On the Convergence of Step Decay Step-Size for Stochastic Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

The convergence of stochastic gradient descent is highly dependent on the step-size, especially on non-convex problems such as neural network training. Step decay step-size schedules (constant and then cut) are widely used in practice because of their excellent convergence and generalization qualities, but their theoretical properties are not yet well understood. We provide convergence results for step decay in the non-convex regime, ensuring that the gradient norm vanishes at an $\mathcal{O}(\ln T/\sqrt{T})$ rate. We also provide near-optimal (and sometimes provably tight) convergence guarantees for general, possibly non-smooth, convex and strongly convex problems. The practical efficiency of the step decay step-size is demonstrated in several large-scale deep neural network training tasks.

1 Introduction

We focus on stochastic programming problems on the form

$$\min_{x\in\mathcal{X}} \; f(x) := \mathbb{E}_{\xi\sim\Xi}[f(x;\xi)]. \tag{1}$$

Here, ξ is a random variable drawn from some source distribution Ξ over an arbitrary probability space and $\mathcal{X}$ is a closed, convex subset of $\mathbb{R}^d$. This problem is often encountered in machine learning applications, such as training of deep neural networks. Depending on the specifics of the application, the function f can either non-convex, convex or strongly convex; it can be smooth or non-smooth; and it may also have additional structure that can be exploited.

Despite the many advances in the field of stochastic programming, the stochastic gradient descent (SGD) method [38, 33] remains important and is arguably still the most popular method for solving (1). The SGD method updates the decision vector x using the following recursion

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta_t \hat{g}_t) \tag{2}$$

where $\hat{g}_t$ is an unbiased estimate of the gradient (or subgradient) at x_t and $\Pi_{\mathcal{X}}$ is the Euclidean projection onto $\mathcal{X}$. The step-size parameter (learning-rate) $\eta_t > 0$ is critical to control the rate at which the model learns and to guarantee that the SGD iterates converge to an optimizer of (1). Setting the step-size too large will result in iterates which never converge; and setting it too small leads to slow convergence and may even cause the iterates to get stuck at bad local minima. As long as the iterates do not diverge, a large constant step-size promotes fast convergence but only to a (large) neighborhood of the optimal solution. To increase the accuracy, we have to decrease the step-size.

The traditional approach is to decrease the step-size in every iteration, typically as η_0/t or $\eta_0/\sqrt{t}$. Both these step-size schedules have been studied extensively and guarantee a non-asymptotic convergence of SGD [32, 27, 37, 20, 39, 16]. However, from a practical perspective, these step-size policies often perform poorly, since they begin to decrease too early. For non-convex problems, such

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

as those which arise in training of deep neural networks, the most successful and popular step-size policy in practice is the step decay step-size [26, 21, 23]. This step-size policy starts with a relatively large constant step-size and then cuts the step-size by a fixed number (called decay factor) at after a given number of epochs. Not only does this step-size result in a faster initial convergence, but it also guarantees that the SGD iterates eventually converge to an exact solution.

The step decay step-size is the default choice in many deep learning libraries, such as TensorFlow [4] and PyTorch [36]; both use a decay rate of 0.1 and user-defined milestones when the step-size is decreased. Although some recent studies attempt to monitor the optimization process to trigger the milestones when certain conditions are met [28, 48], such approaches are difficult to analyze. Another common approach is to simply divide the targeted number of iterations into N intervals of equal length, and trigger a milestone at the end of each such interval. However, the theoretical properties of this simpler step-size policy is also not yet well understood. To make the problem more manageable theoretically, some related step-size policies that let the length of each segment increase linearly [8] or exponentially [20, 9, 47] are analyzed. On the contrary, we consider the more practical case when the horizon T of the step-decay step-size is divided into several equal parts.

If the number of iterations or horizon T is known apriori, [12] analyze a step decay step-size with decay rate $1/2$ applied every $T/\log_2 T$ iterations and establish a near-optimal $\mathcal{O}(\log_2 T/T)$ convergence rate for vanilla SGD on least-squares problems. However, to our knowledge, there are no convergence guarantees in the literature under more general conditions, e.g., for general strongly-convex, general convex, or non-convex problems. Motivated by this, in this paper, we study the non-asymptotic convergence of SGD with the step decay step-size in more general settings.

1.1 Main Contributions

This work establishes novel convergence guarantees for SGD with the step decay step-size on non-convex, convex and strongly convex optimization problems. We make the following contributions:

- We propose a non-uniform probability rule $P_t \propto 1/\eta_t$ for selecting the output in the smooth non-convex setting. Based on this rule, we **(i)** establish a near-optimal $\mathcal{O}(\ln T/\sqrt{T})$ rate for SGD with step decay step-size; **(ii)** improve the results for exponential decay step-size [29]; **(iii)** remove the $\ln T$ factor in the best known convergence rate for the $1/\sqrt{t}$ step-size.
- For strongly convex problems, we establish the following error bounds for the last iterate under step-decay: **(i)** $\mathcal{O}(\ln T/T)$ for smooth problem, which we also prove to be tight; **(ii)** $\mathcal{O}(\ln^2 T/T)$ without the smoothness assumption.
- For the general convex case, we prove that the step decay step-size at the last iterate can achieve a near-optimal convergence rate (up to a $\ln T$ factor).

