r_{0:t}(w)$, where $r_{0:t}(\cdot)$ is a 1-strongly convex loss function with respect to some norm $\|\cdot\|_{(t)}$ that may be different for different iteration t. Data dependent regret bound was provided in the paper, which is $r_{0:T}(w^*) + \sum_{t=1}^{T} \|g_t - m_t\|_{(t)^*}$ for any benchmark $w^* \in \mathcal{K}$. We see that if one selects $r_{0:t}(w) := \langle w, \text{diag}\{\hat{v}_t\}^{1/2} w\rangle$ and $\|\cdot\|_{(t)} := \sqrt{\langle \cdot, \text{diag}\{\hat{v}_t\}^{1/2} \cdot\rangle}$, then the update might be viewed as an optimistic variant of ADAGRAD. However, no experiments was provided in Mohri and Yang [2016].

Comparison to OPTIMISTIC-ADAM Daskalakis et al. [2018]. We are aware that Daskalakis et al. [2018] proposed one version of optimistic algorithm for ADAM, which is called OPTIMISTIC-ADAM in their paper. A slightly modified version is summarized in Algorithm 3. Here, OPTIMISTIC-ADAM+$\hat{v}_t$ is OPTIMISTIC-ADAM in Daskalakis et al. [2018] with the additional max operation $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$ to guarantee that the weighted second moment is monotone increasing.

Algorithm 3 OPTIMISTIC-ADAM DASKALAKIS ET AL. [2018]+$\hat{v}_t$.

Required: parameter β_1, β_2, and η_t.
Init: $w_1 \in \mathcal{K}$ and $\hat{v}_0 = v_0 = \epsilon 1 \in \mathbb{R}^d$.
for $t = 1$ to T **do**
  Get mini-batch stochastic gradient vector $g_t \in \mathbb{R}^d$ at w_t.
  $\theta_t = \beta_1 \theta_{t-1} + (1-\beta_1) g_t$.
  $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$.
  $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$.
  $w_{t+1} = \Pi_k[w_t - 2\eta_t \frac{\theta_t}{\sqrt{\hat{v}_t}} + \eta_t \frac{\theta_{t-1}}{\sqrt{\hat{v}_{t-1}}}]$.
end for

We want to emphasize that the motivations are different. OPTIMISTIC-ADAM in their paper is designed to optimize two-player games (e.g. GANs Goodfellow et al. [2014]), while the proposed algorithm in this paper is designed to accelerate optimization (e.g. solving empirical risk minimization quickly). Daskalakis et al. [2018] focuses on training GANs Goodfellow et al. [2014]. GANs is a two-player zero-sum game. There have been some related works in OPTIMISTIC ONLINE LEARNING like Chiang et al. [2012], Rakhlin and Sridharan [2013], Syrgkanis et al. [2015]) showing that if both players use some kinds of OPTIMISTIC-update, then accelerating the convergence to the equilibrium of the game is possible. Daskalakis et al. [2018] was inspired by these related works and showed that OPTIMISTIC-MIRROR-DESCENT can avoid the cycle behavior in a bilinear zero-sum game, which accelerates the convergence. Furthermore, Daskalakis et al. [2018] did not provide theoretical analysis of OPTIMISTIC-ADAM.

4 Gradient Prediction

From the analysis in the previous section, we know that whether OPTIMISTIC-AMSGRAD converges faster than its counterpart depends on how m_t is chosen. In OPTIMISTIC-ONLINE LEARNING, m_t is usually set to $m_t = g_{t-1}$, which means that it uses the previous gradient as a guess of the next one. The choice can accelerate the convergence to equilibrium in some two-player zero-sum games Rakhlin and Sridharan [2013], Syrgkanis et al. [2015], Daskalakis et al. [2018], in which each player uses an optimistic online learning algorithm against its opponent.

