On the Algorithmic Stability and Generalization of Adaptive Optimization Methods

Abstract

Despite their popularity in deep learning and machine learning in general, the theoretical properties of adaptive optimizers such as Adagrad, RMSProp, Adam or AdamW are not yet fully understood. In this paper, we develop a novel framework to study the stability and generalization of these optimization methods. Based on this framework, we show provable guarantees about such properties that depend heavily on a single parameter β_2. Our empirical experiments support our claims and provide practical insights into the stability and generalization properties of adaptive optimization methods.

1 Introduction

Recent years have witnessed a surge of interest in adaptive optimization methods for deep learning settings. For instance, Adam [1] — despite its drawbacks in theory [2] — is often used in practice. Other popular choices of adaptive methods include Adagrad [3], RMSProp [4] and AdamW [5]. All of those methods are now the standard choices in deep learning communities. However, such adaptive methods are not always the winner, because people still find the traditional methods of Gradient Descent (GD) and Stochastic Gradient Descent (SGD) more superior in certain cases, both in terms of optimization and generalization [2, 6]. Despite their different yet prevalent usages, the choices of which optimizer to use are not entirely clear and often come from practice or user experiences. Likewise, there has been no known theoretical foundation that thoroughly explains the reasons as to why we use a particular optimizer in this case, but a different one in another case. In this work, we build such a theoretical framework that offers reasons concerning that important question.

Retrospectively, the development of adaptive optimization methods could be dated back to the work of [7] and [3] in the online learning setting. Their methods work well in sparse settings [8]. But the adaptive learning rate often decays rapidly for dense gradients, thus slowing down the convergence speed of their methods significantly. To tackle this issue, some approaches such as Adadelta [9], RMSProp [4], and Adam [1] proposed to use an exponential moving average of past squared gradients. In particular, Adam has become an increasingly popular optimizer in deep learning due to its effectiveness in the early training stage [10]. However, few facts are known about these adaptive methods in terms of optimization and generalization.

Generally, when it comes to a learning problem, there are two most important metrics that people care about, which are directly related to the optimizers being used. One is the convergence property such as convergence guarantees and convergence rate, which offer insights about the fundamental quality of the optimizer in terms of reliability and run-time complexity [2]. The other metric is generalization, which evaluates how well the optimizer does on unseen test data comparing to its performance on the train data [11]. Oftentimes there is a tradeoff between those two metrics: one can not usually perform well on train data (i.e. convergence) as well as on test data (i.e. generalization) [12]. Specifically, the generalization error is usually large when the training loss is small and vice versa. Importantly, the generalization error is upper bounded by the stability of the optimizer [13]. Roughly speaking, stability measure the sensitivity of an optimizer with respect to the change of train data. This is an important concept because we do not want to use an optimizer that finds very different solutions

Preprint. Under review.

when we just slightly change the train data. In this paper, we study the adaptive optimization methods through the lens of stability and draw some conclusions about their generalization as a consequence.

Our contributions. First, we provide a novel analysis for the stability of adaptive optimization methods which is different from the previous analysis for the stability of SGD and SGD with momentum by [14, 15]. Second, we show that the stability of the adaptive optimization methods depends on the parameter β_2. Specifically, the bigger the β_2, the more stable the algorithm is. Our experiments confirm this theory. Third, we show that when β_2 goes to 1 at the rate $1 - \alpha/t$, the stability grows as $O(T^r), r < 1$ where T is number of iterations. Our experiments reflect this rate. Finally, we show that weight decay helps improve the stability and generalization of Adam. Our experimental results also validate this theory.

2 Related Work

Convergence. The convergence of adaptive optimization methods has been studied under various conditions. Adaptive learning rate in Adagrad was shown to improve convergence in sparse, convex setting [3, 7] and in non-convex setting [16]. In convex setting, [2] proposed AMSGrad to fix a convergence issue in Adam in certain online and stochastic learning cases due to its constant learning rate, and hence stimulated following work trying to improve Adam. [17] proposed Adabound that used a clipping technique and gradually forced adaptive learning rate to converge to a predefined value. [10] proposed SAdam and SAMSGrad, which applied softmax to each coordinate of the adaptive learning rates to keep them low variance. In non-convex settings, the convergence of Adam has been studied by several works. [18] showed that Adam converged when increasing the batch size. [19] then showed that under some mild assumptions, the deterministic Adam converged. Additionally, [20, 21] proved the convergence of generalize versions of Adam. Recently, some works have proven that AMSGrad converged in the weakly convex setting [22, 23]. While all of them showed that Adam converged at the rate of $O(\frac{1}{\sqrt{T}})$, none of them answered the question as to why Adam-like algorithms have any benefits over SGD in terms of optimization, unlike in our work.

Stability. Algorithmic stability is a fundamental concept in learning theory that dates back to the pioneering work by [24], who showed that the expected sensitivity of a classification algorithm (kNN in particular) to changes of individual examples could be used to obtain a variance bound of a leave-one-out estimator. Later, [13] introduced the concept of uniform stability, from which the empirical risk minimization (EMR) was uniformly stable if the objective function was strongly convex. This concept was further extended to study randomized algorithms such as bagging and boosting [25]. [26] then proposed weaker on-average stability and used it to study unregularized learning algorithms [27]. Very recently, almost optimal high-probability bounds were established for uniformly stable algorithms by developing elegant concentration inequalities for weakly-dependent random variables [28, 29, 30]. Other notion of stability were also proposed such as the uniform argument stability [31], hypothesis stability [13] and hypothesis set stability [32]. The deep connection between algorithmic stability and learnability was identified [26, 33]. Our work is built upon the foundation of those ideas.

Generalization. In a seminal paper, [14] used uniform stability to derive generalization bound for SGD in the strongly convex, convex, and non-convex settings. This stability analysis was further refined by [34] by using the on-average variance and was more recently extended to the non-smooth setting by [35]. In another work, [36] developed a bound for SGD concerning data-dependent stability, a nice property that showed how initialization would affect generalization. Furthermore, [37] developed a tradeoff lower bound between stability and convergence of iterative optimization algorithms. They also proved stability bound for momentum optimizers such as Nesterov acceleration and a heavy-ball method with a fixed momentum when the objective function is quadratic. [15, 38] derived the stability bound for heavy-ball method for general convex and non-convex functions. In contrast to SGD and SGD with momentum, there have been no existing results in the literature on the stability and generalization for adaptive optimization methods, which are addressed by us in this work.

Other approaches were also developed for characterizing the generalization error as well as the estimation error, which are orthogonal to our work. Some of them were based on the information-theoretic approach [39, 38, 40, 41], the algorithm robustness framework [42, 43], large-margin theory [44], and the classical VC theory [45].

3 Preliminaries

In order to establish our main results, we first formulate related fundamental concepts concerning generalization, algorithmic stability, and adaptive optimization methods.

3.1 Excess risk decomposition

In this section, we briefly review fundamental concepts of empirical risk minimization and excess risk decomposition. We consider the standard setting in supervised learning problems. In this setting, we are given a sample $S = \{z_1, \ldots, z_n\}$ of size n where each data point z_i lies in some space $\mathcal{Z}$ and is drawn i.i.d according to an unknown distribution $\mathbb{P} \in \mathcal{P}$. Given a loss function $f(x; z)$, we want to find some $x \in \Omega \subset \mathbb{R}^d$ that minimize the expected loss $F(x) = \mathbb{E}_{z\sim\mathbb{P}} f(x; z) = \int_{\mathcal{Z}} f(x, z) d\mathbb{P}(z)$.

Since the distribution $\mathbb{P}$ is unknown, we instead minimize the empirical loss $F_S(x) = \frac{1}{n}\sum_{i=1}^{n} f(x; z_i)$. We denote by x_S an estimator computed from sample S. The statistical question is how to bound the excess risk in terms of the difference between the population risk evaluated at x_S and the true minimal risk over the entire parameter space Ω, $\mathcal{R}(x_S) = F(x_S) - \inf_{x\in\Omega} F(x)$. We have the following lemma about expected risk decomposition.

Lemma 1. *Let x_S^* be an empirical risk minimizer. We then have that*

$$\mathbb{E}_S[\mathcal{R}(x_S)] \leq \mathcal{E}_{gen} + \mathcal{E}_{opt} = \mathbb{E}_S[F(x_S) - F_S(x_S)] + \mathbb{E}_S[F_S(x_S) - F_S(x_S^*)],$$

where $\mathcal{E}_{gen}$ and $\mathcal{E}_{opt}$ are the expected generalization error and optimization error respectively.

For proof, please see Appendix A. Similar derivations to this result can also be found in [46, 12, 37].

3.2 Algorithmic stability

It turns out that the expected generalization error can be controlled by various notions of algorithmic stability. For a thorough review, please refer to [13, 26]. For the purpose of this paper, we are only interested in the notion of uniform stability introduced by [13].

Definition 1. *An algorithm, which output a model x_S for sample S, is $\tau-uniformly\ stable$ if for all $k \in \{1, \ldots, n\}$, for all data sample pair $S = \{z_1, \ldots, z_k, \ldots, z_n\}$, and $S' = \{z_1, \ldots, z'_k, \ldots, z_n\}$, where z_i and z'_k are i.i.d sampled from $\mathbb{P}$, we have $sup_{z\in\mathcal{Z}} |f(x_S; z) - f(x_{S'}; z)| \leq \tau$.*

This definition implies that when we randomly replace one arbitrary sample from the training data with another i.i.d one, the deviation of the loss between two output models is uniformly within some τ. Moreover, the smaller the τ is, the more stable the algorithm is. One important property of uniform stability is that it implies generalization, as formulated in the following theorem.

Theorem 1. *If an algorithm that outputs a model x_S for sample S is τ-uniformly-stable, then its expected generalization error is bounded as follows,*

$$|\mathbb{E}_S[F(x_S) - F_S(x_S)]| \leq \tau. \tag{1}$$

The proof of Theorem 1 can be found in [14]. For the rest of this paper, we focus on finding an upper bound for τ in Definition 1 when using the adaptive optimization methods.