1.2 Related Work

For SGD, the best known bound for the expected error of the Tth iterate is of $\mathcal{O}(1/\sqrt{T})$ when the objective is convex and smooth with Lipschitz continuous gradient [33, 13], and of $\mathcal{O}(1/T)$ when the objective is also strongly convex [32, 37]. Without any further assumptions, these rates are known to be optimal. If we restrict our attention to diminshing step-sizes, $\eta_t = \eta_0/t$, the best known error bound for strongly convex and nonsmooth problems is of $\mathcal{O}(\ln T/T)$ [39], which is also tight [18]. This rate can be improved to $\mathcal{O}(1/T)$ by averaging strategies [37, 27, 39] or a step decay step-size [20]. For smooth nonconvex functions, [13] established an $\mathcal{O}(1/\sqrt{T})$ rate for SGD with constant step-size $\eta_t = \mathcal{O}(1/\sqrt{T})$. Recently, [10] has proven that this error bound is tight up to a constant, unless additional assumptions are made.

Step decay step-sizes were probably first considered for deterministic subgradient methods [15] and [40]. More recently step decay step-size schemes have been employed to improve the convergence rates for under various conditions: local growth (convex) [44], Polyak-Lójasiewicz (PL) condition [47], sharp growth (non-convex) [9]. Most of these references consider proximal algorithms (which add a quadratic term $\lambda \|\cdot\|^2$ to the original loss function) and, in addition, perform an extra averaging step between stages. In particular, if the function f is μ-weakly convex (i.e. $f + \frac{\mu}{2}\|\cdot\|^2$ is convex), λ is required to be larger than $\frac{\mu}{2}$ [47, 8]. In contrast, we study the performance of step decay step-size for the standard SGD algorithm. Moreover, in [44] the output of each inner-loop is a

uniform average. So their complexity reflects the average performance over iterations. But for convex and strongly convex problems, we focus on the final iterate which is the most preferred choice in practice. In fact, most practitioners just use the final iterate.

Reference [8] considers slowly decaying step-sizes, $\sim 1/t$ where t is the stage index. In other words, the step-size is not cut by some constant factor between stages as we do, but it decays (very) slowly. Compared to [20] and [47], where the inner-loop size S grows exponentially, we use a constant value of S, which is known to work better in practice. In the extreme case when $S = 1$, step-decay reduces to the exponentially decaying step-size recently studied under the PL condition and a general smoothness assumption [29].

A number of adaptive step-size selection strategies have been proposed for SGD (*e.g.*, [11, 42, 24, 31, 30]), some of which result in step-decay policies [43, 28, 48]. For example, [28] develop a statistical procedure to automatically determine when the SGD iterates with a constant step-size no longer make progress, and then halve the step-size. Empirically, this automatic scheme is competitive with the best hand-tuned step-decay schedules, but no formal guarantees for this observed behaviour are given.

The paper is organized as follows. Notation and basic definitions are introduced in Section 2. In Section 3, we analyze the convergence rates of step decay step-size on nonconvex case and propose a novel non-uniform sampling rule for the algorithm output. The convergence for general convex and strongly convex functions are investigated in Sections A and 4, respectively. Numerical results of our algorithms are presented and discussed in Section 5. Finally, conclusions are made in Section 6.

2 Preliminaries

In this part, we will give some definitions and notations used throughout the paper.

Definition 1. ***(1)*** *The stochastic gradient oracle $\tilde{\mathcal{O}}$ is variance-bounded if for any input vector $\hat{x}$,it returns a random vector $\hat{g}$ such that $\mathbb{E}[\|\hat{g} - \mathbb{E}[\hat{g}]\|^2] \leq V^2$.* ***(2)*** *$\tilde{\mathcal{O}}$ is bounded if for any input vector $\hat{x}$, it returns a random vector $\hat{g}$ such that $\mathbb{E}[\|\hat{g}\|^2] \leq G^2$ for some fixed $G > 0$.*

Definition 2. *When the function f is differentiable on $\mathcal{X}$, we say that f is L-smooth on $\mathcal{X}$ if there exists a constant $L > 0$ such that $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$. It also implies that $f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|x - y\|^2$ for any $x, y \in \mathcal{X}$. If f is not differentiable on $\mathcal{X}$, we say that f is L-smooth with respect to x^* if $f(x) - f(x^*) \leq \frac{L}{2}\|x - x^*\|^2, \forall x \in \mathcal{X}$, with $L > 0$.*

The definition of L-smoothness for non-convex functions has been considered by [37].

Definition 3. *The function f is μ-strongly convex on $\mathcal{X} \subseteq \mathbb{R}^d$ if $f(y) \geq f(x) + \langle g, y - x\rangle + \frac{\mu}{2}\|y - x\|^2, \forall x, y \in \mathcal{X}, g \in \partial f(x)$, with $\mu > 0$.*

Definition 4. *The function f is convex on $\mathcal{X}$ if $f(y) \geq f(x) + \langle g, y - x\rangle$ for any $g \in \partial f(x)$ and $x, y \in \mathcal{X}$.*

Throughout the paper, we assume the objective function f is bounded below on $\mathcal{X}$ and let f^* denote its infimum. If f is strongly convex, x^* is the unique minimum point of f and $f^* = f(x^*)$.

Notation: Let $[n]$ denote the set of $\{1, 2, \cdots, n\}$ and $\|\cdot\| := \|\cdot\|_2$ without specific mention. We use $\lfloor r \rfloor$ and $\lceil r \rceil$ to denote the nearest integer to the real number r from below and above. For simplicity of notation, we assume that S, N, $\log_\alpha T/2$, $\log_\alpha T$ and $T/\log_\alpha T$ are all integers.