However, this paper is about solving optimization problems instead of solving zero-sum games. We propose to use the extrapolation algorithm of Scieur et al. [2016]. Extrapolation studies estimating the limit of sequence using the last few iterates Brezinski and Zaglia [2013]. Some classical works include Anderson acceleration Walker and Ni. [2011], minimal polynomial extrapolation Cabay and

Jackson [1976], reduced rank extrapolation Eddy [1979]. These methods typically assume that the sequence $\{x_t\} \in \mathbb{R}^d$ has a linear relation

$$x_t = A(x_{t-1} - x^*) + x^*, \tag{4}$$

and $A \in \mathbb{R}^{d\times d}$ is an unknown, not necessarily symmetric, matrix. The goal is to find the fixed point of x^*. Scieur et al. [2016] relaxes the assumption to certain degrees. It assumes that the sequence $\{x_t\} \in \mathbb{R}^d$ satisfies

$$x_t - x^* = A(x_{t-1} - x^*) + e_t, \tag{5}$$

where e_t is a second order term satisfying $\|e_t\|_2 = O(\|x_{t-1} - x^*\|_2^2)$ and $A \in \mathbb{R}^{d\times d}$ is an unknown matrix. The extrapolation algorithm we used is shown in Algorithm 4. Some theoretical guarantees regarding the distance between the output and x^* are provided in Scieur et al. [2016].

Algorithm 4 REGULARIZED APPROXIMATE MINIMAL POLYNOMIAL EXTRAPOLATION (RMPE) Scieur et al. [2016]

1: **Input:** sequence $\{x_s \in \mathbb{R}^d\}_{s=0}^{s=r}$, parameter $\lambda > 0$.
2: Compute matrix $U = [x_1 - x_0, \ldots, x_r - x_{r-1}] \in \mathbb{R}^{d\times r}$.
3: Obtain z by solving $(U^\top U + \lambda I)z = \mathbf{1}$.
4: Get $c = z/(z^\top \mathbf{1})$.
5: **Output:** $\Sigma_{i=0}^{r-1} c_i x_i$, the approximation of the fixed point x^*.

For OPTIMISTIC-AMSGRAD, we use Algorithm 4 to get m_t. Specifically, m_t is obtained by

- Call Algorithm 4 with input being a sequence of some past $r+1$ gradients, $\{g_t, g_{t-1}, g_{t-2}, \ldots, g_{t-r}\}$, where r is a parameter.
- Set $m_t := \Sigma_{i=0}^{r-1} c_i g_{t-r+i}$ from the output of Algorithm 4.

To see why the output from the extrapolation method may be a reasonable estimation, assume that the update converges to a stationary point (i.e. $g^* := \nabla f(w^*) = 0$ for the underlying function f). Then, we might rewrite (5) as

$$g_t = Ag_{t-1} + O(\|g_{t-1}\|_2^2)u_{t-1}, \tag{6}$$

for some vector u_{t-1} with a unit norm. The equation suggests that the next gradient vector g_t is a linear transform of g_{t-1} plus an error vector that may not be in the span of A whose length is $O(\|g_{t-1}\|_2^2)$. If the algorithm is guaranteed to converge to a stationary point, the magnitude of the error component will eventually go to zero.

We remark that the choice of algorithm for gradient prediction is surely not unique. We propose to use the recent result among various related works. Indeed, one can use any method that can provide reasonable guess of gradient in next iteration.

Two illustrative examples. We provide two toy examples to demonstrate how OPTIMISTIC-AMSGRAD works with the chosen extrapolation method. First, consider minimizing a quadratic function $H(w) := \frac{b}{2}w^2$ with vanilla gradient descent method $w_{t+1} = w_t - \eta_t \nabla H(w_t)$. The gradient $g_t := \nabla H(w_t)$ has a recursive description as $g_{t+1} = bw_{t+1} = b(w_t - \eta_t g_t) = g_t - b\eta_t g_t$. So, the update can be written in the form of (6) with $A = (1 - b\eta)$ and $u_{t-1} = 0$ by setting $\eta_t = \eta$ (constant step size). Therefore, the extrapolation method should predict well.