Remark 1. *The concept of uniform stability has a strong connection with the sensitivity analysis in optimization. Sensitivity analysis studies the sensitivity of the optimal value of an optimization problem with respect to perturbations of the problem's constraints. For more detail about sensitivity analysis, please refer to [47].*

3.3 Adaptive optimization methods

$$m_t = \alpha_t m_{t-1} + (1 - \alpha_t)\nabla f(x_{t-1}, z_t), \tag{2}$$

$$v_t = \beta_t v_{t-1} + (1 - \beta_t)\nabla f(x_{t-1}, z_t)^2, \tag{3}$$

$$x_t = x_{t-1} - \eta_t \frac{m_t}{\sqrt{v_t + \epsilon}}, \tag{4}$$

$$\dot{m}(t) = p(t)(\nabla F(x(t)) - m(t)), \tag{5}$$

$$\dot{v}(t) = q(t)(\nabla F(x(t))^2 - v(t)), \tag{6}$$

$$\dot{x}(t) = -\frac{m(t)}{\sqrt{v(t) + \epsilon}}, \tag{7}$$

In this section, we review the general formulation the adaptive optimization methods (AOM). Let $(\mathcal{Z}, \mathcal{F}, \mathbb{P})$ be a probability space. Consider the minimization problem $\min_{x\in\Omega} F(x)$, where $F(x)$ is defined in Section 3.1, and $f : \Omega \times \mathcal{Z} \to \mathbb{R}$ is a measurable map. For a fixed z, the mapping $x \mapsto f(x, z)$ is supposed to be differentiable and its gradient w.r.t x is $\nabla f(x, z)$. Starting from $(x_0, m_0, v_0) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d_+$, AOM generates updates at time step t as shown in Equations (2, 3, 4), where $\alpha_t, \beta_t \in (0, 1)$ and $\epsilon > 0$ (to prevent zero division). When $\alpha_t = \alpha, \beta_t = \beta$, the above

formulation becomes Adam algorithm. Notice that the α and β in this case corresponding to the β_1 and β_2 in Adam optimizer. When $\alpha_t = 0, \beta_t = 1 - \frac{\alpha}{t}$, we obtain Adagrad optimizer.

In the continuous setting, the formulation above can be shown as the noisy Euler's discretization of the ordinary differential equation (ODE) in Equations (5, 6, 7), where $(x(0), m(0), v(0)) = (x_0, m_0, v_0)$ [48]. This ODE's formulation helps us overcome the difficulties in the latter analysis. We will look at AOM as a system of two equation with variables (x, v) instead of treating v as a separate adaptive learning rate.

4 Stability of adaptive optimization methods

In this section, we study the stability of the AOM optimizer when $\alpha_t = 0$ for all t. We leave the analysis with $\alpha_t > 0$ as a future research direction. In this case, we show that the stability of the AOM method depends on the parameter β_t. Moreover, we show that the AOM method is uniformly stable when $\beta_t \leq 1 - \alpha/t$. As a consequence, the Adagrad algorithm is a uniformly stable algorithm.

For latter convenience, let us first set up some notations and assumptions. We denote $S = \{z_1, \ldots, z_i, \ldots, z_n\}$ and $S' = \{z_1, \ldots, z'_i, \ldots, z_n\}$ to be two training sets which are drawn from the unknown distribution $\mathbb{P}$. All data points in S and S' are the same except the data point at the i-*th* position. Given a fixed model, we consider training that model by using the AOM on the above two data sets. We denote

$$v_{t+1} = \beta_t v_t + (1-\beta_t)\nabla f(x_t, z_{i_t})^2 \quad (8) \qquad v'_{t+1} = \beta_t v'_t + (1-\beta_t)\nabla f(x'_t, z'_{i_t})^2 \quad (10)$$

$$x_{t+1} = x_t - \eta_t \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} \quad (9) \qquad x'_{t+1} = x'_t - \eta_t \frac{\nabla f(x'_t, z'_{i_t})}{\sqrt{v'_{t+1} + \epsilon}} \quad (11)$$

where (x_t, v_t) and (x'_t, v'_t) are the outputs of the AOM when running on S and S' respectively. Specifically, at every time step t, we pick an index i_t uniformly random from the set $\{1, \ldots, n\}$ and update the parameters according to the AOM update rules. Notice that we use two different notation z_{i_t} and z'_{i_t} because we pick data from two different training sets S and S'. In addition, for a fixed $t_0 \in \{1, \ldots, n\}$, we denote

$$\delta_t = \|x_t - x'_t\|_2, \quad \Delta_t = \mathbb{E}[\delta_t | \delta_{t_0} = 0], \quad (12)$$

$$\sigma_t = \|v_t - v'_t\|_2, \quad \Sigma_t = \mathbb{E}[\sigma_t | \delta_{t_0} = 0]. \quad (13)$$

The analysis relies on the following set of assumptions:

(1) $f(., z)$ is L-Lipschitz and μ-smooth for all $z \in \mathcal{Z}$, i.e.
$$|f(x,z) - f(y,z)| \leq \mu\|x - y\|_2, \text{and} \quad \|\nabla f(x,z) - \nabla f(y,z)\|_2 \leq L\|x - y\|_2 \quad (14)$$
which imply $\|\nabla f(x,z)\|_2 \leq \mu$ for all $z \in \mathcal{Z}$.

(2) There exists $M > 0$ such that $f(x, z) \leq M$ for all $x, z \in \Omega \times \mathcal{Z}$.

(3) There exists $\lambda_1, \lambda_2 \geq 0$ such that
$$\min_t \min_i \{v_{t+1}, v'_{t+1}\} \geq \lambda_1, \qquad \max_t \max_i \{v_{t+1}, v'_{t+1}\} \leq \lambda_2. \quad (15)$$

Note that assumption (1) is standard when analyzing the stability of optimization algorithms [13, 14]. Assumption (2) is easily achieved if we restrict our domain to a compact set. Assume that we run the AOM on S and S' for T steps to get the outputs x_T and x'_T. We first have the following lemma which controls the difference of the loss at x_T and x'_T. The proof of this lemma can be found in [14]. For completeness, we also provide a proof of this lemma in the appendix.

Lemma 2. *Assume that assumptions (1) and (2) are satisfied. Let S and S' be two samples of size n differing in only a single example. Denote by x_T and x'_T the output after T steps of AOM on S and S', respectively. Then for all $z \in \mathcal{Z}$ and $t_0 \in \{1, \ldots, n\}$, we have that*

$$\mathbb{E}|f(x_T; z) - f(x'_T; z)| \leq 2M\frac{t_0}{n} + \mu\mathbb{E}[\delta_T | \delta_{t_0} = 0] = 2M\frac{t_0}{n} + \mu\Delta_T.$$

The insight of Lemma 2 is that AOM has to take several steps before it encounters the index i where the two data sets are different. This will be important later in our analysis because our loss function could be non-convex. In real application, n is usually very big so it will take a long time before AOM reach the index i where the two data sets are different. From now on, we will fix $t_0 \in \{1, \ldots, n\}$. Given this lemma, it remains to estimate the quantity Δ_T. Next, we have the following theorem.

Theorem 2. *Assume that assumptions (1), (2), (3) are satisfied. Suppose we run AOM with step size η_t on S and S' respectively, starting from the same initialization. Then for any outputs x_t and x'_t, we*

have that

$$\Delta_{t+1} \leq \Delta_t + \eta_t \left(1 - \frac{1}{n}\right) \frac{L}{\sqrt{\lambda_1 + \epsilon}} \left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1 + \epsilon}\right] \Delta_t + \eta_t \left(1 - \frac{1}{n}\right) \frac{\mu\beta_t}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)} \Sigma_t + \eta_t \frac{1}{n} \frac{2\mu}{\sqrt{\lambda_1 + \epsilon}}. \tag{16}$$

For a full proof, please see Appendix C. This theorem gives us a general upper bound of Δ_t. However, the bound also involves the term Σ_t which is different from that of SGD [14]. This is expected because the formulation of the AOM involves a system of equations. Thus, in order to bound Δ_t, we also have to construct an upper bound for Σ_t, as described in Theorem 3.

Theorem 3. *Assume that assumptions (1), (2), (3) are satisfied. Suppose we run AOM with step size η_t on S and S' respectively, starting from the same initialization. Then for any output x_t and x'_t we have that*

$$\Sigma_{t+1} \leq \beta_t \Sigma_t + 2(1-\beta_t) \left(1 - \frac{1}{n}\right) L\mu\Delta_t + 2\frac{1}{n}(1-\beta_t)\mu^2. \tag{17}$$

The full proof is in Appendix D. Then combining Theorems 2 and 3, we obtain the following dynamical inequality:

$$\begin{bmatrix} \Delta_{t+1} \\ \Sigma_{t+1} \end{bmatrix} \leq \begin{bmatrix} \mathcal{U}_t & \mathcal{P}_t \\ \mathcal{Q}_t & \mathcal{R}_t \end{bmatrix} \begin{bmatrix} \Delta_t \\ \Sigma_t \end{bmatrix} + \frac{1}{n} \begin{bmatrix} \eta_t \frac{2}{\sqrt{\lambda_1 + \epsilon}} \mu \\ (1-\beta_t) 2\mu^2 \end{bmatrix}, \tag{18}$$

where

$$\mathcal{U}_t = 1 + \eta_t \left(1 - \frac{1}{n}\right) \frac{L}{\sqrt{\lambda_1 + \epsilon}} \left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1 + \epsilon}\right], \qquad \mathcal{P}_t = \eta_t \left(1 - \frac{1}{n}\right) \frac{\mu\beta_t}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)},$$

$$\mathcal{R}_t = \beta_t, \qquad \mathcal{Q}_t = 2L\mu(1-\beta_t)\left(1 - \frac{1}{n}\right)$$

Denote $A_t = \begin{bmatrix} \mathcal{U}_t & \mathcal{P}_t \\ \mathcal{Q}_t & \mathcal{R}_t \end{bmatrix}$ and

$$U_t := \left(1 - \frac{1}{n}\right) \frac{L}{\sqrt{\lambda_1 + \epsilon}} \left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1 + \epsilon}\right] \leq U := \left(1 - \frac{1}{n}\right) \frac{L}{\sqrt{\lambda_1 + \epsilon}} \left[1 + \frac{\mu^2}{\lambda_1 + \epsilon}\right],$$

$$V_t := \left(1 - \frac{1}{n}\right) \frac{\mu\beta_t}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)} \leq V := \left(1 - \frac{1}{n}\right) \frac{\mu}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)},$$

$$W := 2\left(1 - \frac{1}{n}\right) L\mu, \qquad Y := \frac{2\mu}{\sqrt{\lambda_1 + \epsilon}}, \qquad Z := 2\mu^2.$$

We then can rewrite $A_t = \begin{bmatrix} 1 + \eta_t U_t & \eta_t V_t \\ (1-\beta_t)W & \beta_t \end{bmatrix}$ and $\begin{bmatrix} \Delta_{t+1} \\ \Sigma_{t+1} \end{bmatrix} \leq A_t \begin{bmatrix} \Delta_t \\ \Sigma_t \end{bmatrix} + \frac{1}{n} \begin{bmatrix} \eta_t Y \\ (1-\beta_t)Z \end{bmatrix}$. Thus the stability of AOM now depends on the norm of the matrix A_t. The following lemma gives an upper bound on the operator norm of A_t for all $t \in \mathbb{N}$.