3 Convergence for Nonconvex Problems

In this section, we provide the first convergence bounds for SGD with step-decay step-sizes on non-convex problems. We also show that our technical approach can be used to improve the best known convergence bounds for both i) standard $1/\sqrt{t}$ step-sizes and ii) exponential decay step-sizes. Before we proceed, we illustrate the theoretical novelty that allows us to derive these results. Most convergence results for SGD on non-convex problems analyze a random iterate drawn from $\{x_t\}_{t=1}^T$ with some probability P_t [13, 14, 29]. For example, [13] provides the following result.

Proposition 3.1. *Suppose that f is L-smooth on $\mathbb{R}^d$ and the stochastic gradient oracle is variance-bounded by V^2. If the step-size $\eta_t = \eta_0/\sqrt{t} \leq 1/L$, then*

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \leq \frac{f(x_1) - f^*}{\eta_0(\sqrt{T}-1)} + \frac{LV^2\eta_0(\ln T+1)}{2(\sqrt{T}-1)}, \tag{3}$$

where $\hat{x}_T$ is randomly chosen from $\{x_t\}_{t=1}^T$ with probability $P_t \propto \eta_t$. [1]

Since η_t is decreasing, $P_t \propto \eta_t$ means that initial iterates are given higher weights in the average in Equation (3) than the final iterates. This contradicts the intuition that the iterates become better as the algorithm progresses. Ideally, we should do the opposite, *i.e.* put high weights on the final iterates and low weights on the initial iterates. This is exactly what we do to obtain convergence bounds for step decay step-size: we use the probability $P_t \propto 1/\eta_t$ instead of $P_t \propto \eta_t$ [13, 29] or uniformly sampling [9]. This is especially important when the step-sizes decrease exponentially fast, like in step decay and exponential decay step-sizes. For example, suppose that $\eta_t \propto 0.9^{-t}$ and $T = 100$. Then with $P_t \propto \eta_t$ we pick the output from the first 10 iteration with 65% probability. On the other hand, with $P_t \propto 1/\eta_t$ we pick the output from the last 10 iterations with 65% probability; see Figure 1.

There are other averaging or sampling rules which favor the last iterate in a similar spirit as we do. References [27] and [17] consider SGD with a $1/t$ step-size and analyze the increasing weighted average $\sim t^r$ $(r > 0)$ of the iterates. [8, 45] selects a random iterate as output with increasing probabilities, $\sim t^r$ $(r > 0)$ for stagewise optimization algorithms, but their step-sizes and output probabilities are very different from ours. Especially, for step-decay, their output probabilities are on the form t^r $(r > 0)$, while we use α^t $(\alpha > 1)$ which increases much quicker with t, hence our sampling rule favours late iterates to a much higher degree. Moreover, our output distribution is directly related to the step-size, and does not include any new parameters that require hyper-tuning.

In the following three subsections we perform such an analysis to 1) provide convergence bounds for step decay step-size (where no bounds existed before) 2) provide improved convergence rate results for exponential decay step-size, and 3) improve the convergence bound for $1/\sqrt{t}$ step-size.

3.1 Convergence Rates under the Step Decay Step-size

Algorithm 1 describes the SGD algorithm with step decay step-size and our probability rule for selecting the output. Here, and in the analysis below, N is the number of stages, and α is the decay factor by which the step-size is divided at the end of each such stage. In practical deep neural network training, $\alpha = 10$ is the default and N is typically a hyper-parameter selected by experience. For nonconvex problems, we will show that it makes sense to let N depend on the desired number of iterations T and the decay factor α. Specifically, we will derive results for $N = \log_\alpha T/2$ (which is typically a small number) and show that this choice works well in numerical experiments.

A technical challenge that appears in the analysis of our novel sampling rule is that we have a new term $\sum_{t=1}^T \mathbb{E}[f(x_t) - f(x_{t+1})]/\eta_t^2$ to account for. To this end, we assume that the objective function f is upper bounded by $f_{\max}$, which also made in many previous works [6, 5, 22, 46, 8]. Note that this assumption does not imply that the random loss function $f(x;\xi)$ for $\xi \in \Xi$ is bounded.

Theorem 3.2. *Suppose that the non-convex objective function f is L-smooth on $\mathbb{R}^d$ and upper bounded by $f_{\max}$*[2] *and the stochastic gradient oracle is variance-bounded by V^2. If we run Algorithm 1 with $T > 1$, $\alpha > 1$, $N \geq 1$, $\eta_0 \leq 1/L$, and $x_1^1 \in \mathbb{R}^d$, then*

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \leq \frac{2(\alpha-1)\Delta_0}{\alpha^2\eta_0} \cdot \frac{N\alpha^N}{T(1-\alpha^{-N})} + \frac{\eta_0 LV^2(\alpha-1)}{\alpha^N - 1}. \tag{4}$$

where $\Delta_0 = f_{\max} - f^*$. Furthermore, if we assume that $N = \log_\alpha T/2$, then

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \leq A\frac{\Delta_0}{\eta_0} \cdot \frac{\ln T}{\sqrt{T}-1} + B\frac{LV^2\eta_0}{\sqrt{T}-1}, \tag{5}$$

where $A = (\alpha-1)/(\alpha^2 \ln \alpha)$ and $B = \alpha - 1$.

[1] In [13], $P_t \propto (2\eta_t - L\eta_t^2)$. If η_t is far smaller than $1/L$, we have $(2\eta_t - L\eta_t^2) \approx 2\eta_t$. For simplicity, we rewrite the probability as $P_t \propto \eta_t$ to show the results.

[2] The function f is upper bounded by $f_{\max}$ if $f(x) \leq f_{\max}$ for any $x \in \mathbb{R}^d$.