Specifically, consider optimizing $H(w) := w^2/2$ by the following three algorithms with the same step size. One is Gradient Descent (GD): $w_{t+1} = w_t - \eta_t g_t$, while the other two are OPTIMISTIC-AMSGRAD with $\beta_1 = 0$ and the second moment term $\hat{v}_t$ being dropped: $w_{t+\frac{1}{2}} = \Pi_{\mathcal{K}}\left[w_{t-\frac{1}{2}} - \eta_t g_t\right]$, $w_{t+1} = \Pi_{\mathcal{K}}\left[w_{t+\frac{1}{2}} - \eta_{t+1} m_{t+1}\right]$. We denote the algorithm that sets $m_{t+1} = g_t$ as Opt-1, and denote the algorithm that uses the extrapolation method to get m_{t+1} as Opt-extra. We let $\eta_t = 0.1$ and the initial point $w_0 = 5$ for all the three methods. The simulation results are on Figure 2 (a) and (b). Sub-figure (a) plots update w_t over iteration, where the updates should go towards the optimal point 0. Sub-figure (b) is about a scaled and clipped version of m_t, defined as $w_t - w_{t-1/2}$, which can be viewed as $-\eta_t m_t$ if the projection (if exists) is lifted. Sub-figure (a) shows that Opt-extra converges faster than the other methods. Furthermore, sub-figure (b) shows that the prediction by the extrapolation method is better than the prediction by simply using the previous gradient. The

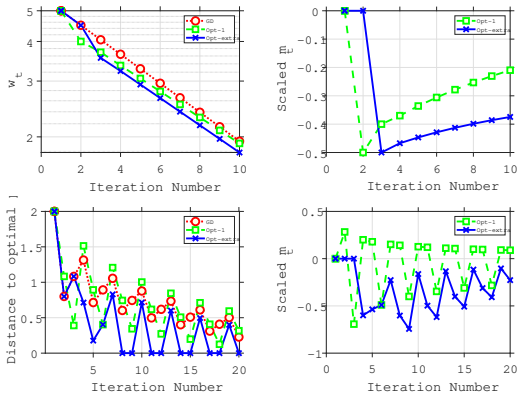

Figure 2: (a): The iterate w_t; the closer to the optimal point 0 the better. (b): A scaled and clipped version of m_t: $w_t - w_{t-1/2}$, which measures how the prediction of m_t drives the update towards the optimal point. In this scenario, the more negative the better. (c): Distance to the optimal point -1. The smaller the better. (d): A scaled and clipped version of m_t: $w_t - w_{t-1/2}$, which measures how the prediction of m_t drives the update towards the optimal point. In this scenario, the more negative the better.

sub-figure shows that $-m_t$ from both methods all point to 0 in all iterations and the magnitude is larger for the one produced by the extrapolation method after iteration 2. [2]

Now let us consider another problem: an online learning problem proposed in Reddi et al. [2018] [3]. Assume the learner's decision space is $\mathcal{K} = [-1, 1]$, and the loss function is $\ell_t(w) = 3w$ if $t \bmod 3 = 1$, and $\ell_t(w) = -w$ otherwise. The optimal point to minimize the cumulative loss is $w^* = -1$. We let $\eta_t = 0.1/\sqrt{t}$ and the initial point $w_0 = 1$ for all the three methods. The parameter λ of the extrapolation method is set to $\lambda = 10^{-3} > 0$. The results are on Figure 2 (c) and (d). Sub-figure (c) shows that Opt-extra converges faster than the other methods while Opt-1 is not better than GD. The reason is that the gradient changes from -1 to 3 at $t \bmod 3 = 1$ and it changes from 3 to -1 at $t \bmod 3 = 2$. Consequently, using the current gradient as the guess for the next clearly is not a good choice, since the next gradient is in the opposite direction of the current one. Sub-figure (d) shows that $-m_t$ by the extrapolation method always points to $w^* = -1$, while the one by using the previous negative direction points to the opposite direction in two thirds of rounds. It shows that the extrapolation method is much less affected by the gradient oscillation and always makes the prediction in the right direction, which suggests that the method can capture the aggregate effect.

5 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 Reddi et al. [2018]. 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 3) 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

[2] The extrapolation method needs at least two gradients for prediction. This is why in the first two iterations, m_t is 0.

[3] Reddi et al. [2018] uses this example to show that ADAM Kingma and Ba [2015] fails to converge.

