Prodigy: An Expeditiously Adaptive Parameter-Free Learner

Konstantin Mishchenko [1] **Aaron Defazio** [2]

Abstract

We consider the problem of estimating the learning rate in adaptive methods, such as AdaGrad and Adam. We propose Prodigy, an algorithm that provably estimates the distance to the solution D, which is needed to set the learning rate optimally. At its core, Prodigy is a modification of the D-Adaptation method for learning-rate-free learning. It improves upon the convergence rate of D-Adaptation by a factor of $\mathcal{O}(\sqrt{\log(D/d_0)})$, where d_0 is the initial estimate of D. We test Prodigy on 12 common logistic-regression benchmark datasets, VGG11 and ResNet-50 training on CIFAR10, ViT training on Imagenet, LSTM training on IWSLT14, DLRM training on Criteo dataset, VarNet on Knee MRI dataset, as well as RoBERTa and GPT transformer training on BookWiki. Our experimental results show that our approach consistently outperforms D-Adaptation and reaches test accuracy values close to that of hand-tuned Adam.

1. Introduction

Optimization is an essential tool in modern machine learning, enabling efficient solutions to large-scale problems that arise in various domains, such as computer vision, natural language processing, and reinforcement learning. One of the key challenges is the selection of appropriate learning rates, which can significantly impact the convergence speed and the quality of the final solution. Learning-rate tuning has been particularly challenging in applications where there are multiple agents that use their own optimizer. For instance, when training Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), there are two neural networks with different architectures. In federated learning, tuning is even more challenging (Khodak et al., 2021), since there might be billions of devices (Kairouz et al., 2021), each optimizing their objective locally. Another example is Neural Architecture Search (NAS) (Zoph & Le, 2017), where the goal is to find the best neural network architecture automatically by training a lot of networks and evaluating them on a validation set. In such cases it becomes very expensive to manually tune the learning rate.

Recently, *parameter-free* adaptive learning rate methods (Orabona & Tommasi, 2017; Cutkosky & Orabona, 2018; Zhang et al., 2022; Carmon & Hinder, 2022; Ivgi et al., 2023) have gained considerable attention due to their ability to automatically adjust learning rates based on the problem structure and data characteristics. Among these, the D-Adaptation method, introduced by Defazio & Mishchenko (2023), has emerged as a promising practical approach for learning-rate-free optimization.

For a convex objective function f, D-Adaptation works by maintaining a lower bound on the initial distance to solution $D = \|x_0 - x_*\|$, for any x_* in the solution set of the following problem:

$$\min_{x \in \mathbb{R}^p} f(x).$$

In practice, the lower bound estimated by D-Adaptation increases rapidly during the course of optimization, plateauing to a value close to the true D. This D quantity is the key unknown constant needed to set the learning rate for adaptive optimization methods, forming the numerator of the step size:

$$\eta_k = \frac{D}{\sqrt{\sum_{i=0}^{k} \|g_i\|^2}}, \qquad \text{where } D = \|x_0 - x_*\|, \quad (1)$$

and the denominator is based on the AdaGrad step size (Duchi et al., 2011; Streeter & McMahan, 2010; Ward et al., 2019). The Gradient Descent form of D-Adaptation simply plugs in the current lower bound at each step in place of D. This simple approach can be applied to estimate the step size in Adam (Kingma & Ba, 2015), which yields state-of-the-art performance across a wide-range of deep learning problems. Defazio & Mishchenko (2023) also show that asymptotically, D-Adaptation is as fast as specifying the step size using the true D (up to small constant factors).

Contributions

We summarize our contributions as follows:

[1]Samsung AI Center [2]Fundamental AI Research Team, Meta. Correspondence to: Konstantin Mishchenko <konsta.mish@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

- We present *Prodigy*, a modification of D-Adaptation that improves its worst-case non-asymptotic convergence rate.
- Through extensive experiments, we demonstrate that Prodigy establishes a new state-of-the-art for learning rate adaptation, outperforming D-Adaptation.
- We develop a lower complexity bound for methods which grow the learning rate at most exponentially fast. We show that this covers all methods that avoid significant overshooting.
- Prodigy is highly practical. Our open-source implementation is already widely used for fine-tuning of vision and language models, and is the recommended optimizer for *Hugging Face Diffusers* DreamBooth LoRA training.

2. Prodigy Approach

To understand how we can improve upon D-Adaptation, let us take a closer look at some details of its analysis. Consider Gradient Descent update

$$x_{k+1} = x_k - \eta_k g_k,$$

where g_k is the sub-gradient used at iteration k. The standard analysis of AdaGrad with a constant numerator gives the following error term:

$$\text{AdaGrad error} = \sum_{k=0}^{n} \eta_k^2 \|g_k\|^2,$$

which is exactly why AdaGrad places $\sqrt{\sum_{i=0}^k \|g_i\|^2}$ in the step-size denominator in equation (1). However, when setting the numerator adaptively with an estimate d_k instead of the unknown D, as done in D-Adaptation, we end up with a different error term:

$$\text{D-adaptation error} = \sum_{k=0}^{n} d_k^2 \eta_k^2 \|g_k\|^2.$$

The theory of D-Adaptation handles this error term by upper bounding by using the upper bound $d_k \le d_n$ to get a similar error term to that of AdaGrad. This upper bound, however, is quite pessimistic since d_n can be as large as D and d_k can be as small as d_0. Therefore, replacing d_k^2 with d_n^2 can introduce a multiplicative error of $\frac{D^2}{d_0^2}$ in this term.

In this paper, we take a different approach and instead handle the error term using modified AdaGrad-like step sizes. Since the error terms are now $d_i^2\|g_i\|^2$ instead of $\|g_i\|^2$, the new adaptive step size should be

$$\eta_k = \frac{d_k^2}{\sqrt{\sum_{i=0}^k d_i^2 \|g_i\|^2}}.$$

Algorithm 1 Prodigy (GD version)

Input: $d_0 > 0$, x_0, $G \ge 0$
for $k = 0$ **to** n **do**
  $g_k \in \partial f(x_k)$
  Choose weight ω_k (default: $\omega_k = 1$)
  $\eta_k = \dfrac{d_k^2 \omega_k}{\sqrt{d_k^2 G^2 + \sum_{i=0}^k d_i^2 \omega_i^2 \|g_i\|^2}}$
  $x_{k+1} = x_k - \eta_k g_k$
  $\hat{d}_{k+1} = \dfrac{\sum_{i=0}^k \eta_i \langle g_i, x_0 - x_i \rangle}{\|x_{k+1} - x_0\|}$
  $d_{k+1} = \max(d_k, \hat{d}_{k+1})$
end for
Return $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2 \omega_k} \sum_{k=0}^n d_k^2 \omega_k x_k$

Algorithm 2 Prodigy (Dual Averaging version)

Input: $d_0 > 0$, x_0, $G \ge 0$; $s_0 = 0 \in \mathbb{R}^p$
for $k = 0$ **to** n **do**
  $g_k \in \partial f(x_k)$
  $s_{k+1} = s_k + d_k^2 g_k$
  $\hat{d}_{k+1} = \dfrac{\sum_{i=0}^k d_i^2 \langle g_i, x_0 - x_i \rangle}{\|s_{k+1}\|}$
  $d_{k+1} = \max(d_k, \hat{d}_{k+1})$
  $\gamma_{k+1} = \dfrac{1}{\sqrt{d_{k+1}^2 G^2 + \sum_{i=0}^k d_i^2 \|g_i\|^2}}$
  $x_{k+1} = x_0 - \gamma_{k+1} s_{k+1}$
end for
Return $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2} \sum_{k=0}^n d_k^2 x_k$

This way, we can still control the error term of D-Adaptation but the obtained step size is provably larger since d_k is non-decreasing:

$$\frac{d_k^2}{\sqrt{\sum_{i=0}^k d_i^2 \|g_i\|^2}} \ge \frac{d_k^2}{\sqrt{\sum_{i=0}^k d_k^2 \|g_i\|^2}} = \frac{d_k}{\sqrt{\sum_{i=0}^k \|g_i\|^2}}.$$

Having larger step sizes while preserving the main error term is the key reason why the new algorithms converge, as we show below, with a faster rate.