Algorithm 1 SGD with step decay step-size on nonconvex case

Input: initial point x_1^1, initial step-size η_0, decay factor $\alpha > 1$, the number of iterations T, outer-loop size N, inner-loop size $S = T/N$
for $t = 1 : N$ **do**
  $\eta_t = \eta_0/\alpha^{t-1}$
  for $i = 1 : S$ **do**
    Query a stochastic gradient oracle at x_i^t to get a vector $\hat{g}_i^t$ such that $\mathbb{E}[\hat{g}_i^t] = \nabla f(x_i^t)$
    $x_{i+1}^t = x_i^t - \eta_t \hat{g}_i^t$
  end for
  $x_1^{t+1} = x_{S+1}^t$
end for
Return: $\hat{x}_T$ is randomly chosen from all the previous iterations $\{x_i^t\}$ with probability $P_i^t = \frac{1/\eta_t}{S\sum_{t=1}^N 1/\eta_t}$ where $i \in [S]$ and $t \in [N]$

The theorem establishes the first convergence guarantee of SGD with step decay step-sizes on non-convex problems. The results in Eq. (4) capture the influence of N (the number of stages) on the convergence rate. For example, if $N = 1$, then the step-decay schedule reduces to the constant step-size and the convergence bound in Theorem 3.2 matches the known results by setting $\eta_t = \eta_0/\sqrt{T}$ [13]. For $N > 1$, the error bound becomes more complicated. Increasing N reduces the noise term, but it increases the error related to the function value. So there is a trade-off between making the two terms in the right hand side of Eq. (4) small.

The convergence rate is optimized by selecting $N = \log_\alpha /2$, which yields the rate given in Eq. (5). It ensures a $\mathcal{O}(\ln T/\sqrt{T})$ rate towards a stationary point, which is comparable to the results for $\eta_t = \mathcal{O}(1/\sqrt{t})$ step-size in [13] or Proposition 3.1. However, as illustrated in our experiments in Section 5, step decay step-size converges faster in practice and tends to find stationary points that generalize better. Moreover, in the deterministic case when $V = 0$, let $N = 1$, Theorem 3.2 yields the standard $\mathcal{O}(1/T)$ convergence rate result for deterministic gradient descent [34, 7, 14]. Another benefit of Theorem 3.2 is that we can make the convergence bound unaffected by the noise variance V^2 by setting $\alpha = 1 + 1/V^2$. This is in contrast to Proposition 3.1 and $1/\sqrt{t}$ step-sizes, where an increased V^2 results in a worse upper bound.

3.2 A Special Case: Exponentially Decaying Step-Size

An interesting special case occurs when we set $S = 1$ in the step decay step-size. In this case, the step-size will decay exponentially, $\eta_t = \eta_0/\alpha^t$. The first convergence rate results for such step sizes, which we will call Exp-Decay, have only recently appeared in the preprint [29]; we compare our work to those results below.

Clearly, for $S = 1$, the exponential decay factor $\alpha > 1$ cannot be chosen arbitrarily: if α is too large, then η_t will vanish in only a few iterations. To avoid this, we choose a similar form of the decaying factor as [29] and set $\alpha = (\beta/T)^{-1/T}$, where $\beta \in [1, T)$. The intuition is that at the final iteration T the step-size is on the order of $1/\sqrt{T}$ (setting $\beta = \mathcal{O}(\sqrt{T})$), and thus does not vanish over the T iterations. We are now ready to prove the algorithm's convergence.

Theorem 3.3. *Suppose that the non-convex objective function f is L-smooth and upper bounded by $f_{\max}$, and that the stochastic gradient oracle is variance-bounded by V^2. If we run Algorithm 1 with $T > 1$, $S = 1$, $\eta_0 \le 1/L$, $x_1^1 \in \mathbb{R}^d$, and $\alpha = (\beta/T)^{-1/T}$, then*

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \le \frac{\eta_0 \ln(\frac{T}{\beta})}{(\frac{T}{\beta} - 1)T}\left[\frac{2\Delta_0}{\eta_0^2}\left(\frac{T}{\beta}\right)^2 + LV^2T\right],$$

where $\Delta_0 = f_{\max} - f^*$ and $\hat{x}_T$ is randomly drawn from $\{x_t\}_{t=1}^T$ with probability $P_t = \frac{1/\eta_t}{\sum_{t=1}^T 1/\eta_t}$. In particular, $\beta = \sqrt{T}$ yields

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \le \left(\frac{\Delta_0}{\eta_0} + \frac{LV^2\eta_0}{2}\right) \cdot \frac{\ln T}{\sqrt{T} - 1}. \tag{6}$$

Theorem 3.3 establishes the convergence of Exp-Decay to a stationary point. With $\beta = \sqrt{T}$, Exp-Decay yields the rate of $\mathcal{O}(\ln T/\sqrt{T})$, which is comparable to the results in [29]. However, in [29], the $\mathcal{O}(\ln T/\sqrt{T})$-rate is obtained under the (arguably impractical) assumption that the initial step-size η_0 is bounded by $\mathcal{O}(1/\sqrt{T})$. This means that the initial step-size η_0 will be small if we plan on running the algorithm for a large number of iterations T. This is in conflict to the original motivation for using exponentially decaying step-sizes, namely to allow for large initial step-sizes which decrease as the algorithm progresses. On the other hand, our results in Theorem 3.3 only require that the initial step-size is bounded by $\eta_0 \leq 1/L$, which matches the largest fixed step-sizes that ensure convergence.