Lemma 3. *Let $\eta_t = \frac{c}{t}$ and $\beta_t = 1 - \alpha_t, \alpha_t \in (0,1)$. We then have that*

$$\|A_t\|_2 \leq exp\left(\frac{c}{t}\sqrt{D_1} + \frac{c^2}{t^2}\frac{D_1}{2} + \alpha_t D_2\right) \tag{19}$$

where $D_1 = U^2 + V^2, D_2 = \sqrt{W^2 + 1} + \frac{1}{2}(W^2 + 1)$.

The proof of this lemma is in Appendix E. Armed with these results, we are now ready to give a quantitative estimate of the deviation of the model's parameters when running AOM on two data sets S and S'.

Theorem 4. *Assume that assumptions (1), (2), (3) are satisfied. Starting from the same initialization, suppose that we run AOM with step size η_t on S and S' respectively for T iterations and output x_T, x'_T. Let $\eta_t = \frac{c}{t}$ and $\beta_t = 1 - \alpha_t, \alpha_t \in (0,1)$. We then have that*

$$\|\Delta_T\|_2 \leq \frac{1}{n} exp\left(\gamma\frac{c^2 D_1}{2}\right) \times \sum_{t=t_0+1}^{T} exp\left(D_2 \sum_{k=t+1}^{T} \alpha_t\right) \left(\frac{T}{t}\right)^{c\sqrt{D_1}} \left(\frac{1}{t} cY + \alpha_t Z\right) \tag{20}$$

where we define $\gamma = \sum_{k=1}^{\infty} \frac{1}{k^2}$.

For a full proof, please see appendix F. Theorem 4 and Lemma 2 provide a general bound on the stability of the AOM. If we replace $\{\beta_t\}$ by specific values, we obtain the stability bound for some well-know instances of AOM.

Corollary 1. *(Stability bound for Adam with $\beta_1 = 0$) Let $\eta_t = \frac{c}{t}$ and $\beta_t = \beta$. We then have that for any $z \in \mathcal{Z}$*

$$\mathbb{E}|f(x_T; z) - f(x'_T; z)| \leq 2M\frac{t_0}{n} + \frac{\mu}{n} exp\left(\gamma\frac{c^2 D_1}{2} + (1-\beta)D_2 T\right) \times T^{c\sqrt{D_1}}\left(\frac{Y}{\sqrt{D_1}} + (1-\beta)Z\sum_{t=t_0+1}^{T}\frac{1}{t^{c\sqrt{D_1}}}\right).$$

Remark 2. *Although this bound is loose, it provides some insights about the stability of Adam with $\beta_1 = 0$. First, the stability bound depends on the parameter β. Specifically, the optimizer is more stable when β is close to 1 because it reduces the effect of the term $exp((1-\beta)D_2 T)$ and the term $(1-\beta)Z\sum_{t=t_0+1}^{T}\frac{1}{t^{c\sqrt{D_1}}}$. Our experiments confirm this insight. Second, the stability bound also depends on the initial step size c. Small c reduce the effect of the term $exp(\gamma\frac{c^2 D_1}{2})$.*

In order to overcome the exponential bound in the previous corollary, we can let $\{\beta_t\}$ gradually converge to 1. The following corollary shows that this is indeed the case.

Corollary 2. *Let $\eta_t = \frac{c}{t}$ and $\beta_t = 1 - \alpha_t, \alpha_t \in (0,1)$ such that $\alpha_t \leq \frac{\alpha}{t}$ for all t. We then have that* $\mathbb{E}|f(x_T; z) - f(x'_T; z)| \leq 2M\frac{t_0}{n} + \frac{\mu}{n}(cY + \alpha Z)\exp\left(\gamma\frac{c^2 D_1}{2}\right) \times \frac{T^{c\sqrt{D_1}+\alpha D_2}}{c\sqrt{D_1}+\alpha D_2} \times \frac{1}{t_0^{c\sqrt{D_1}+\alpha D_2}}.$

We could actually choose t_0 to minimize the bound in the above corollary as in [14]. We have the following corollary.

Corollary 3. *Let $\eta_t = \frac{c}{t}$ and $\beta_t = 1 - \alpha_t, \alpha_t \in (0,1)$ such that $\alpha_t \leq \frac{\alpha}{t}$ for all t. Then the uniform stability error of AOM is given by*

$$\mathbb{E}|f(x_T; z) - f(x'_T; z)| \leq \frac{1}{n}\left[2M\left(\frac{AB}{2M}\right)^{\frac{1}{A+1}} + \frac{B}{\left(\frac{AB}{2M}\right)^{\frac{A^2}{A+1}}}\right] T^{\frac{A}{A+1}}$$

where we define

$$A := c\sqrt{D_1} + \alpha D_2, \qquad B := (cY + \alpha Z) exp\left(\gamma\frac{c^2 D_1}{2}\right)\frac{1}{c\sqrt{D_1}+\alpha D_2}.$$

Remark 3. *When $\beta_t = 1 - \frac{\alpha}{t}$, we obtain Adagrad algorithm. Thus, this corollary implies that Adagrad algorithm is uniformly stable at rate $O(T^r), 0 < r < 1$. To the best of our knowledge, this is the first result which shows that Adagrad algorithm is a uniformly stable algorithm. Our experiments confirm the rate in the above corollary.*

5 Stability of adaptive optimization methods with weight decay

Weight decay is one of common techniques when training neural network [5]. In this section, we prove that weight decay can actually help improve the stability of adaptive optimization methods. Our analysis focus on AdamW algorithm. Remind that the update of AdamW on two data set S and S' has the form

$$x_{t+1} = (1 - \eta_t\lambda)x_t - \eta_t\frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1}+\epsilon}} \quad (21) \qquad x'_{t+1} = (1 - \eta_t\lambda)x'_t - \eta_t\frac{\nabla f(x'_t, z'_{i_t})}{\sqrt{v'_{t+1}+\epsilon}} \quad (23)$$

$$v_{t+1} = \beta_t v_t + (1-\beta_t)\nabla f(x_t, z_{i_t})^2 \quad (22) \qquad v'_{t+1} = \beta_t v'_t + (1-\beta_t)\nabla f(x'_t, z'_{i_t})^2 \quad (24)$$

where λ is the weight decay parameter. We have the following theorem.

Theorem 5. *Assume that assumptions (1), (2), (3) are satisfied. Suppose we run AdamW with step size η_t and weight decay λ on S and S' respectively. We then have that*

$$\Delta_{t+1} \leq (1 - \eta_t\lambda)\Delta_t + \eta_t\left(1 - \frac{1}{n}\right)\frac{L}{\sqrt{\lambda_1+\epsilon}}\left[1 + \frac{\mu L(1-\beta_t)}{\lambda_1+\epsilon}\right]\Delta_t + \eta_t\left(1-\frac{1}{n}\right)\frac{\mu\beta_t}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}\Sigma_t + \eta_t\frac{1}{n}\frac{2}{\sqrt{\lambda_1+\epsilon}}\mu.$$

The proof of this theorem is similar to the proof of Theorem 2. From now on, we fix $\beta_t = \beta, \eta_t = \eta$, the this inequality then becomes

$$\Delta_{t+1} \leq \left\{1 - \eta\left(\lambda - \left(1 - \frac{1}{n}\right)\frac{L}{\sqrt{\lambda_1 + \epsilon}}\left[1 + \frac{\mu L(1-\beta)}{\lambda_1 + \epsilon}\right]\right)\right\}\Delta_t + \eta\left(1 - \frac{1}{n}\right)\frac{\mu\beta}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)}\Sigma_t + \eta\frac{1}{n}\frac{2}{\sqrt{\lambda_1 + \epsilon}}\mu. \tag{25}$$

In addition, we also have that $\Sigma_{t+1} \leq \beta\Sigma_t + (1-\beta)\left(1 - \frac{1}{n}\right)2L\mu\Delta_t + \frac{1}{n}(1-\beta)2\mu^2$. Define

$$U := \left(1 - \frac{1}{n}\right)\frac{L}{\sqrt{\lambda_1 + \epsilon}}\left[1 + \frac{\mu L(1-\beta)}{\lambda_1 + \epsilon}\right], \quad V := \left(1 - \frac{1}{n}\right)\frac{\mu\beta}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)},$$

$$W := \left(1 - \frac{1}{n}\right)2L\mu.$$

We then can rewrite our system of inequalities as

$$\begin{bmatrix}\Delta_{t+1} \\ \Sigma_{t+1}\end{bmatrix} \leq \begin{bmatrix}1 - \eta(\lambda - U) & \eta V \\ (1-\beta)W & \beta\end{bmatrix}\begin{bmatrix}\Delta_t \\ \Sigma_t\end{bmatrix} + \frac{1}{n}\begin{bmatrix}Y \\ Z\end{bmatrix}, \tag{26}$$

Denote

$$A_t := \begin{bmatrix}\Delta_t \\ \Sigma_t\end{bmatrix} \quad B := \begin{bmatrix}Y \\ Z\end{bmatrix} \quad R := \begin{bmatrix}1 - \eta(\lambda - U) & \eta V \\ (1-\beta)W & \beta\end{bmatrix},$$

we then have that

$$\|R\|_F^2 \leq (1 - \eta(\lambda - U))^2 + \eta^2 V^2 + (1-\beta)^2 W^2 + \beta^2. \tag{27}$$

First, we show that the inequality $(1-\beta)^2W^2 + \beta^2 < 1$ has a solution $\beta \in (0, 1)$. By direct computation, we can show that the equation $(1-\beta)^2W^2 + \beta^2 = 1$ has two distinct solutions $\beta_1 = 1$ and $\beta_2 = \frac{W^2-1}{W^2+1}$. If $W^2 > 1$, then we have that $0 < \beta_2 < \beta_1 = 1$. On the other hand, if $W^2 < 1$, then we have that $\beta_2 < 0 < \beta_1 = 1$. Thus, we conclude that for any $\beta \in (\max\{0, \beta_2\}, 1)$, we always have that $(1-\beta)^2W^2 + \beta^2 < 1$.