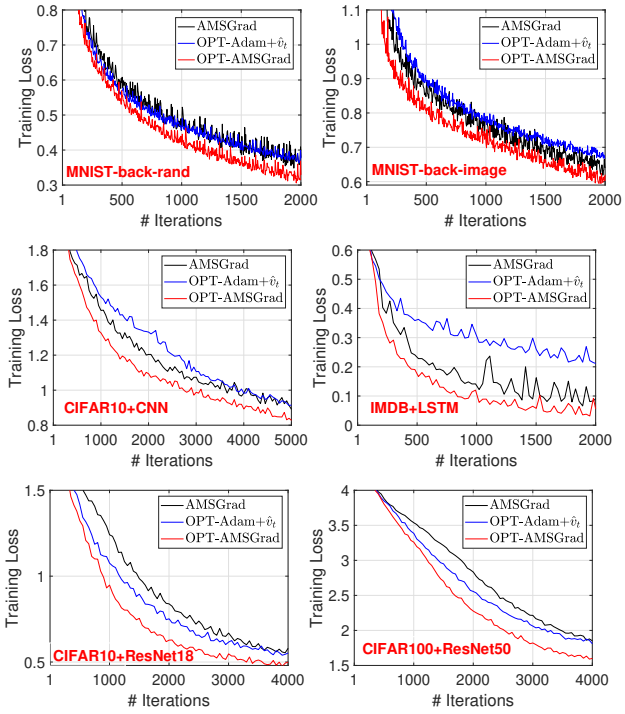

Figure 3: Training loss vs. Number of iterations. The first row are results with fully-connected neural network.

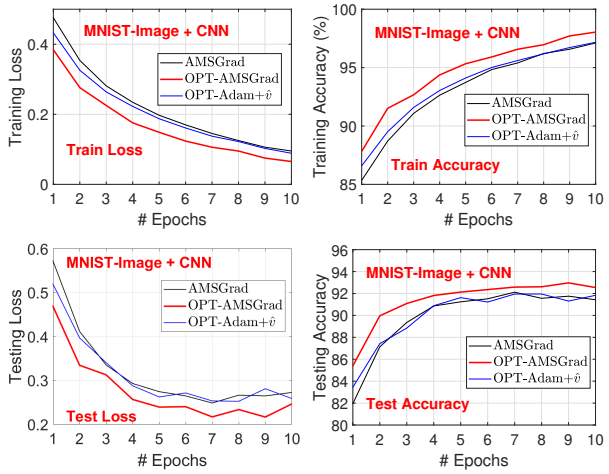

Figure 4: *MNIST-back-image* + convolutional neural network.

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 Reddi et al. [2018] and Kingma and Ba [2015], 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 Larochelle et al. [2007]. 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

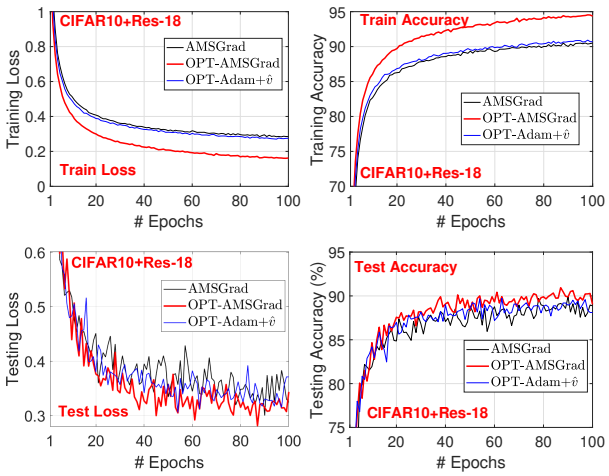

Figure 5: *CIFAR10* + Res-18. We compare three methods in terms of training (cross-entropy) loss, training accuracy, testing loss, and testing accuracy. We observe that OPTIMISTIC-AMSGRAD consistently improves the two baselines.

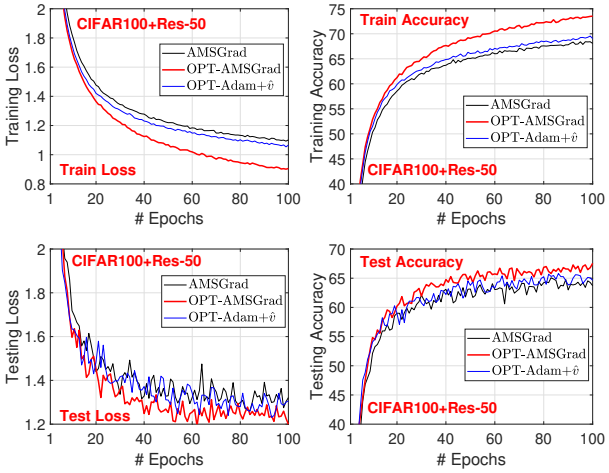

Figure 6: *CIFAR100* + Res-50. We compare three methods in terms of training (cross-entropy) loss, training accuracy, testing loss, and testing accuracy.