Another advantage of Theorem 3.3 is that it uses the output probability $P_t \propto 1/\eta_t$, instead of $P_t \propto \eta_t$ as in [29]. As discussed in the beginning of the section (and in Figure 1), $P_t \propto 1/\eta_t$ means that the output is much more likely to be chosen during the final iterates, whereas with $P_t \propto \eta_t$ the output is more likely to come from the initial iterates. Therefore, Theorem 3.3 better reflects the actual convergence of the algorithm, since in practice it is typically the final iterate that is used as the trained model. We illustrate this better in our experiments in Section 5.

3.3 Improved Convergence for the $1/\sqrt{t}$ Step-Size

The next theorem demonstrates how the idea of using the output distribution $P_t \propto 1/\eta_t$ instead of $P_t \propto \eta_t$ can improve standard convergence bounds for $1/\sqrt{t}$ step-sizes.

Theorem 3.4. *Suppose that the objective function is L-smooth and upper bounded by $f_{\max}$, and that the stochastic gradient oracle is variance-bounded by V^2. If $\eta_t = \eta_0/\sqrt{t} \leq 1/L$, then*

$$\mathbb{E}[\|\nabla f(\hat{x}_T)\|^2] \leq \left(\frac{3\Delta_0}{\eta_0} + \frac{3LV^2\eta_0}{2}\right) \cdot \frac{1}{\sqrt{T}}$$

where $\Delta_0 = f_{\max} - f^*$ and $\hat{x}_T$ is randomly drawn from $\{x_t\}_{t=1}^T$ with probabilities $P_t = \frac{1/\eta_t}{\sum_{t=1}^T 1/\eta_t}$.

Theorem 3.4 establishes an $\mathcal{O}(1/\sqrt{T})$ convergence rate of SGD with $1/\sqrt{t}$ step-sizes. This improves the best known convergence bounds $\mathcal{O}(\ln T/\sqrt{T})$ for the algorithm, see Proposition 3.1 or [13].

4 Convergence for Strongly Convex Problems

We now investigate the convergence of the step decay step-sizes in the strongly convex case. We allow general convex and closed constraint set $X \subset \mathbb{R}^d$ and possibly non-smooth convex problems. To capture this setting, Algorithm 1 needs to be adjusted by including a projection step and subgradients (see Algorithm 2 in supplementary material). We also give the convergence for general convex case (see Appendix A).

4.1 Strongly Convex and L-Smooth Functions

We first consider the case when f is strongly convex and L-smooth (with respect to x^*).

Theorem 4.1. *Assume that the objective function f is μ-strongly convex on $\mathcal{X}$ and the stochastic gradient oracle is bounded by G^2. If we consider Algorithm 2 with $T > 1$, $N = \log_\alpha T$, $x_1^1 \in \mathcal{X}$, and $\eta_0 < 1/(2\mu)$, then we have*

$$\mathbb{E}\left\|x_{S+1}^N - x^*\right\|^2 \leq \frac{R}{\exp\left(A_3 \frac{T-1}{\ln T}\right)} + \frac{\alpha\eta_0 G^2 \exp\left(\frac{A_3}{\ln T}\right)}{2\mu A_3} \frac{\ln T}{T}$$

where $A_3 = 2\mu\eta_0\alpha \ln\alpha/(\alpha - 1)$ and $R = \left\|x_1^1 - x^*\right\|^2$. Further, if the objective function f is L-smooth with respect to x^* then we can bound the objective function as follows

$$\mathbb{E}[f(x_{S+1}^N) - f(x^*)] \leq \frac{L}{2}\mathbb{E}\left\|x_{S+1}^N - x^*\right\|^2.$$

The theorem provides an $\mathcal{O}(\ln T/T)$ theoretical guarantee for the last iterate under step decay step-size.[3] This bound matches the convergence rate of [12] for strongly convex least squares problems.

[3] Note that $\exp(A_3/\ln T) \leq \exp(A_3/\ln 2)$ for all $T \geq 2$ and $\exp(A_3/\ln T)$ converges to 1 as T goes to ∞.

Therefore, Theorem 4.1 can be considered a generalization of the result in [12] to general smooth and strongly convex problems.

From Theorem 4.1, we see that the effect of the initial condition $\|x_1^1 - x^\star\|^2$ is forgotten exponentially fast, unlike for $1/t$ step-sizes (e.g.[32]). For $1/t$ step-sizes, the initial learning rate is critical (both in theory and practice) to the convergence rate of SGD [33, 32]. For example, an initial step-size of $\eta_0 < 2/\mu$, will lead to a $t^{-\mu\eta_0/2}$ rate, far from the optimal rate $1/t$. For step decay, the initial step-size only affects the constant in front of the theoretical convergence bound and is not sensitive to the choice of initial step-size $\eta_0 < 1/(2\mu)$. We also observe this property of step-decay step-size in the experiments in Figure 4(b).

Under some step-sizes, e.g. $\eta_t = 1/t$, it is possible to get an $\mathcal{O}(1/T)$ convergence rate for SGD for smooth and strongly convex problems [32, 37, 20]. However, our next result shows that the rate in Theorem 4.1 is tight for Algorithm 2.

Theorem 4.2 (Lower Bound)**.** *Consider Algorithm 2 with $S = T/\log_\alpha T$ and $x_1^1 = 0$. For any $T \in \mathbb{N}^+$ and $\delta \in (0,1)$, there exists a function $\tilde{f}_T : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = [-4,4]$, that is both 1-strongly convex and 1-smooth such that*

$$\tilde{f}_T(x_{S+1}^N) - \tilde{f}_T(x^*) \geq \frac{\ln(1/\delta)}{9\exp(2)\ln\alpha} \cdot \frac{\ln T}{T}$$

with probability at least δ, where $x^* = \min_{x\in\mathcal{X}} \tilde{f}_T(x)$.