Now, let's assume that we choose β such that the above inequality holds. We can then choose η small enough such that $\eta^2V^2 + (1-\beta)^2W^2 + \beta^2 < 1$. Given β, η, we want to choose λ such that $(1 - \eta(\lambda - U))^2 + \eta^2V^2 + (1-\beta)^2W^2 + \beta^2 < 1$. First, we need $1 - \eta(\lambda - U) > 0$, which is equivalent to $\lambda < U + \frac{1}{\eta}$. In addition, we also want $(1-\eta(\lambda-U))^2 + \eta^2V^2 + (1-\beta)^2W^2 + \beta^2 < 1$, or equivalently, $\lambda > U + \frac{1-\sqrt{1-\eta^2V^2-(1-\beta)^2W^2-\beta^2}}{\eta}$. Thus we can find λ such that $\|R\|_F < 1$. With this choice of λ, we have that

$$\|A_{t+1}\| \leq \alpha\|A_t\| + \frac{1}{n}B \leq \frac{1}{n}\frac{B}{1-\alpha}, \tag{28}$$

where $\alpha = \|R\|_F$.

The above derivation shows that with an appropriate choice of β, η, λ, AdamW is a uniformly stable algorithm. In addition, the stability bound is independent of the number of iterations.

6 Experiments

We study the generalization and stability of AOMs with different angles using synthetic to real-world datasets. Following [14], for each training dataset, we first remove a random sample x and make two copies of the training set. Then for the first training set, we randomly replace a data point with x. We then train two models on these two train sets staring from the same initialization. At each iteration, we record the Euclidean distance between the parameters of the two output models. We also record the training loss and test loss of the first model, from which we calculate the generalization error which is the absolute difference between train and test losses. We note that for each metric, we run 20 trials and plot the corresponding mean and variation. All the codes are included in the supplementary material and will be publicly released.

6.1 Synthetic data

Datasets and Tasks. We consider both classification (CLS) and regression (REG) tasks. The details for network architecture, training parameters and additional results can be found in Appendix J. Data generation process is as follows.

1. For CLS, we generate 151 data points in 2D from three isotropic Gaussian blobs (3 classes) where each blob has a standard deviation 1 and the same number of data points. We use the

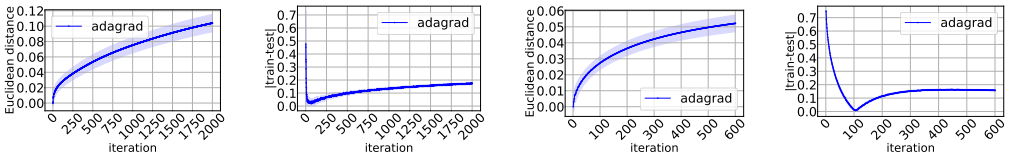

Figure 1: Parameter distance and generalization error when using Adagrad to train neural networks to solve CLS (left) and REG (right) tasks. Both metrics grow in similar fashion, which agree with Corollary 3.

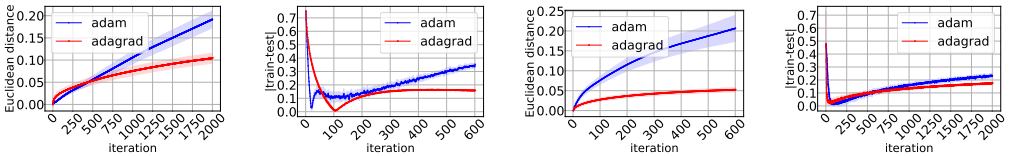

Figure 2: Comparing parameter distance and generalization error between Adam and Adagrad in the classification (left) and regression (right). Results show that Adagrad is more stable than Adam.

package *sklearn.datasets.make_blobs* for this data generation procedure. We then randomly split the data set into an original train set (with 31 data points) and a test set (with 120 data points).

2. For REG, we generate 151 data points from the uniform distribution on $[-1, 1] \times [-1, 1]$. The targets are then generated by the following formula: $y = x_1^2 + x_2^2 + \epsilon$ where the noise $\epsilon \sim \mathcal{N}(0, 0.5^2)$. We then also randomly split the data set into an original train set (with 31 data points) and a test set (with 120 data points).

We choose a small training set in both tasks because we want our models to overfit the training data easily. Thus, we can compare the generalization errors between different algorithms.

Stability and generalization of Adagrad. In this experiment, we consider AOMs with $\alpha_t = 0, \beta_t = 1 - \alpha/t$. This corresponds to Adagrad. For comparison with Adagrad, we consider Adam with $\beta_1 = 0, \beta_2 = 0.999$. As you can see from Figure 1, the parameter distance in both tasks grow as $O(T^r)$ where $r < 1$. This aligns with the result in Corollary 2. The generalization error in both tasks also grows at a similar rate. Additionally, in Figure 2, we can also see that the parameter distance and the generalization error of Adagrad always grow slower than the ones of Adam although we use the same or bigger initial learning rate for Adagrad. This shows that letting β_t goes to 1 at rate $1 - \alpha/t$ make the AOM more stable than using fixed $\beta_t = 0.999$.

Dependence of stability and generalization on fixed β_t. In this experiment, we study the stability and generalization of AOMs when $\alpha_t = 0$ and $\beta_t = \beta$. This corresponds to Adam with $\beta_1 = 0$ and $\beta_2 = \beta$.

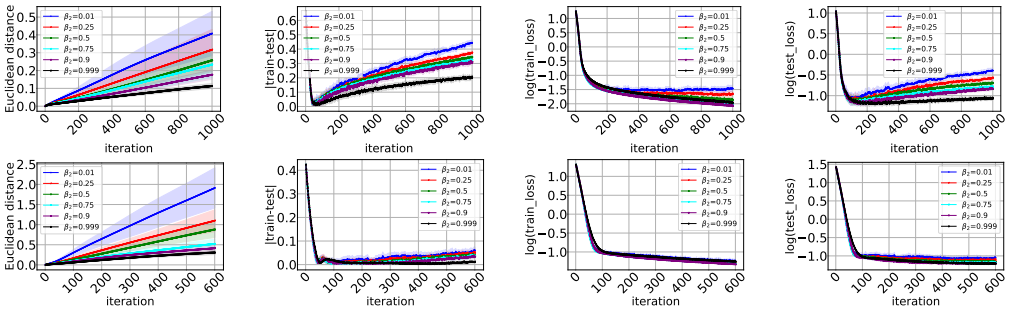

Figure 3: Performances when solving the CLS (top) and REG (bottom) using Adam with different β_2. The bigger β_2 is, the more stable the model gets. *Left to right*: parameter distance, generalization error, train loss and test loss. The losses are plotted in log space.

From Figure 3, we can see that the stability of Adam depends on the parameter β_2, in which the bigger the β_2 is, the more stable the algorithm is. The generalization error also exhibits the same pattern: bigger β_2 gives smaller generalization error. This observation aligns with Corollary 1 although the bound there seems to be loose.

Weight decay helps improve the stability and generalization of Adam. In this experiment, we study how weight decay affects the stability and generalization of Adam in each of CLS and REG tasks, in which the weight decay is set at 5.0 and 1.0, respectively. For Adam with weight decay, we use AdamW [49], and both optimizers use $\beta_1 = 0, \beta_2 = 0.999$.

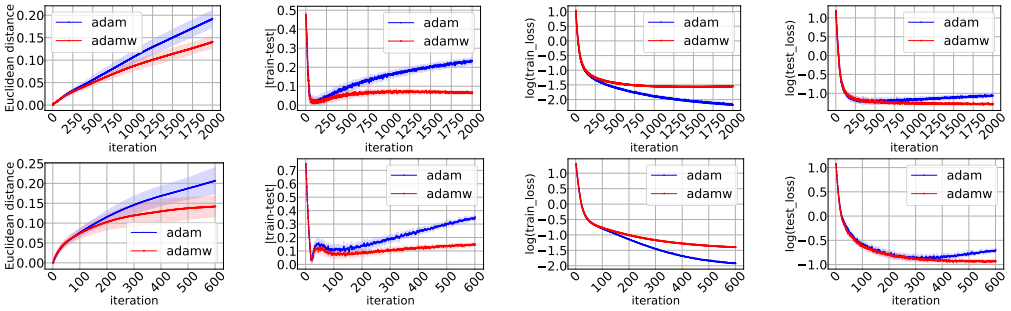

Figure 4: Performance comparison between Adam and AdamW when solving the CLS (top) and REG (bottom). Less overfitting happens to AdamW than to Adam. *Left to right*: parameter distance, generalization error, train loss and test loss. The losses are plotted in log space.

As we can see from Figure 4, AdamW has lower parameter distances and generalization errors comparing to those of Adam in both classification and regression tasks. From the loss patterns, we can also see that the model trained with Adam overfits the train data faster than the model trained with AdamW. These observations show that weight decay helps improve the stability and generalization of Adam, which validates the theory in Section 5.