Notice, however, that the methods might still be slow because the denominator in the step size might grow too large over time. To remedy this, we introduce a modification for the step size by placing an extra weight ω_k next to the gradients:

$$\eta_k = \frac{d_k^2 \omega_k}{\sqrt{\sum_{i=0}^k d_i^2 \omega_i^2 \|g_i\|^2}}.$$

In fact, the modified step size might even increase between iterations, whereas the AdaGrad step size always decreases.

We will show that as long as ω_k does not grow too quickly, the worst-case convergence rate is almost the same.

To establish non-asymptotic theory, we also introduce in our algorithms an extra term G^2 in the denominator which upper bound the gradient norm, in case such a bound exists. We define it formally in the assumption below.

Assumption 1. *We assume that the objective f is G-Lipschitz, which implies that its gradients are bounded by G: for any $x \in \mathbb{R}^p$ and $g \in \partial f(x)$, it holds $\|g\| \le G$.*

Algorithm 1 and Algorithm 2 give Gradient Descent and the Dual Averaging variants of our new method. In contrast to AdaGrad, they estimate the *pro*duct of D and G in the denominator, so we call the proposed technique *Prodigy*. We give the following convergence result for Algorithm 1:

Theorem 1. *Assume f is convex and G-Lipschitz. Given any weights $1 \le \omega_0 \le \cdots \le \omega_n$, the functional gap of the average iterate of Algorithm 1 converges as*

$$f(\overline{x}_n) - f_* \le \sqrt{2\omega_n} DG \frac{d_{n+1}(2 + \log(1 + \sum_{k=0}^n \omega_k^2))}{\sqrt{\sum_{k=0}^n \omega_k d_k^2}}, \tag{2}$$

where $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2} \sum_{k=0}^n d_k^2 x_k$ is the weighted average iterate.

Notice that we have the freedom to choose any non-decreasing sequence ω_k as long as the right-hand side is decreasing. This allows us to put much more weight on the recent gradients and get more reasonable step sizes. For instance, we can choose $\omega_k = k^p$, where $p > 0$ and since $\sum_{k=0}^k k^{2p} = \mathcal{O}(k^{2p+2})$, it would result in an extra factor of $1 + p$ in the numerator due to the log term. The denominator, on the other hand, would increase as well, giving us a trade-off that depends on the values of d_k's. We note that weights $\omega_k = k$ have appeared previously in Accelegrad (Levy et al., 2018) and UniXGrad (Kavis et al., 2019), which combine AdaGrad step sizes with momentum, and $\omega_k = \sqrt{k}$ weighting is used in the recent MADGRAD method (Defazio & Jelassi, 2022).

To understand why this improves the convergence rate, consider the following lemma, which we prove in the appendix. The lemma presents an upper bound on the terms related to the d_k sequence in the right-hand side of equation 2.

Lemma 1. *Let $d_0 \le d_1 \le \cdots \le d_N$ be positive numbers and assume $N \ge 2\log_{2+}(\frac{d_N}{d_0})$, where $\log_{2+}(\cdot) = 1 + \log_2(\cdot)$. Then,*

$$\min_{t<N} \frac{d_{t+1}}{\sqrt{\sum_{k=0}^t d_k^2}} \le \frac{4\sqrt{\log_{2+}\left(\frac{d_N}{d_0}\right)}}{\sqrt{N}}.$$

In contrast to the bound in (Defazio & Mishchenko, 2023), we bound $\frac{d_{t+1}}{\sqrt{\sum_{k=0}^t d_k^2}}$ instead of $\frac{d_{t+1}}{\sum_{k=0}^t d_k}$. This is the reason why the overall guarantee improves by a factor of $\sqrt{\log_2(D/d_0)}$. For instance, if we set $\omega_k = 1$ for all k and substitute the bound from Lemma 1, we get the convergence rate

$$f(\overline{x}_t) - f_* = \mathcal{O}\left(\frac{GD\log(n)\sqrt{\log_{2+}(D/d_0)}}{\sqrt{n}}\right).$$

where $t \le n$ is chosen as the argmin from Lemma 1. Furthermore, for arbitrary increasing positive weights, we get the following guarantee by applying Lemma 1 directly to the bound in Theorem 1:

$$f(\overline{x}_t) - f_* = \mathcal{O}\left(\frac{GD\log(n)\sqrt{\log_{2+}(\frac{\omega_{n+1}D}{d_0})}}{\sqrt{n}} \log \sum_{k=0}^n \omega_k^2\right).$$

Even though our theory does not guarantee that it is beneficial to use increasing weights ω_k, this result is, to the best of our knowledge, new for AdaGrad-like methods. It allows for a wide range of choices in ω_k. For example, if we set $\omega_k = \beta_2^{-k^p}$ with $\beta_2 < 1$ and $p < 1/3$, then the method is still guaranteed to converge at the rate of $\mathcal{O}\left(\frac{1}{n^{(1-3p)/2}}\right)$. This is of particular interest when we study Adam-like methods, see Section 5 for a discussion.

The logarithmic term $\log(n)$ is, however, not necessary and only arises due to the use of Gradient Descent update. The Dual Averaging update of Algorithm 2, provides a tighter guarantee as given in the next theorem.

Theorem 2. *Let f be a convex and G-Lipschitz function. For Algorithm 2, it holds that:*

$$f(\overline{x}_t) - f_* \le \frac{4GD}{\sqrt{n}} \sqrt{\log_{2+}\left(\frac{D}{d_0}\right)},$$

where $t = \arg\min_{k \le n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^k d_i^2}}$ and $\log_{2+}(\cdot) = 1 + \log_2(\cdot)$.

Comparing this with the previous rate, the only difference is the removal of a multiplicative $\log(n)$ factor. This improvement, however, is mostly theoretical as Gradient Descent typically performs better in practice than Dual Averaging. We also note that we provide convergence results for Algorithm 2 without extra weights ω_k.

2.1. Smooth analysis

One appealing property of AdaGrad is that it is a *universal* method, i.e., it works for both non-smooth and smooth problems (Levy et al., 2018), and the same is also known for DoG (Ivgi et al., 2023). Here, we show that Prodigy can adapt to smoothness as well, and works if the gradients are not bounded by G.

Assumption 2. *We say f is L-smooth if it holds for any $x \in \mathbb{R}^p$ that*

$$\|\nabla f(x)\|^2 \leq 2L(f(x) - f_*). \tag{3}$$

We give the following result for Algorithm 1 when applied to smooth objectives.

Theorem 3. *Assume f is L-smooth, and set $G = 0$ and $\omega_0 = \cdots = \omega_n = 1$ in Algorithm 1. Let $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2} \sum_{k=0}^n d_k^2 x_k$, then we get the following convergence guarantee:*

$$f(\overline{x}_t) - f_* = \mathcal{O}\left(\frac{\log_{2+}\left(\frac{D}{d_0}\right) \log_{2+}^2\left(\frac{LD^2}{d_0\|g_0\|}\right)}{n}\right),$$

where $t = \arg\min_{k \leq n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^k d_i^2}}$.

Compared to the non-smooth $\mathcal{O}(1/\sqrt{n})$ convergence rate, the result in Theorem 3 offers a faster $\mathcal{O}(1/n)$ convergence rate. This also affects the first logarithmic term, which changes from $\sqrt{\log_{2+}(D/d_0)}$ to $\log_{2+}(D/d_0)$, reflecting that n should be proportional to $\log_{2+}(D/d_0)$ in both cases. There is, unfortunately, an extra $\log_{2+}^2\left(\frac{LD^2}{d_0\|g_0\|}\right)$ term in the bound, which we believe to be an artifact of our analysis.

3. Lower Complexity Bounds for Exponentially Bounded Algorithms

We can obtain an interesting class of algorithms, which contains our two D-Adaptation variants, by restricting the rate of growth.

Definition 1. *An optimization algorithm is exponentially bounded if there exists a constant d_0, so that for any sequence of G-bounded gradients it returns a sequence of iterates such that for all k:*

$$\|x_k - x_0\| \leq 2^k d_0.$$

Theorem 4. *D-Adaptation, DoG and Prodigy are exponentially bounded.*

A simple lower complexity bound can be established via a simple 1-dimensional resisting oracle. The bound depends on the "scale" of the initial step of the algorithm, which is the size of the initial step from x_0 to x_1. This initial step is $g_0 \cdot d_0/\sqrt{G^2 + \|g_0\|^2}$ for D-Adaptation, and can be thought of as an algorithm-agnostic measure of d_0.