The high probability lower bound $\mathcal{O}(\ln T/T)$ matches the upper bound in Theorem 4.1. This suggests that the bound in Theorem 4.1 is actually tight for Algorithm 2 in the smooth strongly-convex case.

4.2 Strongly Convex and Non-Smooth Functions

For general, not necessarily smooth, strongly convex functions we have the following result.

Theorem 4.3. *Suppose that the objective function f is μ-strongly convex on $\mathcal{X}$ and the stochastic gradient oracle is bounded by G^2. If we run Algorithm 2 with $T > 1$, $S = T/\log_\alpha T$, $x_1^1 \in \mathcal{X}$, and $\eta_0 < 1/(2\mu)$, then we have*

$$\mathbb{E}[f(x_S^N) - f(x^*)] \leq \frac{R\ln T}{A_4\exp\left(B_4\frac{T-\alpha}{\ln T}\right)} + C_4\frac{\ln T + 2}{T} + D_4\exp\left(\frac{E_4}{\ln T}\right)\frac{\ln^2 T}{T}$$

where $A_4 = \eta_0\alpha\ln\alpha$, $B_4 = 2\mu A_4/(\alpha-1)$, $C_4 = G^2\eta_0\alpha$, $D_4 = G^2/(2\mu\ln\alpha B_4)$, $E_4 = \alpha B_4$.

The theorem shows that even without the smoothness assumption, we can still ensure convergence at the rate $\mathcal{O}(\ln^2 T/T)$. We can improve the rate to $\mathcal{O}(\ln T/T)$ with an averaging technique, as illustrated next. Moreover, the averaging technique improves the dependence of the parameter μ from $1/\mu^2$ (see Theorem 4.3) to $1/\mu$.

Theorem 4.4. *Under the assumptions of Theorem 4.3*

$$\mathbb{E}[f(\hat{x}_T) - f(x^*)] \leq \frac{A_5 R}{\exp\left(B_4\frac{T}{\ln T} - 1\right)} + C_5\frac{\ln T}{T}$$

where $\hat{x}_T = \sum_{t=t^*}^N \eta_t \sum_{i=1}^S x_i^t/(S\sum_{t=t^*}^N \eta_t)$, $t^* := \max\{0, \lfloor\log_\alpha(\eta_0\alpha A_5 T/\log_\alpha T)\rfloor\}$, $A_5 = 2\mu\alpha/(\alpha-1)$, $C_5 = \alpha(2 + 1/(\alpha^2-1))G^2/(2\mu\ln\alpha)$, and B_4 is as defined in Theorem 4.3.

5 Numerical Experiments

In this section, we evaluate the practical performance of step decay step-size and compare it against the following popular step-size policies: 1) constant step-size, $\eta_t = \eta_0$; 2) $1/t$ step-size, $\eta_t = \eta_0/(1 + a_0 t)$; 3) $1/\sqrt{t}$ step-size $\eta_t = \eta_0/(1 + a_0\sqrt{t})$; 4) Exp-Decay [29], $\eta_t = \eta_0/\alpha^t$ with $\alpha = (\beta/T)^{-1/T}$ for $\beta \geq 1$. In each experiment, we perform a grid search to select the best values for the free parameters η_0, a_0, β as well as for the step decay step-size parameter α. More details about the relationship between the different step-size policies are given in the supplementary material.

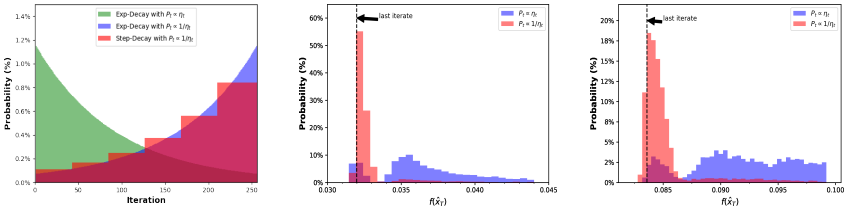


Figure 1: Output probability distributions (left); training loss (middle) and testing loss (right)

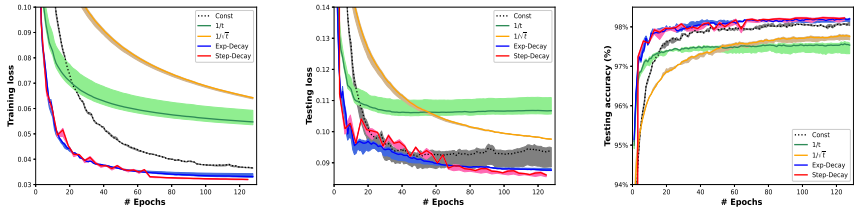


Figure 2: Results on MNIST: left - training loss; middle - testing loss; right - testing accuracy

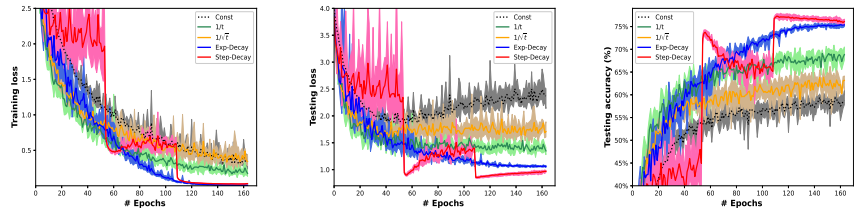


Figure 3: Results on CIFAR100: left - training loss; middle - testing loss; right - testing accuracy

5.1 Experiments on MNIST with Neural Networks

Consider the classification task on MNIST database of handwritten digits using a fully-connected 2-layer neural network with 100 hidden nodes (784-100-10) [3]. We first explore the output x_T at the last iterate T, which is the output of the algorithm that is typically used in practice. The theoretical output $\hat{x}_T$ (in Theorem 3.2) is drawn from all the previous iterates $\{x_i^t\}$ with probability $P_t \propto 1/\eta_t$. To get an insight into the relationship between the last iterate and theoretical output in Theorem 3.2, we randomly choose 6000 iterates (10% of the total iterates) with probability $P_t \propto 1/\eta_t$, record their exact training loss and testing loss, and calculate the probabilities (shown in Figure 1, middle and right). We can see that the theoretical output can reach the results of the last iterate with high probability, no matter training loss or testing loss.