6.2 Real data

We solve the image classification task on Cifar10 [50] dataset that has 50,000 train and 10,000 test color images of the same resolution 3×32×32. In terms of data augmentation, we only apply mean normalization for both train and test data. For model architecture, we use VGG11 [51], of which the weights are initialized the same for all configurations and later trained with 60 epochs. Due to space limitation, additional information is moved to Appendix K.

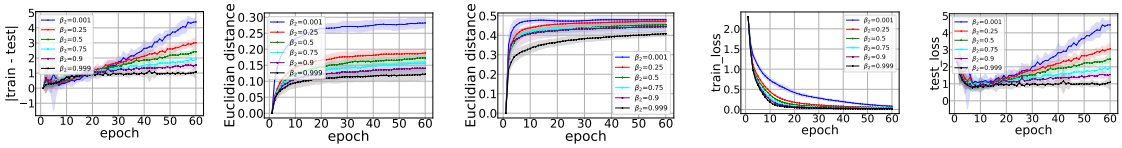

Figure 5: Performances of models on Cifar10 dataset using VGG11. The models get better as β_2 increases from 0 to 1. *Left to right*: generalization error (loss), parameter distance for the first convolution layer, the last fully-connected layer, train loss, and test loss.

As seen in Figure 5, the results for Cifar10 agree with that of synthetic data. In detail, the more β_2 increases, the better the model is in terms of all metrics measured: parameter distance, generalization error, and both losses. In summary, both empirical experiments with real datasets support our theoretical results, in that the bigger β_2 is, the more stable and generalizable AOMs can get.

7 Conclusion

This work establishes a novel theoretical result for stability and generalization for adaptive optimizers, which have been used intensively for, e.g., training neural networks. We show that AOMs depend heavily on a single β_2 value, providing helpful hints in tuning the training process, which is usually dependent on many hyperparameters. Our empirical experiments, which consider the applications of classification and regression, and employ synthetic to real-world datasets, reflect the results claimed in our theory. By building this theoretical framework with empirical validation, we hope to stimulate more work in this optimization area to help thoroughly answer the important question of choices: given a specific case, which optimizer should we use?

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[4] G. Hinton, N. Srivastava, , and K Swersky. Lecture 6d - a separate, adaptive learning rate for each connection. *Slides of Lecture Neural Networks for Machine Learning*, 2012.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[6] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[7] H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

[8] John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems*, pages 2832–2840, 2013.

[9] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[10] Qianqian Tong, Guannan Liang, and Jinbo Bi. Calibrating the learning rate for adaptive gradient methods to improve generalization performance. *arXiv preprint arXiv:1908.00700*, 2019.

[11] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

[12] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. Curran Associates, Inc., 2008.

[13] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[14] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent, 2016.

[15] Ali Ramezani-Kebrya, Ashish Khisti, and Ben Liang. On the stability and convergence of stochastic gradient descent with momentum, 2018.

[16] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization, 2019.

[17] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

[18] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31:9793–9803, 2018.

[19] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration, 2018.

[20] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization, 2020.

[21] Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks, 2020.

[22] Parvin Nazari, Davoud Ataee Tarzanagh, and George Michailidis. Adaptive first-and zeroth-order methods for weakly convex stochastic optimization problems, 2020.

[23] Ahmet Alacaoglu, Yura Malitsky, and Volkan Cevher. Convergence of adaptive algorithms for weakly convex constrained optimization, 2020.

[24] W. H. Rogers and T. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *Annals of Statistics*, 6:506–514, 1978.

[25] Andre Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6(3):55–79, 2005.

[26] S. Shalev-Shwartz, O. Shamir, Nathan Srebro, and K. Sridharan. Learnability, stability and uniform convergence. *J. Mach. Learn. Res.*, 11:2635–2670, 2010.

[27] Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems, 2017.

[28] Andreas Maurer. A second-order look at stability and generalization. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1461–1475, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

[29] Vitaly Feldman and Jan Vondrak. High probability generalization bounds for uniformly stable algorithms with nearly optimal rate, 2019.

[30] Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. Sharper bounds for uniformly stable algorithms, 2020.

[31] Tongliang Liu, Gábor Lugosi, Gergely Neu, and Dacheng Tao. Algorithmic stability and hypothesis complexity, 2017.

[32] Dylan J. Foster, Spencer Greenberg, Satyen Kale, Haipeng Luo, Mehryar Mohri, and Karthik Sridharan. Hypothesis set stability and generalization, 2020.

[33] Alexander Rakhlin, Sayan Mukherjee, and Tomaso Poggio. Stability results in learning theory, 2005.

[34] Yi Zhou, Yingbin Liang, and Huishuai Zhang. Generalization error bounds with probabilistic guarantee for sgd in nonconvex optimization, 2019.

[35] Yunwen Lei and Yiming Ying. Fine-grained analysis of stability and generalization for stochastic gradient descent, 2020.

[36] Ilja Kuzborskij and Christoph H. Lampert. Data-dependent stability of stochastic gradient descent, 2018.

[37] Yuansi Chen, Chi Jin, and Bin Yu. Stability and convergence trade-off of iterative optimization algorithms, 2018.

[38] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 2524–2533. Curran Associates, Inc., 2017.

[39] Daniel Russo and James Zou. Controlling bias in adaptive data analysis using information theory. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1232–1240, Cadiz, Spain, 09–11 May 2016. PMLR.

[40] Jeffrey Negrea, Mahdi Haghifam, Gintare Karolina Dziugaite, Ashish Khisti, and Daniel M. Roy. Information-theoretic generalization bounds for sgld via data-dependent estimates, 2020.

[41] Sharu Theresa Jose and Osvaldo Simeone. Information-theoretic generalization bounds for meta-learning and applications, 2021.

[42] Huan Xu and Shie Mannor. Robustness and generalization, 2010.

[43] Tom Zahavy, Bingyi Kang, Alex Sivak, Jiashi Feng, Huan Xu, and Shie Mannor. Ensemble robustness and generalization of stochastic deep learning algorithms, 2017.

[44] Peter Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks, 2017.

[45] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[46] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. *Introduction to Statistical Learning Theory*, pages 169–207. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[47] J. F. Bonnans and A. Shapiro. Optimization problems with perturbations: A guided tour. *SIAM Rev.*, 40:228–264, 1998.

[48] A. Barakat, P. Bianchi, W. Hachem, and Sh. Schechtman. Stochastic optimization with momentum: convergence, fluctuations, and traps avoidance, 2020.

[49] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[50] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Lemma 1

In our analysis, x_S is the output of an optimization algorithm at a particular time T when the algorithm is used to minimized the empirical risk $F_S(x)$. For convenience, we also denote x_S^* an empirical risk minimizer, i.e. $F_S(x_S^*) = \inf_{x\in\Omega} F_S(x)$. Note that x_S and x_S^* are in general not the same estimator, e.g, in deep learning we can only find local minimums due to non-convexity of the loss landscapes and early stopping is usually employed to terminate optimizers before they reach local minimums. For simplicity, we assume that there is some $x^* \in \Omega$ such that $F(x^*) = \inf_{x\in\Omega} F(x)$.

Recall that $\mathcal{R}(x_S) = F(x_S) - \inf_{x\in\Omega} F(x)$. Given this set up, the excess risk decomposition is usually done in the following manner:

$$\mathcal{R}(x_S) = \underbrace{F(x_S) - F_S(x_S)}_{T_1} + \underbrace{F_S(x_S) - F_S(x^*)}_{T_2} + \underbrace{F_S(x^*) - F(x^*)}_{T_3}. \tag{29}$$

Term T_1 is the generalization of error of the model x_S. Term T_2 is the empirical risk difference between the model x_S and the population risk minimizer x^*. Term T_3 is the generalization error of x^*. By taking the expectation of (29) with respect to the training set S and notice that $\mathbb{E}[F_S(x^*) - F(x^*)] = 0$, we arrive at

$$\begin{aligned}
\mathbb{E}_S[\mathcal{R}(x_S)] &= \mathbb{E}_S[F(x_S) - F_S(x_S)] + \mathbb{E}_S[F_S(x_S) - F_S(x^*)] \\
&= \mathbb{E}_S[F(x_S) - F_S(x_S)] + \mathbb{E}_S[F_S(x_S) - F_S(x_S^*)] + \underbrace{\mathbb{E}_S[F_S(x_S^*) - F_S(x^*)]}_{\leq 0} \\
&\leq \underbrace{\mathbb{E}_S[F(x_S) - F_S(x_S)]}_{\mathcal{E}_{gen}} + \underbrace{\mathbb{E}_S[F_S(x_S) - F_S(x_S^*)]}_{\mathcal{E}_{opt}},
\end{aligned} \tag{30}$$

where $\mathcal{E}_{gen}$ and $\mathcal{E}_{opt}$ are the expected generalization error and the expected optimization error respectively.

B Proof of Lemma 2

Let S and S' be two samples of size n differing in only a single example, and let $z \in Z$ be an arbitrary example. Consider running AOM on sample S and S', respectively. Let $A = \{\delta_{t_0} = 0\}$, we then have that

$$\begin{aligned}
\mathbb{E}|f(x_T; z) - f(x_T'; z)| &= \mathbb{P}(A)\mathbb{E}[|f(x_T; z) - f(x_T'; z)||A] + \mathbb{P}(A^c)\mathbb{E}[|f(x_T; z) - f(x_T'; z)||A^c] \\
&\leq \mu\mathbb{E}[\|x_T - x_T'\||A] + 2\mathbb{P}(A^c)\sup_{x,z}|f(x; z)| \\
&\leq \mu\mathbb{E}[\|x_T - x_T'\||A] + 2M\mathbb{P}(A^c).
\end{aligned}$$

Let i^* be the position where S and S' are different and denote the first time AOM uses the example z_{i^*} by the random variable I. Since $\{I > i_0\} \subset \{\delta_{i_0} = 0\}$, we have that

$$\mathbb{P}(A^c) = \mathbb{P}(\{\delta_{t_0} \neq 0\}) \leq \mathbb{P}(\{I \leq t_0\}) \leq \frac{t_0}{n}.$$

C Proof of Theorem 2

Proof. For any $t > t_0$, we have that

$$\begin{aligned}
\delta_t &= \left\| x_t - \eta_t \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - x_t' + \eta_t \frac{\nabla f(x_t', z_{i_t}')}{\sqrt{v_{t+1}' + \epsilon}} \right\|_2 \\
&\leq \|x_t - x_t'\|_2 + \eta_t \left\| \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x_t', z_{i_t}')}{\sqrt{v_{t+1}' + \epsilon}} \right\|_2 \\
&= \|x_t - x_t'\|_2 + \eta_t R_t,
\end{aligned}$$

where we define

$$R_t = \left\| \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x_t', z_{i_t}')}{\sqrt{v_{t+1}' + \epsilon}} \right\|_2.$$

We consider two cases which depends on the realization of the selected index i_t.