Our lower bound allows the resisting oracle to choose a constant D after seeing x_1, which is a much stronger oracle than required for establishing a lower bound. Ideally, a lower bound could be established where the constant D is fixed but unknown to the algorithm, and the actual distance to solution $\|x_0 - x_*\| \leq D$ given by the oracle is allowed to depend on the iterate sequence.

The primary consequence of this difference is that our construction only tells us that hard problems exist for n small relative to D/d_0, of the scale $n < \log(D/d_0)$. It remains an open problem to show a more general lower bound for larger n. This is in a sense a trivial consequence of the exponentially bounded property, but is actually representative of the real behavior of the methods during the early steps of the algorithm, where both n and d_k actually are small. Any more general lower bound must cover this case.

Our new D-Adaptation variants are optimal among exponentially bounded algorithms for this complexity class:

Theorem 5. *Consider any exponentially bounded algorithm for minimizing a convex G-Lipschitz function starting from x_0, which has no knowledge of problem constants G and D. There exists a fixed gradient oracle such that any sequence of $x_{1,\ldots}, x_n$, there exists a convex Lipschitz problem f with $G = 1$ and $\|x_0 - x_*\| \leq D$ for all minimizing points x_*, consistent with the gradient oracle such that:*

$$\min_{k \leq n} f(x_k) - f_* \geq \frac{DG\sqrt{\log_2(D/x_1)}}{2\sqrt{n+1}}.$$

Using the simple construction from Theorem 5, we show in Appendix B that the class of exponentially bounded methods (potentially with an exponent other than 2) covers all Gradient Descent approaches that use an estimate of $d_k \leq cD$ for some constant c, and use a step size $\gamma_k \leq d_k/G$ without line-search or other additional queries. So the only way to achieve a $\log\log$ dependence on d_0 is by using a method that performs some queries that overshoot the standard D/G step size by more than a fixed constant factor. Although using larger step sizes is not problematic for Lipschitz functions, it comes with the risk of causing training divergence when applied to functions whose gradients are only locally bounded by G, which is common in machine learning settings.

Lower complexity bounds for the average regret in the more general online learning setting also apply here. They are of the form (Zhang et al., 2022):

$$\frac{1}{n}\sum_{k=0}^{n} \langle g_k, x_k - x_* \rangle = \Omega\left(\frac{DG\sqrt{\log_2(D/\epsilon)} + \epsilon}{\sqrt{n}}\right).$$

where ϵ is a "user-specified constant" playing a similar role to x_1. Bounds on the average regret directly bound function value sub-optimality as

$$\begin{aligned} f(\bar{x}) - f_* &\leq \frac{1}{n+1}\sum_{k=0}^{n} [f(x_k) - f_*] \\ &\leq \frac{1}{n+1}\sum_{k=0}^{n} \langle g_k, x_k - x_* \rangle, \end{aligned}$$

where $\bar{x} = \frac{1}{n+1}\sum_{k=0}^{n} x_k$.

4. Related Work

In this section, we review the major classes of techniques for optimizing convex Lipschitz functions with some level of problem parameter independence.

The Polyak step size (Polyak, 1987) trades the knowledge of D for f_*, achieving optimal convergence rate without additional log factors. Stable convergence requires accurate f_* estimates. A restarting scheme converges within a multiplicative log factor of the optimal rate (Hazan & Kakade, 2019). There has been substantial recent research on modifications of the Polyak step size to make it better suited to machine learning tasks (Loizou et al., 2021; Gower et al., 2021; Orvieto et al., 2022) but as of yet they have not seen widespread adoption.

Normalized sub-gradient descent and AdaGrad have received a lot of attention due to their universality (Grimmer, 2019; Levy et al., 2018). AdaGrad's extensions with momentum have been studied by Levy et al. (2018) and Ene et al. (2021).

Coin-betting (Orabona & Tommasi, 2017; McMahan & Orabona, 2014; Cutkosky & Orabona, 2018; Zhang et al., 2022; Orabona & Pál, 2021) is a family of approaches from the online learning setting which are also applicable for convex non-smooth optimization. They work by establishing a relationship by duality between regret minimization and wealth-maximization. Existing approaches for wealth-maximization can then be mapped to algorithms for regret minimization. Coin-betting approaches give convergence rates for an equal-weighted average of the iterates of the form:

$$f(\bar{x}_n) - f_* = \mathcal{O}\left(\frac{DG\sqrt{\log\left(1 + D/d_0\right)}}{\sqrt{n}}\right).$$

Standard D-Adaptation obtains asymptotic rates without the log factor, but was otherwise (theoretically) slower in finite time, as it had a $\log(\cdot)$ rather than a $\sqrt{\log(\cdot)}$ dependence on D/d_0:

$$f(\hat{x}_n) - f_* \le \frac{16DG\log_{2+}(D/d_0)}{\sqrt{n}}.$$

The Prodigy method closes the gap, giving the same sqrt-log dependence as coin betting.

The DoG method (Ivgi et al., 2023), based on the bisection approach of Carmon & Hinder (2022), is the only other approach that we are aware of that estimates D in an online fashion. DoG estimates D by $\bar{r}_k$:

$$\bar{r}_k = \max_{i \le k} \|x_i - x_0\|.$$

Algorithm 3 Prodigy (Adam version)

1: **Input:** $d_0 > 0$ (default 10^{-6}), x_0, β_1 (default 0.9), β_2 (default 0.999), ϵ (default 10^{-8}), γ_k (default 1 with cosine annealing)
2: $r_0 = 0$, $s_0 = 0$, $m_0 = 0$, $v_0 = 0$
3: **for** $k = 0$ **to** n **do**
4: $\quad g_k \in \partial f(x_k)$
5: $\quad m_{k+1} = \beta_1 m_k + (1-\beta_1) d_k g_k$
6: $\quad v_{k+1} = \beta_2 v_k + (1-\beta_2) d_k^2 g_k^2$
7: $\quad r_{k+1} = \sqrt{\beta_2} r_k + (1-\sqrt{\beta_2})\gamma_k d_k^2 \langle g_k, x_0 - x_k\rangle$
8: $\quad s_{k+1} = \sqrt{\beta_2} s_k + (1-\sqrt{\beta_2})\gamma_k d_k^2 g_k$
9: $\quad \hat{d}_{k+1} = \dfrac{r_{k+1}}{\|s_{k+1}\|_1}$
10: $\quad d_{k+1} = \max(d_k, \hat{d}_{k+1})$
11: $\quad x_{k+1} = x_k - \gamma_k d_k m_{k+1}/(\sqrt{v_{k+1}} + d_k\epsilon)$
12: **end for**

Ivgi et al. (2023) use this quantity as a plug-in estimate for the numerator of the step size, similar to D-Adaptation's approach. This approach can divergence in theory, but with additional modifications to the step size, the "tamed" T-DoG method is shown to converge. It has a $\log_+(D/d_0)$ dependence on the initial sub-optimally of the D estimate, so our approach improves on this dependence by a $\sqrt{\log_+(D/d_o)}$ factor.

Malitsky & Mishchenko (2020) proposed AdGD, a method for convex optimization that does not require any hyperparameters and has a rate that is at least as good as that of the optimally tuned Gradient Descent. However, AdGD requires the objective to be locally smooth, which hinders its use in many practical problems. Latafat et al. (2023) partially addressed this gap by proposing a proximal extension, but the case of non-smooth differentiable functions has remained unstudied.