In the same way, in order to show the advantages of probability $P_t \propto 1/\eta_t$ over $P_t \propto \eta_t$, we also implement Step-Decay with probability $P_t \propto \eta_t$ in Figure 1. We can see that the output with probability $P_t \propto 1/\eta_t$ is more concentrated at the last phase and has a higher probability, especially, in terms of loss, compared to the result of probability $P_t \propto \eta_t$.

The performance of the various step-size schedules is shown in Figure 2. Exp-Decay and Step-Decay perform better than other step-sizes both in loss (training and testing) and testing accuracy. Step-Decay has an advantage over Exp-Decay in the later stages of training where it attains a lower training and testing loss.

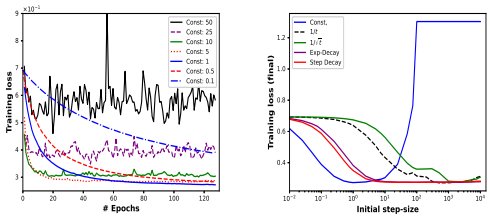
5.2 Experiments on CIFAR10 and CIFAR100

To illustrate the practical implications of the step decay step-size, we perform experiments with deep learning tasks on the CIFAR [1]. We will focus on results for CIFAR100 here, and present complementary results for CIFAR10 in the supplementary material. To eliminate the influence of stochasticity, all experiments are repeated 5 times.

We consider the benchmark experiments for CIFAR100 on a 100-layer DenseNet [23]. We employ vanilla SGD without dampening and use a weight decay of 0.0005. The optimal step-size and algorithm parameters are selected using a grid search detailed in the supplementary material. The results are shown in Figure 3. We observe that Step-Decay achieves the best results in terms of both testing loss and testing accuracy, and that it is also fast in reaching a competitive solution. Another observation is that as the iterates proceeds in each phase, its testing loss and accuracy is getting worse because its generalization ability is weakened. Therefore, deciding when to stop the iteration or reduce the step-size is important.

Table 1: Testing accuracy on CIFAR100.

Method	CIFAR100-DenseNet Testing accuracy
AdaGrad	0.6197 ± 0.00518
Adam + Exp-Decay	$0.6936 \pm$ **0.00483**
Adam + Step-Decay	**0.7041** ± 0.00971
AdamW + Exp-Decay	0.7165 ± 0.00353
AdamW + Step-Decay	**0.7335** $\pm$ **0.00261**
NAG + Exp-Decay	0.7531 ± 0.00606
NAG + Step-Decay	**0.7568** $\pm$ **0.00156**



(a) Constant step-size (b) Initial step-sizes

Figure 4: The results on rcv1.binary - logistic(L_2)

Finally, we compare the performance of Exp-Decay and Step-Decay on Nesterov's accelerated gradient (NAG) [35, 41] and other adaptive gradient methods, including AdaGrad [11], Adam [24] and AdamW [31]. The results are shown in Table 1. All the parameters involved in step-sizes, algorithms, and models are best-tuned (shown in supplementary material). The $\pm$ shows 95% confidence intervals of the mean accuracy value over 5 runs. We can see that compared to the Exp-Decay step-size, Step-Decay can reach higher testing accuracy on Adam, AdamW and NAG. This suggest that Step-Decay step-sizes can be extended to other methods.

5.3 Experiments on Regularized Logistic Regression

We now turn our attention to how the step-decay step-size and other related step-sizes behave in the strongly convex setting. We consider the regularized logistic regression problem on the binary classification dataset rcv1.binary ($n = 20242; d = 47236$) from LIBSVM [2], where a 0.75 partition is used for training and the rest is for testing.

Figure 4(a) shows how a constant step-size has to be well-tuned to give good performance: if we choose it too small, convergence will be painstakingly slow; if we set it too large, iterates will stagnate or even diverge. This effect is also visualized in blue in Figure 4(b), where only a narrow range of values results in a low training loss for the constant step-size. In contrast, the initial step-size of both Exp-Decay and Step-Decay can be selected from a wide range ($\approx 10 - 10^4$) and still yield good results in the end. In other words, Exp-Decay and Step-Decay are more robust to the choice of initial step-size than the alternatives. A more thorough evaluation of all the considered step-sizes on logistic regression can be found in Supplementary D.3.

6 Conclusion

We have provided theoretical guarantees for SGD under the step decay family of step-sizes, widely used in deep learning. Our first results established a near-optimal $\mathcal{O}(\ln T/\sqrt{T})$ rate for step decay step-sizes in the non-convex setting. A key step in our analysis was to use a novel non-uniform probability distribution $P_t \propto 1/\eta_t$ for selecting the output of the algorithm. We showed that this approach allows to improve the convergence results for SGD under other step-sizes as well, e.g., by removing the $\ln T$ term in the best known convergence rate for $\eta_t = 1/\sqrt{t}$ step-sizes. Moreover, we established near-optimal (compared to the min-max rate) convergence rates for general convex, strongly convex and smooth, and strongly convex and non-smooth problems. We illustrated the superior performance of step-decay step-sizes for training of large-scale deep neural networks. In the experiments, we observed that as the iterates proceeding in each phase, their generalization abilities are getting worse. Therefore, it will be an interesting to study how to best select the size of inner-loop S (instead of constant or exponentially growing) to avoid the loss of generalization.