Case 1: With probability $1 - \frac{1}{n}$, we have $z_{i_t} = z'_{i_t}$. Thus,

$$\begin{aligned}
R_t &:= \left\| \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x'_t, z_{i_t})}{\sqrt{v'_{t+1} + \epsilon}} \right\|_2 \\
&\leq \left\| \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x'_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} \right\|_2 + \left\| \frac{\nabla f(x'_t; z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x'_t; z_{i_t})}{\sqrt{v'_{t+1} + \epsilon}} \right\|_2 \\
&= I_t + J_t,
\end{aligned}$$

where we denote

$$\begin{aligned}
I_t &:= \left\| \frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x'_t, z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} \right\|_2, \\
J_t &:= \left\| \frac{\nabla f(x'_t; z_{i_t})}{\sqrt{v_{t+1} + \epsilon}} - \frac{\nabla f(x'_t; z_{i_t})}{\sqrt{v'_{t+1} + \epsilon}} \right\|_2.
\end{aligned}$$

We now derive bounds for both I_t and J_t. We have

$$\begin{aligned}
I_t &= \left[\sum_{j=1}^{d} \left(\frac{\nabla_j f(x_t, z_{i_t})}{\sqrt{v_{t+1,j} + \epsilon}} - \frac{\nabla_j f(x'_t; z_{i_t})}{\sqrt{v_{t+1,j} + \epsilon}} \right)^2 \right]^{1/2} \\
&= \left[\sum_{j=1}^{d} \left(\frac{1}{v_{t+1,j} + \epsilon} \right) (\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t; z_{i_t}))^2 \right]^{1/2} \\
&\leq \left[\sum_{j=1}^{d} \left(\frac{1}{\lambda_1 + \epsilon} \right) (\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t; z_{i_t}))^2 \right]^{1/2} \\
&= \frac{1}{\sqrt{\lambda_1 + \epsilon}} \|\nabla f(x_t, z_{i_t}) - \nabla f(x'_t, z_{i_t})\|_2 \\
&\leq \frac{L}{\sqrt{\lambda_1 + \epsilon}} \|x_t - x'_t\|_2,
\end{aligned}$$

where the first and last inequalities follows from assumptions (3) and (1) respectively.

$$\begin{aligned}
J_t &= \left[\sum_{j=1}^{d} \left(\frac{\nabla_j f(x'_t, z_{i_t})}{\sqrt{v_{t+1,j} + \epsilon}} - \frac{\nabla_j f(x'_t; z_{i_t})}{\sqrt{v'_{t+1,j} + \epsilon}} \right)^2 \right]^{1/2} \\
&= \left[\sum_{i=1}^{d} \nabla_j f(x'_t, z_{i_t})^2 \left(\frac{1}{\sqrt{v_{t+1,i} + \epsilon}} - \frac{1}{\sqrt{v'_{t+1,j} + \epsilon}} \right)^2 \right]^{1/2} \\
&\leq \left[\sum_{i=1}^{d} \mu^2 \left(\frac{1}{\sqrt{v_{t+1,i} + \epsilon}} - \frac{1}{\sqrt{v'_{t+1,j} + \epsilon}} \right)^2 \right]^{1/2} \\
&= \mu \left\| \frac{1}{\sqrt{v_{t+1} + \epsilon}} - \frac{1}{\sqrt{v'_{t+1} + \epsilon}} \right\|_2,
\end{aligned}$$

where the inequality is from assumption (1).

By Lemma 4, we deduce that

$$\begin{aligned}
R_t &\leq \frac{L}{\sqrt{\lambda_1 + \epsilon}} \|x_t - x'_t\|_2 + \mu \left\| \frac{1}{\sqrt{v_{t+1} + \epsilon}} - \frac{1}{\sqrt{v'_{t+1} + \epsilon}} \right\|_2 \\
&\leq \frac{L}{\sqrt{\lambda_1 + \epsilon}} \|x_t - x'_t\|_2 + \frac{\mu}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)} \|v_{t+1} - v'_{t+1}\|_2.
\end{aligned}$$

Since in this case $i_t = i'_t$, we further have that

$$
\begin{aligned}
&\|v_{t+1} - v'_{t+1}\|_2 \\
&= \|\beta_t v_t + (1-\beta_t)\nabla f(x_t, z_{i_t})^2 - \beta_t v'_t - (1-\beta_t)\nabla f(x'_t, z_{i_t})^2\|_2 \\
&\leq \beta_t \|v_t - v'_t\|_2 + (1-\beta_t)\|\nabla f(x_t, z_{i_t})^2 - \nabla f(x'_t, z_{i_t})^2\|_2 \\
&= \beta_t \|v_t - v'_t\|_2 + (1-\beta_t)\left[\sum_{j=1}^{d}\left(\nabla_j f(x_t, z_{i_t})^2 - \nabla_j f(x'_t, z_{i_t})^2\right)^2\right]^{1/2} \\
&= \beta_t \|v_t - v'_t\|_2 + (1-\beta_t)\left[\sum_{j=1}^{d}(\nabla_j f(x_t, z_{i_t}) + \nabla_j f(x'_t, z_{i_t}))^2(\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t, z_{i_t}))^2\right]^{1/2} \\
&\leq \beta_t \|v_t - v'_t\|_2 + (1-\beta_t)2\mu\left[\sum_{j=1}^{d}(\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t, z_{i_t}))^2\right]^{1/2} \\
&= \beta_t \|v_t - v'_t\|_2 + 2\mu(1-\beta_t)\|\nabla f(x_t, z_{i_t}) - \nabla f(x'_t, z_{i_t})\|_2 \\
&\leq \beta_t \|v_t - v'_t\|_2 + 2\mu L(1-\beta_t)\|x_t - x'_t\|_2,
\end{aligned}
$$

where the second inequality follows from the fact that $(a+b)^2 \leq 2(a^2+b^2)$ and the assumption (1). Thus, we deduce that

$$
\begin{aligned}
R_t &\leq \frac{L}{\sqrt{\lambda_1+\epsilon}}\|x_t - x'_t\|_2 + \frac{\mu}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}(\beta_t\|v_t - v'_t\|_2 + 2\mu L(1-\beta_t)\|x_t - x'_t\|_2) \\
&= \frac{L}{\sqrt{\lambda_1+\epsilon}}\|x_t - x'_t\|_2 + \frac{\mu\beta_t}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}\|v_t - v'_t\|_2 + \frac{\mu^2 L(1-\beta_t)}{\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}\|x_t - x'_t\|_2 \\
&= \frac{L}{\sqrt{\lambda_1+\epsilon}}\left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1+\epsilon}\right]\|x_t - x'_t\|_2 + \frac{\mu\beta_t}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}\|v_t - v'_t\|_2.
\end{aligned}
$$

Thus with probability $1 - \frac{1}{n}$, we obtain

$$
\begin{aligned}
R_t \leq \frac{L}{\sqrt{\lambda_1+\epsilon}}&\left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1+\epsilon}\right]\|x_t - x'_t\|_2 \\
&+ \frac{\mu\beta_t}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)}\|v_t - v'_t\|_2.
\end{aligned}
$$

Case 2: With probability $\frac{1}{n}$, we have $z_{i_t} \neq z'_{i_t}$. Thus,

$$
\begin{aligned}
R_t &\leq \left\|\frac{\nabla f(x_t, z_{i_t})}{\sqrt{v_{t+1}+\epsilon}}\right\|_2 + \left\|\frac{\nabla f(x'_t, z'_{i_t})}{\sqrt{v_{t+1}+\epsilon}}\right\|_2 \\
&\leq \left[\sum_{j=1}^{d}\frac{\nabla_j f(x_t, z_{i_t})^2}{v_{t+1,j}+\epsilon}\right]^{1/2} + \left[\sum_{j=1}^{d}\frac{\nabla_j f(x'_t, z'_{i_t})^2}{v'_{t+1,j}+\epsilon}\right]^{1/2} \\
&\leq \frac{[\sum_{j=1}^{d}\nabla_j f(x_t, z_{i_t})^2]^{1/2}}{\sqrt{\lambda_1+\epsilon}} + \frac{[\sum_{j=1}^{d}\nabla_j f(x'_t, z'_{i_t})^2]^{1/2}}{\sqrt{\lambda_1+\epsilon}} \\
&= \frac{\|\nabla f(x_t, z_{i_t})\|_2}{\sqrt{\lambda_1+\epsilon}} + \frac{\|\nabla f(x'_t, z'_{i_t})\|_2}{\sqrt{\lambda_1+\epsilon}} \\
&\leq \frac{2}{\sqrt{\lambda_1+\epsilon}}\mu,
\end{aligned}
$$

where the second inequality follows from assumption (1).