5. Deriving Adam-Like Step Sizes

To derive an Adam-like method, which should use an exponential moving average for the step size, we want to approximate Adam's update of the exponential moving average of squared gradients:

$$v_{k+1} = \beta_2 v_k + (1-\beta_2) g_k^2 = (1-\beta_2)\sum_{i=0}^{k} \beta_2^{k-i} g_i^2,$$

where g_k^2 is the coordinate-wise square of the gradient g_k. We can achieve this using exponential weights, $\omega_k = \beta_2^{-k/2}$, which after substituting the definition of η_k give us the following identity:

$$\frac{d_k^4}{\eta_k^2} = \frac{d_k^2}{\omega_k^2} G^2 + d_k^2\|g_k\|^2 + \sum_{i=0}^{k-1} \beta_2^{k-i} d_i^2 \|g_i\|^2.$$

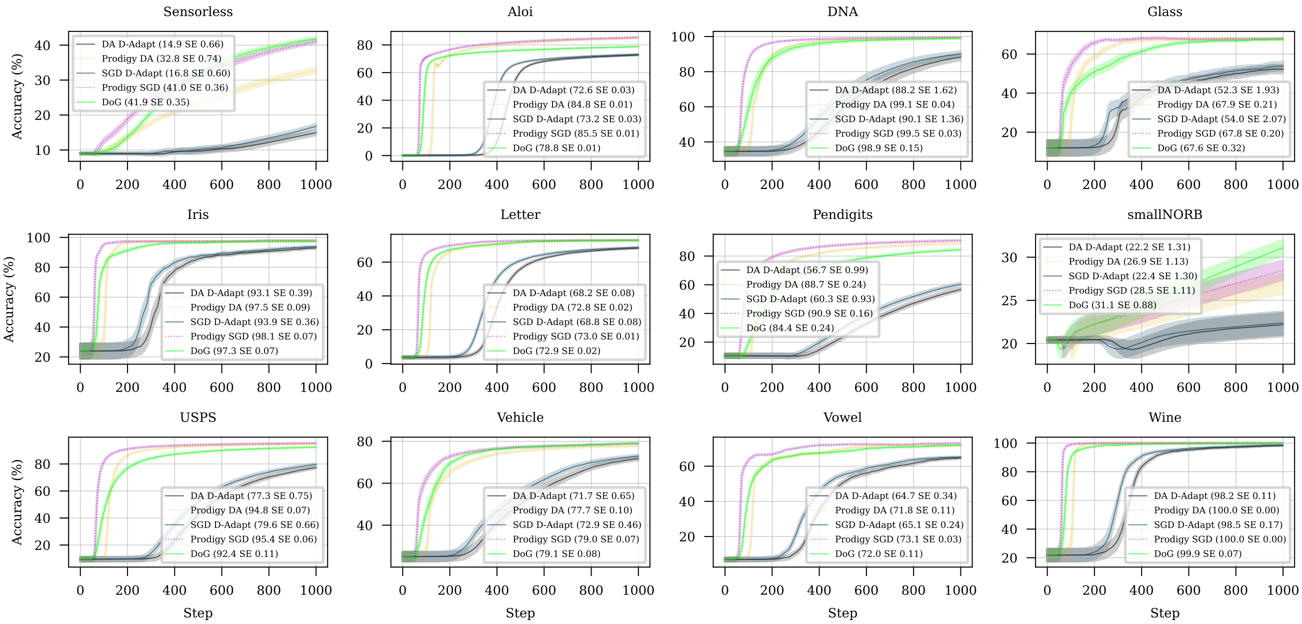

Figure 1. Convex multi-class classification problems. Error bars show a range of 1 standard error over 10 seeds.

This can be seen as computing an exponential moving average of $d_k g_k$ rather than g_k itself. In addition, in Appendix A.6, we provide a coordinate-wise version of Algorithm 2 and study its convergence properties. Based on the theory presented there, the denominator for $\hat{d}_{k+1}$ should use the ℓ_1 norm of the weighted gradient sum to estimate the ℓ_∞ distance to the solution $D_\infty = \|x_0 - x_*\|_\infty$. Thus, combining this insight with the design of Algorithm 1, we obtain the following expression for the Adam estimate of D_∞:

$$\hat{d}_{k+1} = \frac{\sum_{i=0}^{k} \beta_2^{(k-i)/2} d_i^2 \langle g_i, x_0 - x_i \rangle}{\|\sum_{i=0}^{k} \beta_2^{(k-i)/2} d_i^2 g_i\|_1}.$$

The update uses exponential moving average as well, although it is more conservative as it uses $\sqrt{\beta_2}$ instead of β_2. Note that there is an extra of $(1 - \beta_2)$ in the update for v_k, which can be optionally compensated for by using the bias correction discussed by Kingma & Ba (2015). These update rules are summarized in Algorithm 3. This is the main algorithm that we study numerically in the next section.

6. Experiments

We test our methods on convex logistic regression as well as deep learning problems. The Prodigy method is used as presented in Algorithm 3 in all deep learning experiments.

Logistic regression. For the convex setting, we ran a set of classification experiments. For each dataset, we used the multi-margin loss (Weston & Watkins, 1999), a multi-class generalization of the hinge loss. This non-smooth loss results in bounded gradients, which is required by our theory. We perform full-batch rather that stochastic optimization, for two reasons. Firstly, it matches the assumptions of our theory. Secondly, fast learning rate adaptation is more crucial for full-batch optimization than stochastic optimization as fewer total steps will be performed. Our convex experiments use the theoretical variants Algorithm 1 and Algorithm 2, but with $G = 0$ following standard practice.

We performed 1,000 steps for each dataset, using a randomized x_0 and plot the results of 10 seeds. We ran both DA and GD variants of each method. Each plot shows the accuracy of the average iterate for each method. Figure 1 shows that our proposed algorithm greatly out-performs regular D-Adaptation. Our weighted GD variant of D-Adaptation is faster consistently across each dataset. Additionally, it adapts faster than the DoG method (Ivgi et al., 2023) on 10 of the 12 problems.

CIFAR10. For neural network experiments[1], we consider training on CIFAR10 (Krizhevsky, 2009) with batch size 256, where D-Adapted Adam has a gap of a few percent compared to the standard Adam. We use cosine annealing with initial step size 1 for all Adam-based methods and initial step size 10^{-3} for Adam itself. The considered networks are VGG11 (Simonyan & Zisserman, 2014) and ResNet-50 (He et al., 2016)[2]. To simplify the experiment, we do not use weight decay, so both networks slightly overfit and do not reach high test accuracy values. All methods were run

[1]The PyTorch code of our optimizer is available at `https://github.com/konstmish/prodigy`

[2]VGG11 and ResNet-50 implementation along with the data loaders were taken from `https://github.com/kuangliu/pytorch-cifar`

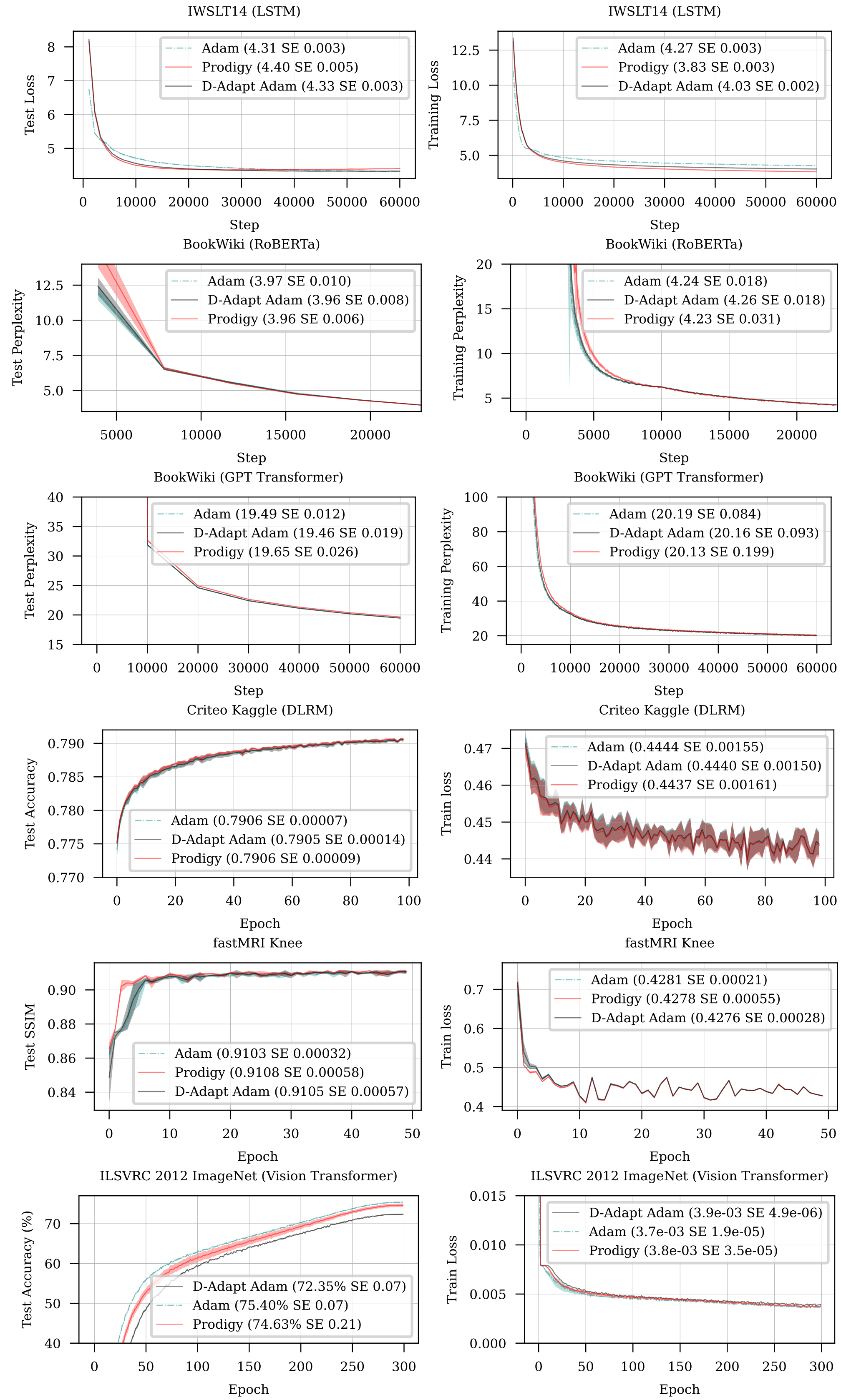

Figure 2. Adam-family experiments.