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 Springenberg et al. [2015], which is constructed with several convolutional blocks and dropout layers. In addition, we also apply residual networks, Resnet-18 and Resnet-50 He et al. [2016], which have achieved many state-of-the-art results. For the texture *IMDB* dataset, we consider training a Long-Short Term

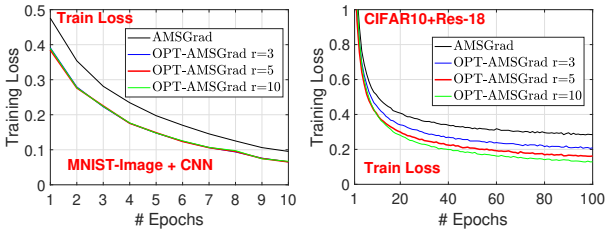

Figure 7: The training loss of OPTIMISTIC-AMSGRAD with different r.

Memory (LSTM) network. The 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 3. 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.

We are also curious about the long-term performance and generalization of the proposed method in test phase. In Figure 4, Figure 5 and Figure 6, 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.

5.1 Choice of parameter r

Recall that our proposed algorithm has the parameter r that governs the use of past information. Figure 7 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.

6 Concluding Remarks

6.1 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 5 and 6, 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., Martens and Grosse [2015]), as Natural gradient method needs to store some matrix inverse.

6.2 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.

References

J. Abernethy, K. A. Lai, K. Y. Levy, and J.-K. Wang. Faster rates for convex-concave games. *COLT*, 2018.

N. Agarwal, B. Bullins, X. Chen, E. Hazan, K. Singh, C. Zhang, and Y. Zhang. Efficient full-matrix adaptive regularization. *ICML*, 2019.

C. Brezinski and M. R. Zaglia. Extrapolation methods: theory and practice. *Elsevier*, 2013.

S. Cabay and L. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 1976.

X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *ICLR*, 2019a.

Z. Chen, Z. Yuan, J. Yi, B. Zhou, E. Chen, and T. Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. *ICLR*, 2019b.

C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu. Online optimization with gradual variations. *COLT*, 2012.

C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training gans with optimism. *ICLR*, 2018.

T. Dozat. Incorporating nesterov momentum into adam. *ICLR (Workshop Track)*, 2016.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011.

R. Eddy. Extrapolating to the limit of a vector sequence. *Information linkage between applied mathematics and industry, Elsevier*, 1979.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, 2014.

A. Graves, A. rahman Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. *ICASSP*, 2013.

E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *ICML*, 2007.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *NIPS*, 2017.

X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *AISTAT*, 2019.

J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *ICML*, 2015.

H. B. McMahan and M. J. Streeter. Adaptive bound optimization for online convex optimization. *COLT*, 2010.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *NIPS (Deep Learning Workshop)*, 2013.

M. Mohri and S. Yang. Accelerating optimization via adaptive prediction. *AISTATS*, 2016.

Y. Nesterov. Introductory lectures on convex optimization: A basic course. *Springer*, 2004.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *Mathematics and Mathematical Physics*, 1964.

A. Rakhlin and K. Sridharan. Optimization, learning, and games with predictable sequences. *NIPS*, 2013.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration. *NIPS*, 2016.

J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *ICLR*, 2015.

V. Syrgkanis, A. Agarwal, H. Luo, and R. E. Schapire. Fast convergence of regularized learning in games. *NIPS*, 2015.

T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. 2008.

H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 2011.

R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *ICML*, 2019.

M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. *NeurIPS*, 2018.

M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv:1212.5701*, 2012.

D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv:1808.05671*, 2018.

F. Zou and L. Shen. On the convergence of adagrad with momentum for training deep neural networks. *arXiv:1808.03408*, 2018.

A Proof of Theorem ??

Proof This completes the proof. □