Thus with probability $\frac{1}{n}$, we have

$$
R_t \leq \frac{2}{\sqrt{\lambda_1+\epsilon}}\mu.
$$

Thus, we obtain

$$\delta_{t+1} \leq \delta_t + \eta_t \left(1 - \frac{1}{n}\right) \frac{L}{\sqrt{\lambda_1 + \epsilon}} \left[1 + \frac{\mu^2(1-\beta_t)}{\lambda_1 + \epsilon}\right] \delta_t + \eta_t \left(1 - \frac{1}{n}\right) \frac{\mu\beta_t}{2\sqrt{\lambda_1 + \epsilon}(\lambda_1 + \epsilon)} \sigma_t + \eta_t \frac{1}{n} \frac{2}{\sqrt{\lambda_1 + \epsilon}} \mu.$$

By taking the expectation conditioned over $\delta_{t_0} = 0$ of the above inequality, we get the conclusion of the theorem. □

D Proof of Theorem 3

Proof. For any $t > t_0$, we have that

$$\begin{aligned}
\|v_{t+1} - v'_{t+1}\|_2 &\leq \|\beta_t v_t + (1-\beta_t)\nabla f(x_t; z_{i_t})^2 - \beta_t v'_t - (1-\beta_t)\nabla f(x'_t; z'_{i_t})^2\|_2 \\
&\leq \beta_t \|v_t - v'_t\|_2 + (1-\beta_t)\|\nabla f(x_t; z_{i_t})^2 - \nabla f(x'_t; z'_{i_t})^2\|_2 \\
&= \beta_t \|v_t - v'_t\|_2 + (1-\beta_t) N_t,
\end{aligned}$$

where we define

$$N_t = \|\nabla f(x_t; z_{i_t})^2 - \nabla f(x'_t; z'_{i_t})^2\|_2.$$

We again consider two cases which depends on the realization of the selected index i_t.

Case 1: With probability $1 - \frac{1}{n}$, we have $z_{i_t} = z'_{i_t}$. Thus,

$$\begin{aligned}
N_t &= \|\nabla f(x_t, z_{i_t})^2 - \nabla f(x'_t, z_{i_t})^2\|_2 \\
&= \left[\sum_{j=1}^{d} \left(\nabla_j f(x_t, z_{i_t})^2 - \nabla_j f(x'_t, z_{i_t})^2\right)^2\right]^{1/2} \\
&= \left[\sum_{j=1}^{d} (\nabla_j f(x_t, z_{i_t}) + \nabla_j f(x'_t, z_{i_t}))^2 (\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t, z_{i_t}))^2\right]^{1/2} \\
&\leq 2\mu \left[\sum_{j=1}^{d} (\nabla_j f(x_t, z_{i_t}) - \nabla_j f(x'_t, z_{i_t}))^2\right]^{1/2} \\
&= 2\mu \|\nabla f(x_t, z_{i_t}) - \nabla f(x'_t, z_{i_t})\|_2 \\
&\leq 2L\mu \|x_t - x'_t\|_2,
\end{aligned}$$

where the first inequality follows from the fact that $(a+b)^2 \leq 2(a^2+b^2)$ and the assumption (1).

Thus with probability $1 - \frac{1}{n}$, we have that

$$N_t \leq 2L\mu\delta_t.$$

Case 2: With probability $\frac{1}{n}$, we have $z_{i_t} \neq z'_{i_t}$. Thus

$$\begin{aligned}
N_t &\leq \|\nabla f(x_t, z_{i_t})^2\|_2 + \|\nabla f(x'_t, z'_{i_t})^2\|_2 \\
&= \left[\sum_{j=1}^{d} \nabla_j f(x_t, z_{i_t})^4\right]^{1/2} + \left[\sum_{j=1}^{d} \nabla_j f(x'_t, z'_{i_t})^4\right]^{1/2} \\
&\leq \sum_{j=1}^{d} \nabla_j f(x_t, z_{i_t})^2 + \sum_{j=1}^{d} \nabla_j f(x'_t, z'_{i_t})^2 \\
&= \|\nabla f(x_t, z_{i_t})\|_2^2 + \|\nabla f(x'_t, z'_{i_t})\|_2^2 \\
&\leq 2\mu^2,
\end{aligned}$$

where the first inequality follows from the fact that for $a_1, \ldots, a_d \geq 0$, $(a_1 + \ldots + a_d)^2 \geq a_1^2 + \ldots + a_d^2$.

Thus with probability $\frac{1}{n}$, we have that

$$N_t \leq 2\mu^2.$$

By combining the above two cases, we obtain

$$\sigma_{t+1} \leq \beta_t \sigma_t + (1-\beta_t)\left(1-\frac{1}{n}\right) 2L\mu\delta_t + \frac{1}{n}(1-\beta_t)2\mu^2.$$

Taking the expectation conditioned over $\delta_{t_0} = 0$ of the above inequality, we get the conclusion of the theorem. □

E Proof of Lemma 3

Proof. By substituting $\eta_t = \frac{c}{t}$ and $\beta_t = 1 - \alpha_t$ into the matrix A_t, we have that

$$A_t = \begin{bmatrix} 1 + \frac{c}{t}U_t & \frac{c}{t}V_t \\ \alpha_t W & 1-\alpha_t \end{bmatrix}$$

For any $v = (v_1, v_2) \in \mathbb{R}^2$ such that $v_1^2 + v_2^2 = 1$, we have that

$$\begin{aligned}
\|A_t v\|_2^2 &= \left\| \begin{matrix} v_1 + \frac{c}{t}U_t v_1 + \frac{c}{t}V_t v_2 \\ \alpha_t W v_1 + v_2 - \alpha_t v_2 \end{matrix} \right\|_2^2 \\
&= \left(v_1 + \frac{c}{t}U_t v_1 + \frac{c}{t}V_t v_2\right)^2 + \left(\alpha_t W v_1 + v_2 - \alpha_t v_2\right)^2 \\
&= v_1^2 + 2\frac{cv_1}{t}(U_t v_1 + V_t v_2) + \frac{c^2(U_t v_1 + V_t v_2)^2}{t^2} \\
&\qquad + v_2^2 + 2v_2\alpha_t(Wv_1 - v_2) + \alpha_t^2(Wv_1 - v_2)^2 \\
&\leq 1 + \frac{2c}{t}\sqrt{U_t^2 + V_t^2} + c^2\frac{U_t^2 + V_t^2}{t^2} + 2\alpha_t\sqrt{W_t^2 + 1} + \alpha_t^2(W_t^2 + 1) \\
&\leq 1 + \frac{2c}{t}\sqrt{U^2 + V^2} + c^2\frac{U^2 + V^2}{t^2} + 2\alpha_t\sqrt{W^2 + 1} + \alpha_t^2(W^2 + 1) \\
&\leq 1 + \frac{2c}{t}\sqrt{U^2 + V^2} + c^2\frac{U^2 + V^2}{t^2} + \alpha_t\left(2\sqrt{W^2 + 1} + (W^2 + 1)\right) \\
&\leq exp\left[\frac{2c}{t}\sqrt{U^2 + V^2} + c^2\frac{U^2 + V^2}{t^2} + \alpha_t\left(2\sqrt{W^2 + 1} + (W^2 + 1)\right)\right].
\end{aligned}$$

Where we have used the fact that $|v_1|, |v_2| \leq 1$, the Cauchy–Schwarz inequality and $1 + x \leq e^x$ for all $x > 0$.

Thus we deduce that

$$\begin{aligned}
\|A_t\|_2 &\leq \left\{exp\Big(\frac{2c}{t}\sqrt{U^2 + V^2} + c^2\frac{U^2 + V^2}{t^2} + \alpha_t\left(2\sqrt{W^2 + 1} + (W^2 + 1)\right)\Big)\right\}^{1/2} \\
&= exp\left(\frac{c}{t}\sqrt{U^2 + V^2} + c^2\frac{U^2 + V^2}{2t^2} + \alpha_t\left(\sqrt{W^2 + 1} + \frac{1}{2}(W^2 + 1)\right)\right).
\end{aligned}$$

Denote $D_1 = U^2 + V^2, D_2 = \sqrt{W^2 + 1} + \frac{1}{2}(W^2 + 1)$, we can then rewrite

$$\|A_t\|_2 \leq exp\left(\frac{c}{t}\sqrt{D_1} + \frac{c^2}{2t^2}D_1 + \alpha_t D_2\right).$$

Thus, we obtain the desired conclusion of the lemma. □

F Proof of Theorem 4

Proof. Denote

$$H_t := \left\| \begin{bmatrix} \Delta_t \\ \Sigma_t \end{bmatrix} \right\|_2$$

By taking the norm on both sides of the relation (4), we obtain

$$\begin{aligned}
H_{t+1} &\leq \|A_t\| H_t + \frac{1}{n}\sqrt{\frac{c^2 Y^2}{t^2} + \alpha_t^2 Z^2} \\
&\leq exp\left(\frac{c}{t}\sqrt{D_1} + \frac{c^2}{t^2}\frac{D_1}{2} + \alpha_t D_2\right) H_t + \frac{1}{n}\left(\frac{c}{t}Y + \alpha_t Z\right).
\end{aligned}$$