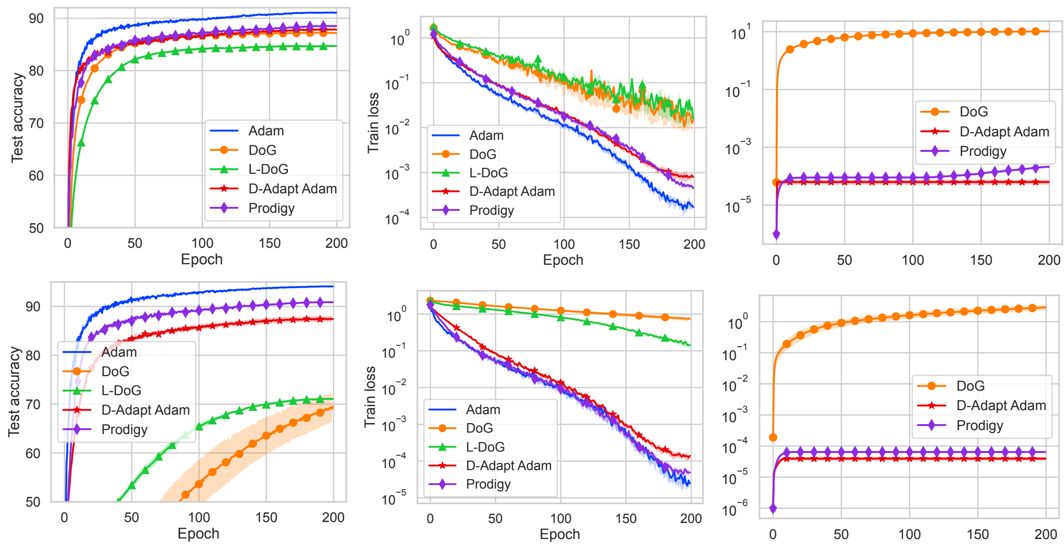

Figure 3. VGG11 (top) and ResNet-50 (bottom) training on CIFAR10. Left: test accuracy (%), middle: train loss, right: step sizes. "Prodigy" is used as given in Algorithm 3. Prodigy estimates a larger step size than D-Adaptation, which helps it reach test accuracy closer to Adam.

using same 8 random seeds.

We show the results in Figure 3. As we can see, this gap is closed by Prodigy, which is achieved by the larger estimates of the step size.

For DoG and L-DoG, we compute the polynomial-averaging iterate and then report the best of the average and the last iterate. We average with $\gamma = 8$, see (Ivgi et al., 2023) for the details. While DoG produces larger step size estimate than Prodigy (see the right column in Figure 3, this is counterbalanced by the larger denominator in DoG. We also tried to modify DoG to use Adam-like step sizes but our heuristic modification diverged on this problem. We also observed that among DoG and its layer-wise version, L-DoG, there is no clear winner as the former performed better on VGG11 and the latter was better when training ResNet-50.

nanoGPT transformer. We also train a 6-layer transformer network from nanoGPT[3] on the Shakespeare dataset. For all methods, we use batch size 256, clip the gradients to have norm not exceeding 1 and use float16 numbers. We use AdamW with hyperparameters given in the repository, i.e., $\beta_2 = 0.99$, weight decay 0.1, step size 10^{-3}, cosine annealing with warmup over 100 steps. The same weight decay value and cosine annealing is used for Prodigy and D-Adapted Adam, except that the latter two methods use step size 1. We accumulate minibatches of size 12 into a batch of size 480. We tuned the weight decay for DoG and L-DoG and found the value 10^{-4} to work well for this problem. We ran each method with 8 random seeds and report the average as well as one-standard-deviation confidence intervals.

See Figure 4 for the results. In terms of the test loss, all methods are roughly equivalent except that DoG and L-DoG were slower to reach the best value of roughly 1.5. For the train loss, Prodigy was on par with tuned AdamW and slightly better than D-Adapted Adam. Surprisingly, the estimated step size in Prodigy was very consistent across the 8 random seeds.

6.1. Large-scale Adam experiments

To validate the performance on large-scale practical applications directly against D-Adaptation, we ran the subset of the experiments from Defazio & Mishchenko (2023) that use the Adam optimizer. Methods without coordinate adaptivity are not competitive on these problems and so we exclude SGD and DoG from these comparisons.

LSTM, RoBERTa, GPT, DLRM, VarNet. On the smallest problem of LSTM training, Prodigy appears to converge significantly faster in training loss and slightly overfits in test loss compared to the baselines. For RoBERTa (Liu et al., 2019) and GPT (Radford et al., 2019) training on BookWiki, Prodigy matches the performance of the baseline with only negligible differences. For the application problems, DLRM (Naumov et al., 2019) on the Criteo Kaggle Display Advertising dataset, and fastMRI VarNet (Zbontar et al., 2018), Prodigy again closely matches the baselines.

[3] https://github.com/karpathy/nanoGPT

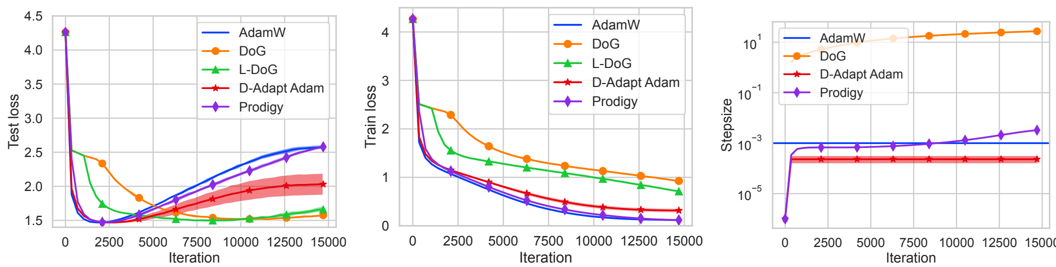

Figure 4. The test (left) and train (middle) loss curves as well as the estimated step size (right) when training a 6-layer nanoGPT transformer.

ViT training. Defazio & Mishchenko (2023) present a negative result for training vision transformer (Dosovitskiy et al., 2021), where D-Adaptation significantly underperforms tuned Adam. We investigated this effect, and we were able to reproduce this gap across a wide range of weight-decay values, although this problem has high run-to-run variance of 1-2% of test accuracy, which makes comparison difficult. Using weight decay 0.05 instead of 0.1 significantly improved performance of each variant, and so we present results for both the baselines and Prodigy at that value. We can see that Prodigy almost closes the gap between tuned Adam and D-Adaptation, giving a test accuracy of 74.63% compared to 75.4% for Adam, and more than 2% higher than D-Adaptation. See Figure 2 for the results.

7. Conclusion

We have presented Prodigy, a new method for learning-rate adaptation that improves upon the adaptation rate of the state-of-the-art D-Adaptation method. We proved convergence of Prodigy on both non-smooth and smooth problems. Prodigy was also shown to adapt faster than other known methods across a range of experiments. A practical limitation of our method is the increased memory requirement as it needs to store vectors x_0 and s_k in addition to Adam's vectors m_k and v_k. To remedy this issue, one can compress x_0 and s_k, for instance, by using lower precision data types, especially since we only use these vectors to produce scalar values for our d_k estimates.