References

[1] The CIFAR data set. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2021-01-05.

[2] The LIBSVM data set. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Accessed: 2021-01-05.

[3] The MNIST data set. http://yann.lecun.com/exdb/mnist/. Accessed: 2021-01-05.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *Software available from tensorflow.org*, 2015.

[5] Christian Brownlees, Emilien Joly, Gábor Lugosi, et al. Empirical risk minimization for heavy-tailed losses. *Annals of Statistics*, 43(6):2507–2536, 2015.

[6] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Gábor Lugosi. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11):7711–7717, 2013.

[7] Coralia Cartis, Nicholas IM Gould, and Ph L Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

[8] Zaiyi Chen, Zhuoning Yuan, Jinfeng Yi, Bowen Zhou, Enhong Chen, and Tianbao Yang. Universal stagewise learning for non-convex problems with convergence on averaged solutions. In *International Conference on Learning Representations*, 2019.

[9] Damek Davis, Dmitriy Drusvyatskiy, and Vasileios Charisopoulos. Stochastic algorithms with geometric step decay converge linearly on sharp functions. *arXiv:1907.09547*, 2019.

[10] Yoel Drori and Ohad Shamir. The complexity of finding stationary points with stochastic gradient descent. In *International Conference on Machine Learning*, pages 2658–2667. PMLR, 2020.

[11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[12] Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, pages 14977–14988, 2019.

[13] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[14] Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

[15] Jean-Louis Goffin. On convergence rates of subgradient optimization methods. *Mathematical Programming*, 13(1):329–347, 1977.

[16] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

[17] Zhishuai Guo, Zixuan Wu, Yan Yan, Xiaoyu Wang, and Tianbao Yang. Revisiting sgd with increasingly weighted averaging: Optimization and generalization perspectives. *arXiv preprint arXiv:2003.04339*, 2020.

[18] Nicholas JA Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. Tight analyses for non-smooth stochastic gradient descent. In *Conference on Learning Theory*, pages 1579–1613. PMLR, 2019.

[19] Nicholas JA Harvey, Christopher Liaw, and Sikander Randhawa. Simple and optimal high-probability bounds for strongly-convex stochastic gradient descent. *arXiv:1909.00843*, 2019.

[20] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Daniel Hsu and Sivan Sabato. Loss minimization and parameter estimation with heavy tails. *Journal of Machine Learning Research*, 17(18):1–40, 2016.

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[24] Diederik P Kingma and Jimmy Lei Ba. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[25] Philip Klein and Neal E Young. On the number of iterations for dantzig–wolfe optimization and packing-covering approximation algorithms. *SIAM Journal on Computing*, 44(4):1154–1172, 2015.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

[27] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $\mathcal{O}(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv:1212.2002*, 2012.

[28] Hunter Lang, Lin Xiao, and Pengchuan Zhang. Using statistics to automate stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 9540–9550, 2019.

[29] Xiaoyu Li, Zhenxun Zhuang, and Francesco Orabona. A second look at exponential and cosine step sizes: Simplicity, convergence, and performance. *arXiv:2002.05273*, 2020.

[30] Nicolas Loizou, Sharan Vaswani, Issam Laradji, and Simon Lacoste-Julien. Stochastic Polyak step-size for SGD: A adaptive learning rate for fast convergence. *arXiv:2002.10542*, 2020.

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[32] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[33] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[34] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2004.

[35] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, Long Beach, CA, US, 2017.

[37] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *International Conference on Machine Learning*, pages 1571–1578, 2012.

[38] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[39] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[40] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[41] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

[42] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *Technical Report, University of Toronto*, 2012.

[43] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, 2019.

[44] Yi Xu, Qihang Lin, and Tianbao Yang. Stochastic convex optimization: Faster local growth implies faster global convergence. In *International Conference on Machine Learning*, pages 3821–3830. PMLR, 2017.

[45] Yi Xu, Qi Qi, Qihang Lin, Rong Jin, and Tianbao Yang. Stochastic optimization for DC functions and non-smooth non-convex regularizers with non-asymptotic convergence. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6942–6951. PMLR, 09–15 Jun 2019.

[46] Yi Xu, Shenghuo Zhu, Sen Yang, Chi Zhang, Rong Jin, and Tianbao Yang. Learning with non-convex truncated losses by sgd. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 701–711. PMLR, 22–25 Jul 2020.

[47] Zhuoning Yuan, Yan Yan, Rong Jin, and Tianbao Yang. Stagewise training accelerates convergence of testing error over SGD. *Advances in Neural Information Processing Systems*, 32:2608–2618, 2019.

[48] Pengchuan Zhang, Hunter Lang, Qiang Liu, and Lin Xiao. Statistical adaptive stochastic gradient methods. *arXiv:2002.10597*, 2020.

Checklist

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] See the conclusion part in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is mostly a theoretical work and, therefore, it does not have any direct negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We do.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2 and the theorems in Section 3 and 4
   (b) Did you include complete proofs of all theoretical results? [Yes] We do. Please see Appendix A, B, C.
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] We do. See the references [1, 2, 3] and also Appendix D.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We do, see Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We do, see Section 5 and Appendix D.
   (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] We do, see Appendix D.
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? [Yes] We do.
   (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? [No] We do not include any new assets.
   (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]