By expanding the above inequality, for any $T > t_0$ we have

$$
\begin{aligned}
H_T &\leq \sum_{t=t_0+1}^{T} \left[\prod_{k=t+1}^{T} exp\left(\frac{c}{k}\sqrt{D_1} + \frac{c^2}{k^2}\frac{D_1}{2} + \alpha_k D_2 \right) \right] \times \frac{1}{n}\left(\frac{c}{t} Y + \alpha_t Z \right) \\
&= \frac{c}{n} Y \sum_{t=t_0+1}^{T} \left[\prod_{k=t+1}^{T} exp\left(\frac{c}{k}\sqrt{D_1} + \frac{c^2}{k^2}\frac{D_1}{2} + \alpha_k D_2 \right) \right] \frac{1}{t} \\
&\qquad + \frac{1}{n} Z \sum_{t=t_0+1}^{T} \left[\prod_{k=t+1}^{T} exp\left(\frac{c}{k}\sqrt{D_1} + \frac{c^2}{k^2}\frac{D_1}{2} + \alpha_k D_2 \right) \right] \alpha_t \\
&\leq \frac{c}{n} Y \sum_{t=t_0+1}^{T} exp\left(c\sqrt{D_1} \sum_{k=t+1}^{T} \frac{1}{k} + \frac{c^2 D_1}{2} \sum_{k=t+1}^{T} \frac{1}{k^2} + D_2 \sum_{k=t+1}^{T} \alpha_t \right) \frac{1}{t} \\
&\qquad + \frac{1}{n} Z \sum_{t=t_0+1}^{T} exp\left(c\sqrt{D_1} \sum_{k=t+1}^{T} \frac{1}{k} + \frac{c^2 D_1}{2} \sum_{k=t+1}^{T} \frac{1}{k^2} + D_2 \sum_{k=t+1}^{T} \alpha_t \right) \alpha_t \\
&= \frac{c}{n} Y \sum_{t=t_0+1}^{T} exp\left(c\sqrt{D_1} \log\left(\frac{T}{t} \right) + \gamma \frac{c^2 D_1}{2} + D_2 \sum_{k=t+1}^{T} \alpha_t \right) \frac{1}{t} \\
&\qquad + \frac{1}{n} Z \sum_{t=t_0+1}^{T} exp\left(c\sqrt{D_1} \log\left(\frac{T}{t} \right) + \gamma \frac{c^2 D_1}{2} + D_2 \sum_{k=t+1}^{T} \alpha_t \right) \alpha_t \\
&= \frac{c}{n} Y exp\left(\gamma \frac{c^2 D_1}{2} \right) \times \sum_{t=t_0+1}^{T} exp\left(D_2 \sum_{k=t+1}^{T} \alpha_t \right) \left(\frac{T}{t} \right)^{c\sqrt{D_1}} \frac{1}{t} \\
&\qquad + \frac{1}{n} Z exp\left(\gamma \frac{c^2 D_1}{2} \right) \times \sum_{t=t_0+1}^{T} exp\left(D_2 \sum_{k=t+1}^{T} \alpha_t \right) \left(\frac{T}{t} \right)^{c\sqrt{D_1}} \alpha_t
\end{aligned}
$$

where we define

$$\gamma = \sum_{k=1}^{\infty} \frac{1}{k^2}.$$

Thus, we obtain

$$H_t \leq \frac{1}{n} exp\left(\gamma \frac{c^2 D_1}{2} \right) \times \sum_{t=t_0+1}^{T} exp\left(D_2 \sum_{k=t+1}^{T} \alpha_t \right) \left(\frac{T}{t} \right)^{c\sqrt{D_1}} \times \left(\frac{1}{t} c Y + \alpha_t Z \right)$$

which the conclusion of the theorem. □

G Proof of Corollary 2

Proof. By the assumption, we have

$$\begin{aligned}
H_t &\leq \frac{1}{n} exp\left(\gamma \frac{c^2 D_1}{2}\right) \times \sum_{t=t_0+1}^{T} exp\left(D_2 \alpha \sum_{k=t+1}^{T} \frac{1}{k}\right) \left(\frac{T}{t}\right)^{c\sqrt{D_1}} \left(\frac{1}{t} cY + \frac{1}{t} \alpha Z\right) \\
&= \frac{1}{n} exp\left(\gamma \frac{c^2 D_1}{2}\right) \sum_{t=t_0+1}^{T} \left(\frac{T}{t}\right)^{\alpha D_2} \left(\frac{T}{t}\right)^{c\sqrt{D_1}} \left(\frac{1}{t} cY + \frac{1}{t} \alpha Z\right) \\
&= \frac{1}{n} (cY + \alpha Z) exp\left(\gamma \frac{c^2 D_1}{2}\right) \sum_{t=t_0+1}^{T} \left(\frac{T}{t}\right)^{c\sqrt{D_1}+\alpha D_2} \frac{1}{t} \\
&= \frac{1}{n} (cY + \alpha Z) exp\left(\gamma \frac{c^2 D_1}{2}\right) T^{c\sqrt{D_1}+\alpha D_2} \sum_{t=t_0+1}^{T} t^{-c\sqrt{D_1}-\alpha D_2-1} \\
&\leq \frac{1}{n} (cY + \alpha Z) exp\left(\gamma \frac{c^2 D_1}{2}\right) T^{c\sqrt{D_1}+\alpha D_2} \int_{t_0}^{T} x^{-c\sqrt{D_1}-\alpha D_2-1} dx \\
&\leq \frac{1}{n} (cY + \alpha Z) exp\left(\gamma \frac{c^2 D_1}{2}\right) \frac{T^{c\sqrt{D_1}+\alpha D_2}}{c\sqrt{D_1} + \alpha D_2} \frac{1}{t_0^{c\sqrt{D_1}+\alpha D_2}}.
\end{aligned}$$

□

H Proof of Corollary 3

Proof. By Lemma 2, we have that

$$\begin{aligned}
R(f(.,z)) &:= \mathbb{E}|f(x_T; z) - f(x'_T; z)| \\
&\leq 2M \frac{t_0}{n} + \mu \Delta_T \\
&\leq 2M \frac{t_0}{n} + \frac{1}{n} (cY + \alpha Z) \times exp\left(\gamma \frac{c^2 D_1}{2}\right) \frac{T^{c\sqrt{D_1}+\alpha D_2}}{c\sqrt{D_1} + \alpha D_2} \frac{\mu}{t_0^{c\sqrt{D_1}+\alpha D_2}}
\end{aligned}$$

Denote

$$\begin{aligned}
A &:= c\sqrt{D_1} + \alpha D_2 \\
B &:= (cY + \alpha Z) \times exp\left(\gamma \frac{c^2 D_1}{2}\right) \frac{\mu}{c\sqrt{D_1} + \alpha D_2}
\end{aligned}$$

We then can rewrite

$$R(f(.,z)) \leq \frac{1}{n} \left[2M t_0 + B \left(\frac{T}{t_0}\right)^A\right]$$

We want to choose t_0 to minimize the right hand-side. We consider the function

$$g : x \mapsto 2Mx + B \left(\frac{T}{x}\right)^A$$

We have

$$g'(x) = 2M - AB \frac{T^A}{x^{A+1}}$$

Thus the function is approximately minimize when

$$t_0 = \left(\frac{AB}{2M}\right)^{\frac{1}{A+1}} T^{\frac{A}{A+1}}$$

Substitute this t_0 into the stability bound, we obtain

$$R(f(.,z)) \leq \frac{1}{n} \left[2M \left(\frac{AB}{2M}\right)^{\frac{1}{A+1}} + \frac{B}{\left(\frac{AB}{2M}\right)^{\frac{A^2}{A+1}}}\right] T^{\frac{A}{A+1}}.$$

□

I Supportive lemmas

Lemma 4. *Let $f(.,z)$ be μ-Lipschitz and L-smooth for all z. Assume that the assumption (3) is satisfied. For any $t \in \mathbb{N}$, we then have*

$$K := \left\| \frac{1}{\sqrt{v_{t+1}+\epsilon}} - \frac{1}{\sqrt{v'_{t+1}+\epsilon}} \right\|_2 \leq \frac{1}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)^2} \|v_{t+1} - v'_{t+1}\|_2$$

Proof. We have

$$\begin{aligned}
K &= \left[\sum_{j=1}^{d} \left(\frac{1}{\sqrt{v_{t+1,j}+\epsilon}} - \frac{1}{\sqrt{v'_{t+1,j}+\epsilon}} \right)^2 \right]^{1/2} \\
&= \left[\sum_{j=1}^{d} \left(\frac{\sqrt{v_{t+1,j}+\epsilon} - \sqrt{v'_{t+1,j}+\epsilon}}{\sqrt{v_{t+1,j}+\epsilon}\sqrt{v'_{t+1,j}+\epsilon}} \right)^2 \right]^{1/2} \\
&\leq \left[\sum_{j=1}^{d} \frac{\left(\sqrt{v_{t+1,j}+\epsilon} - \sqrt{v'_{t+1,j}+\epsilon} \right)^2}{(\lambda_1+\epsilon)(\lambda_1+\epsilon)} \right]^{1/2} \\
&= \frac{1}{\lambda_1+\epsilon} \left[\sum_{i=1}^{d} \left(\sqrt{v_{t+1,j}+\epsilon} - \sqrt{v'_{t+1,j}+\epsilon} \right)^2 \right] \\
&= \frac{1}{\lambda_1+\epsilon} \left[\sum_{j=1}^{d} \frac{(v_{t+1,j} - v'_{t+1,j})^2}{\left(\sqrt{v_{t+1,j}+\epsilon} + \sqrt{v'_{t+1,j}+\epsilon} \right)^2} \right]^{1/2} \\
&\leq \frac{1}{2\sqrt{\lambda_1+\epsilon}(\lambda_1+\epsilon)} \|v_{t+1} - v'_{t+1}\|_2.
\end{aligned}$$

□

J Additional Training Parameters for Synthetic Data

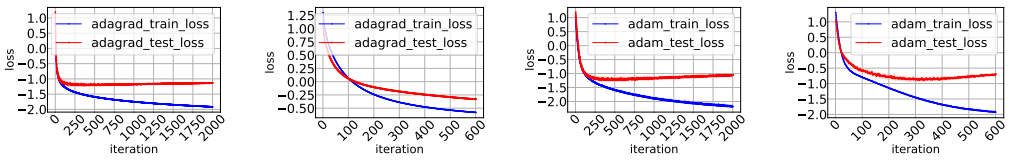

Figure 6: Train loss and test loss of CLS and REG tasks for Adagrad (left) and Adam (right).

For CLS training, we use a shallow neural network with one hidden fully-connected layer, which has 1024 neurons and an output of 3 classes. For the last layer, we use cross-entropy loss. For optimizser, we use Adagrad with learning rate 0.001. For comparison, we also run Adam with learning rate 0.0001 and $\beta_1 = 0, \beta_2 = 0.999$. A batch size of 3 is used in this task.

Compared to CLS task, we only use 128 neurons for the hidden layer and MSE loss and batch size of 5 in REG task. In this case, we use Adam with learning rate 0.001 and $\beta_1 = 0, \beta_2 = 0.999$.

Finally, for each experiment, we run 20 trials and then calculate the mean and standard deviation before plotting the results.

K Additional Training Parameters for Real Data

For the Cifar10 classification task, we train the model using 1 GPU with batch size 32. Similar to the synthetic setting, For each setting, we run 20 trials and for each trial, we run 60 epochs to calculate means and standard deviations. In terms of weight initialization for convolution and fully-connected layers of VGG11, we use He initialization method ("fan-out" mode for convolution). For more details, please see the code attached to this supplemental material.