Impact Statement

This paper presents work whose goal is to improve the training speed of existing Machine Learning models. We believe the societal consequences of our work are minimal.

References

Carmon, Y. and Hinder, O. Making SGD parameter-free. In *Conference on Learning Theory*. PMLR, 2022.

Cutkosky, A. and Orabona, F. Black-box reductions for parameter-free online learning in banach spaces. In *Proceedings of the 31st Conference On Learning Theory*, Proceedings of Machine Learning Research. PMLR, 2018.

Defazio, A. and Jelassi, S. Adaptivity without compromise: A momentumized, adaptive, dual averaged gradient method for stochastic optimization. *Journal of Machine Learning Research*, 23:1–34, 2022.

Defazio, A. and Mishchenko, K. Learning-rate-free learning by D-adaptation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7449–7479. PMLR, 23–29 Jul 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61), 2011.

Ene, A., Nguyen, H. L., and Vladu, A. Adaptive gradient methods for constrained convex optimization and variational inequalities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7314–7321, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gower, R. M., Defazio, A., and Rabbat, M. Stochastic Polyak stepsize with a moving target. *arXiv preprint arXiv:2106.11851*, 2021.

Grimmer, B. Convergence rates for deterministic and stochastic subgradient methods without Lipschitz continuity. *SIAM Journal on Optimization*, 29(2):1350–1365, 2019.

Hazan, E. and Kakade, S. M. Revisiting the Polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Ivgi, M., Hinder, O., and Carmon, Y. DoG is SGD's best friend: A parameter-free dynamic step size schedule. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14465–14499. PMLR, 23–29 Jul 2023.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1), 2021.

Kavis, A., Levy, K. Y., Bach, F., and Cevher, V. UniXGrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. *Advances in neural information processing systems*, 32, 2019.

Khodak, M., Tu, R., Li, T., Li, L., Balcan, M.-F. F., Smith, V., and Talwalkar, A. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *Advances in Neural Information Processing Systems*, 34: 19184–19197, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6980.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Latafat, P., Themelis, A., Stella, L., and Patrinos, P. Adaptive proximal algorithms for convex optimization under local Lipschitz continuity of the gradient. *arXiv preprint arXiv:2301.04431*, 2023.

Levy, K. Y., Yurtsever, A., and Cevher, V. Online adaptive methods, universality and acceleration. *Advances in neural information processing systems*, 31, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Loizou, N., Vaswani, S., Laradji, I., and Lacoste-Julien, S. Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research. PMLR, 2021.

Malitsky, Y. and Mishchenko, K. Adaptive gradient descent without descent. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6702–6712. PMLR, 13–18 Jul 2020.

McMahan, H. B. and Orabona, F. Unconstrained online linear learning in Hilbert spaces: Minimax algorithms and normal approximations. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*. PMLR, 2014.

Naumov, M., Mudigere, D., Shi, H. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C., Azzolini, A. G., Dzhulgakov, D., Mallevich, A., Cherniavskii, I., Lu, Y., Krishnamoorthi, R., Yu, A., Kondratenko, V., Pereira, S., Chen, X., Chen, W., Rao, V., Jia, B., Xiong, L., and Smelyanskiy, M. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, 2019.

Orabona, F. and Pál, D. Parameter-free stochastic optimization of variationally coherent functions, 2021.

Orabona, F. and Tommasi, T. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Orvieto, A., Lacoste-Julien, S., and Loizou, N. Dynamics of SGD with stochastic Polyak stepsizes: Truly adaptive variants and convergence to exact solution. In *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

Polyak, B. T. *Introduction to optimization*. Optimization Software, Inc., 1987.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. Technical report, OpenAI, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Streeter, M. and McMahan, H. B. Less regret via online conditioning. *arXiv preprint arXiv:1002.4862*, 2010.

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, 2019.

Weston, J. and Watkins, C. Support vector machines for multi-class pattern recognition. pp. 219–224, 01 1999.

Zbontar, J., Knoll, F., Sriram, A., Muckley, M. J., Bruno, M., Defazio, A., Parente, M., Geras, K. J., Katsnelson, J., Chandarana, H., et al. fastMRI: An open dataset and benchmarks for accelerated MRI. *arXiv preprint arXiv:1811.08839*, 2018.

Zhang, Z., Cutkosky, A., and Paschalidis, I. C. PDE-based optimal strategy for unconstrained online learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, 2022.

Zoph, B. and Le, Q. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=r1Ue8Hcxg`.

A. Analysis of Prodigy

As a reminder, we use the notation $\log_{2+}(a) = 1 + \log_2(a)$ to denote the logarithm that is lower bounded by 1 for any $a \geq 1$.

A.1. Useful propositions

Proposition 1 (Lemma A.2 in (Levy et al., 2018)). *For any sequence of nonnegative real numbers $a_0, \ldots, a_n$*

$$\sqrt{\sum_{k=0}^{n} a_i} \leq \sum_{k=0}^{n} \frac{a_k}{\sqrt{\sum_{i=0}^{k} a_i}} \leq 2\sqrt{\sum_{k=0}^{n} a_i}. \tag{4}$$

Proof. For completeness, we prove both statements here. Notice that for any $\alpha \in [0,1]$, it holds $1 - \sqrt{1-\alpha} \leq \alpha \leq 2(1 - \sqrt{1-\alpha})$. Substituting $\alpha = \frac{a_k}{\sum_{i=0}^{k} a_i}$ gives

$$1 - \sqrt{1 - \frac{a_k}{\sum_{i=0}^{k} a_i}} \leq \frac{a_k}{\sum_{i=0}^{k} a_i} \leq 2\left(1 - \sqrt{1 - \frac{a_k}{\sum_{i=0}^{k} a_i}}\right).$$

If we multiply all sides by $\sqrt{\sum_{i=0}^{k} a_i}$, the inequality above becomes

$$\sqrt{\sum_{i=0}^{k} a_i} - \sqrt{\sum_{i=0}^{k-1} a_i} \leq \frac{a_k}{\sqrt{\sum_{i=0}^{k} a_i}} \leq 2\left(\sqrt{\sum_{i=0}^{k} a_i} - \sqrt{\sum_{i=0}^{k-1} a_i}\right).$$

Summing over $k = 0, \ldots, n$, we get the stated bound. □

Proposition 2. *For any sequence of nonnegative numbers $a_0, \ldots, a_n$ and $A > 0$, it holds*

$$\sum_{k=0}^{n} \frac{a_k}{A + \sum_{i=0}^{k} a_i} \leq \log\left(A + \sum_{k=0}^{n} a_k\right) - \log(A). \tag{5}$$

Proof. If $a_i = 0$ for some i, we can simply ignore the corresponding summands, so let us assume that $a_i > 0$ for all i. For any $t > 0$ it holds $1/(1+t) \leq \log(1 + 1/t)$. Substituting $t = S_k/a_k$, where $S_k = A + \sum_{i=0}^{k-1} a_i$ for $k > 0$ and $S_0 = A$, we get

$$\frac{1}{1 + \frac{S_k}{a_k}} = \frac{a_k}{a_k + S_k} = \frac{a_k}{A + \sum_{i=0}^{k} a_i} \leq \log(1 + a_k/S_k) = \log(S_{k+1}) - \log(S_k).$$

Summing this over k from 0 to n, we get

$$\begin{aligned}\sum_{k=0}^{n} \frac{a_k}{A + \sum_{i=0}^{k} a_i} &\leq \sum_{k=0}^{n} (\log(S_{k+1}) - \log(S_k)) = \log(S_{n+1}) - \log(S_0) \\ &= \log\left(A + \sum_{k=0}^{n} a_k\right) - \log(A).\end{aligned}$$

This is exactly what we wanted to prove. □

A.2. Proof of Lemma 1

Proof. Following the proof in (Ivgi et al., 2023), we define $K = \left\lceil \log_2\left(\frac{d_N}{d_0}\right) \right\rceil$ and $n = \left\lfloor \frac{N}{K} \right\rfloor$. Consider a partitioning of the sequence $t \leq N$ into half-open intervals $I_k = [nk, n(k+1))$ for $k = 0$ to $K - 1$. We want to show that there is at least one interval such that d_k changes by at most a factor of 2 on that interval. We will use proof by contradiction.

Suppose that for all intervals, $d_{nk} < \frac{1}{2}d_{n(k+1)}$. Then d_k at least doubles in every interval, and so:

$$d_0 < \frac{1}{2}d_n < \frac{1}{4}d_{2n} \cdots < \frac{1}{2^K}d_{nK} < \frac{1}{2^K}d_N,$$

which implies that $d_N/d_0 > 2^K$ and so $K < \log_2(d_N/d_0)$ which contradicts our definition $K = \left\lceil \log_2\left(\frac{d_N}{d_0}\right)\right\rceil$. Therefore, there exists some $\hat{k}$ such that $d_{n\hat{k}} \geq \frac{1}{2}d_{n(\hat{k}+1)}$. We can now proceed with proving the Lemma by considering the summation over interval $I_{\hat{k}}$ only:

$$\begin{aligned}
\min_{t<N} \frac{d_{t+1}}{\sqrt{\sum_{k=0}^{t} d_k^2}} &\leq \frac{d_{n(\hat{k}+1)}}{\sqrt{\sum_{k=0}^{n(\hat{k}+1)-1} d_k^2}} \leq \frac{d_{n(\hat{k}+1)}}{\sqrt{\sum_{k=n\hat{k}}^{n(\hat{k}+1)-1} d_k^2}} \leq \frac{d_{n(\hat{k}+1)}}{\sqrt{\sum_{k=n\hat{k}}^{n(\hat{k}+1)-1} d_{n\hat{k}}^2}} \\
&= \frac{d_{n(\hat{k}+1)}}{\sqrt{n d_{n\hat{k}}^2}} \leq \frac{d_{n(\hat{k}+1)}}{\sqrt{\frac{1}{4} n d_{n(\hat{k}+1)}^2}} = \frac{2}{\sqrt{n}} = \frac{2}{\sqrt{\lfloor \frac{N}{K} \rfloor}} \\
&\leq \frac{2}{\sqrt{\frac{N}{K} - 1}} \leq \frac{2}{\sqrt{\frac{N}{\log_2(d_N/d_0)+1} - 1}} = \frac{2\sqrt{\log_{2+}\left(\frac{d_N}{d_0}\right)}}{\sqrt{N - \log_{2+}\left(\frac{d_N}{d_0}\right)}} \\
&\overset{N \geq 2\log_{2+}\left(\frac{d_N}{d_0}\right)}{\leq} \frac{4\sqrt{\log_{2+}\left(\frac{d_N}{d_0}\right)}}{\sqrt{N}}.
\end{aligned}$$

$\square$

A.3. GD analysis

Lemma 2. *Assume that $d_0 \leq D$. Then, the estimate d_k in Algorithm 1 satisfies $d_k \leq D$ for all k.*

Proof. By optimality of f_*, we have $f(x_k) - f_* \geq 0$, so

$$0 \leq \sum_{k=0}^{n} \eta_k (f(x_k) - f_*) \leq \sum_{k=0}^{n} \eta_k \langle g_k, x_k - x_* \rangle = \sum_{k=0}^{n} \eta_k \langle g_k, x_0 - x_* \rangle + \sum_{k=0}^{n} \eta_k \langle g_k, x_k - x_0 \rangle.$$

Collecting the gradients in the first sum together and using Cauchy-Schwarz inequality, we obtain

$$\begin{aligned}
0 \leq \sum_{k=0}^{n} \eta_k (f(x_k) - f_*) &\leq \langle x_0 - x_{n+1}, x_0 - x_* \rangle + \sum_{k=0}^{n} \eta_k \langle g_k, x_k - x_0 \rangle \\
&\leq \|x_0 - x_{n+1}\| \|x_0 - x_*\| + \sum_{k=0}^{n} \eta_k \langle g_k, x_k - x_0 \rangle. \qquad (6)
\end{aligned}$$

Using the definition of $\hat{d}_{n+1}$, this is equivalent to $0 \leq (D - \hat{d}_{n+1})\|x_0 - x_{n+1}\|$, which implies $\hat{d}_{n+1} \leq D$. Therefore, since $d_0 \leq D$, we can show by induction $d_{n+1} \leq D$ as well. $\square$

Lemma 3. *The following inequality holds for the iterates of Algorithm 1:*

$$\|x_{n+1} - x_0\| \leq 2d_{n+1} + \frac{1}{2d_{n+1}} \sum_{k=0}^{n} \eta_k^2 \|g_k\|^2.$$

Proof. Let us rewrite $\hat{d}_{n+1}$ in a slightly different manner:

$$\begin{aligned}
\hat{d}_{n+1}\|x_{n+1}-x_0\| &\overset{\text{def}}{=} \sum_{k=0}^{n}\langle x_k - x_{k+1}, x_0 - x_k\rangle \\
&= \sum_{k=0}^{n}\frac{1}{2}\left(\|x_{k+1}-x_0\|^2 - \|x_k - x_{k+1}\|^2 - \|x_k - x_0\|^2\right) \\
&= \frac{1}{2}\|x_{n+1}-x_0\|^2 - \frac{1}{2}\sum_{k=0}^{n}\|x_k - x_{k+1}\|^2.
\end{aligned}$$

Combining this with the property $\hat{d}_{n+1} \le d_{n+1}$, we derive

$$\frac{1}{2}\left\|x_{n+1}-x_0\right\|^2 - \frac{1}{2}\sum_{k=0}^{n}\left\|x_k - x_{k+1}\right\|^2 = \hat{d}_{n+1}\left\|x_{n+1}-x_0\right\| \le d_{n+1}\left\|x_{n+1}-x_0\right\|.$$

Applying inequality $2\alpha\beta \le \alpha^2+\beta^2$ with $\alpha^2 = 2d_{n+1}^2$ and $\beta^2 = \frac{1}{2}\|x_{n+1}-x_0\|^2$ and plugging-in the bound above, we establish

$$\begin{aligned}
2d_{n+1}\|x_{n+1}-x_0\| = 2\alpha\beta \le \alpha^2+\beta^2 &= 2d_{n+1}^2 + \frac{1}{2}\|x_{n+1}-x_0\|^2 \\
&\le 2d_{n+1}^2 + d_{n+1}\|x_{n+1}-x_0\| + \frac{1}{2}\sum_{k=0}^{n}\|x_k - x_{k+1}\|^2.
\end{aligned}$$

Rearranging the terms, we obtain

$$d_{n+1}\|x_{n+1}-x_0\| \le 2d_{n+1}^2 + \frac{1}{2}\sum_{k=0}^{n}\|x_k - x_{k+1}\|^2 = 2d_{n+1}^2 + \frac{1}{2}\sum_{k=0}^{n}\eta_k^2\|g_k\|^2.$$

It remains to divide this inequality by d_{n+1} to get the desired claim. □

Lemma 4. *Assuming the weights $\omega_0, \ldots, \omega_n$ are positive, it holds for the iterates of Algorithm 1:*

$$\sum_{k=0}^{n}\frac{d_k^4\omega_k^2\|g_k\|^2}{d_k^2 G^2 + \sum_{i=0}^{k} d_i^2\omega_i^2\|g_i\|^2} \le d_n^2 \log\left(1 + \sum_{k=0}^{n}\omega_k^2\right). \tag{7}$$

Proof. The lemma follows straightforwardly from Proposition 2 by substituting $a_k = \frac{d_k^2}{d_n^2}\omega_k^2\|g_k\|^2$ for k from 0 to n:

$$\begin{aligned}
\sum_{k=0}^{n}\frac{d_k^4\omega_k^2\|g_k\|^2}{d_k^2 G^2 + \sum_{i=0}^{k} d_i^2\omega_i^2\|g_i\|^2} &= d_n^2\sum_{k=0}^{n}\frac{\frac{d_k^2}{d_n^2}\omega_k^2\|g_k\|^2}{G^2 + \sum_{i=0}^{k}\frac{d_i^2}{d_k^2}\omega_i^2\|g_i\|^2} \\
&\overset{d_k \le d_n}{\le} d_n^2\sum_{k=0}^{n}\frac{\frac{d_k^2}{d_n^2}\omega_k^2\|g_k\|^2}{G^2 + \sum_{i=0}^{k}\frac{d_i^2}{d_n^2}\omega_i^2\|g_i\|^2} \\
&\overset{(5)}{\le} d_n^2\left(\log\left(G^2 + \sum_{k=0}^{n}\frac{d_k^2}{d_n^2}\omega_k^2\|g_k\|^2\right) - \log(G^2)\right) \\
&\le d_n^2\log\left(1 + \sum_{k=0}^{n}\omega_k^2\right),
\end{aligned}$$

where in the last step we used $\frac{d_k^2}{d_n^2}\omega_k^2\|g_k\|^2 \le \omega_k^2 G^2$. □

Let us restate Theorem 1:

Theorem 6 (Same as Theorem 1)**.** *Given any weights $1 \le \omega_0 \le \dots \le \omega_n$, the functional gap of the average iterate of Algorithm 1 converges as*

$$f(\overline{x}_n) - f_* \le \sqrt{2\omega_n} DG \frac{2d_{n+1} + d_{n+1}\log(1 + \sum_{k=0}^n \omega_k^2)}{\sqrt{\sum_{k=0}^n \omega_k d_k^2}}.$$

Proof. The first steps in the proof follow the same lines as the theory in (Defazio & Mishchenko, 2023), but we still provide them for completeness.

Firstly, let us continue developing the bound proved in the proof of Lemma 2:

$$\begin{aligned}
\sum_{k=0}^n \eta_k (f(x_k) - f_*) &\le \|x_0 - x_{n+1}\| D + \sum_{k=0}^n \eta_k \langle g_k, x_k - x_0 \rangle \\
&= \|x_0 - x_{n+1}\| D + \sum_{k=0}^n \langle x_k - x_{k+1}, x_k - x_0 \rangle \\
&= \|x_0 - x_{n+1}\| D + \frac{1}{2}\sum_{k=0}^n \left[\|x_k - x_{k+1}\|^2 + \|x_k - x_0\|^2 - \|x_{k+1} - x_0\|^2 \right] \\
&\le \|x_0 - x_{n+1}\| D + \frac{1}{2}\sum_{k=0}^n \|x_k - x_{k+1}\|^2.
\end{aligned}$$

We upper bound the first term with the help of Lemma 3:

$$\sum_{k=0}^n \eta_k (f(x_k) - f_*) \le 2Dd_{n+1} + \frac{D}{2d_{n+1}} \sum_{k=0}^n \eta_k^2 \|g_k\|^2 + \frac{1}{2}\sum_{k=0}^n \eta_k^2 \|g_k\|^2.$$

Since by Lemma 2, $1 \le \frac{D}{d_{n+1}}$, we can simplify it to

$$\begin{aligned}
\sum_{k=0}^n \eta_k (f(x_k) - f_*) &\le 2Dd_{n+1} + \frac{D}{d_{n+1}} \sum_{k=0}^n \eta_k^2 \|g_k\|^2 \\
&= 2Dd_{n+1} + \frac{D}{d_{n+1}} \sum_{k=0}^n \frac{d_k^4 \omega_k^2}{d_k^2 G^2 + \sum_{i=0}^k d_i^2 \omega_i^2 \|g_i\|^2} \|g_k\|^2 \\
&\overset{(7)}{\le} 2Dd_{n+1} + \frac{D}{d_{n+1}} d_n^2 \log\Bigl(1 + \sum_{k=0}^n \omega_k^2\Bigr).
\end{aligned}$$

Using the convexity of f, we can apply Jensen's inequality on the iterate $\overline{x}_n$ to get

$$\begin{aligned}
f(\overline{x}_n) - f_* &\le \frac{1}{\sum_{k=0}^n \eta_k} \sum_{k=0}^n \eta_k (f(x_k) - f_*) \le \frac{2Dd_{n+1} + \frac{D}{d_{n+1}} d_n^2 \log(1 + \sum_{k=0}^n \omega_k^2)}{\sum_{k=0}^n \eta_k} \\
&\le D \frac{2d_{n+1} + d_{n+1}\log(1 + \sum_{k=0}^n \omega_k^2)}{\sum_{k=0}^n \eta_k}.
\end{aligned} \tag{8}$$

Notice that $\|g_i\| \le G$ and $\omega_i \le \omega_n$, so

$$\eta_k = \frac{d_k^2 \omega_k}{\sqrt{d_k^2 G^2 + \sum_{i=0}^k d_i^2 \omega_i^2 \|g_i\|^2}} \ge \frac{d_k^2 \omega_k}{G\sqrt{d_k^2 + \sum_{i=0}^k d_i^2 \omega_i^2}} \ge \frac{d_k^2 \omega_k}{G\sqrt{2\omega_n}\sqrt{\sum_{i=0}^k d_i^2 \omega_i}}.$$

Summing over k from 0 to n gives

$$\sum_{k=0}^n \eta_k \ge \frac{1}{\sqrt{2\omega_n} G} \sum_{k=0}^n \frac{d_k^2 \omega_k}{\sqrt{\sum_{i=0}^k d_i^2 \omega_i}} \overset{(4)}{\ge} \frac{1}{\sqrt{2\omega_n} G} \sqrt{\sum_{k=0}^n d_k^2 \omega_k}.$$

Hence,

$$f(\overline{x}_n) - f_* \overset{(8)}{\leq} \sqrt{2\omega_n} DG \frac{d_{n+1}}{\sqrt{\sum_{k=0}^{n} d_k^2 \omega_k}} \left(2 + \log\left(1 + \sum_{k=0}^{n} \omega_k^2\right)\right).$$

□

Corollary 1. *Consider Algorithm 1 with $n \geq 2\log_2\left(\frac{2D}{d_0}\right)$ and define $t = \arg\min_{k \leq n} \frac{d_k}{\sqrt{\sum_{i=0}^{k} d_i^2}}$. If we choose weights $\omega_k = 1$, then it holds*

$$f(\overline{x}_t) - f_* \leq 4\sqrt{2} DG \frac{2 + \log(n+2)}{\sqrt{n}} \sqrt{\log_2\left(\frac{2D}{d_0}\right)}.$$

Proof. Substituting ω_k in the bound of Theorem 1, we get for any n

$$f(\overline{x}_n) - f_* \overset{(8)}{\leq} \sqrt{2} DG \frac{d_{n+1}}{\sqrt{\sum_{k=0}^{n} d_k^2}} \log(n+2).$$

If $n \geq 2\log_2\left(\frac{2D}{d_0}\right)$, using the definition of t, the result of Lemma 1 and the property $d_n \leq D$, we obtain

$$\begin{aligned} f(\overline{x}_t) - f_* &\leq \sqrt{2} DG \min_{k \leq n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^{k} d_i^2}} \left(2 + \log(n+2)\right) \\ &\leq 4\sqrt{2} DG \frac{2 + \log(n+2)}{\sqrt{n}} \sqrt{\log_2\left(\frac{2D}{d_0}\right)}. \end{aligned}$$

If, in contrast, $n \leq 2\log_2\left(\frac{2D}{d_0}\right)$, then it trivially holds by convexity and Cauchy-Schwarz

$$f(\overline{x}_t) - f_* \leq f(x_0) - f_* \leq \langle g_0, x_0 - x_* \rangle \leq \|g_0\| \|x_0 - x_*\| \leq GD$$

□

Corollary 2. *Choose any $p \geq 0$ ans set the weights to be $\omega_k = (k+1)^p$. Then,*

$$f(\overline{x}_n) - f_* = \mathcal{O}\left(\frac{DG(p+1)^{3/2} \log(n+1)}{\sqrt{n+1}}\right).$$

Proof. Since the sequence $d_0, d_1, \ldots$ is non-decreasing and upper bounded by D, there exists an index $\hat{n}$ such that $d_k \leq 2d_{\hat{n}}$ for any $k \geq \hat{n}$. Moreover, we have for $n \geq 2(\hat{n}+1)$

$$\sum_{k=\hat{n}}^{n} \omega_k \geq \frac{1}{p+1}\left((n+1)^{p+1} - (\hat{n}+1)^{p+1}\right) \geq \frac{1}{2(p+1)}(n+1)^{p+1}$$

and

$$\sum_{k=0}^{n} \omega_k^2 = \sum_{k=1}^{n+1} k^{2p} \leq \int_2^{n+2} x^{2p} dx \leq \frac{1}{2p+1}(n+2)^{2p+1} - 1 \leq (n+2)^{2p+1} - 1.$$

Let us plug this into the bound of Theorem 1 for $n \geq 2(\hat{n}+1)$:

$$\begin{aligned} f(\overline{x}_n) - f_* &\leq \sqrt{2\omega_n} DG \frac{d_{n+1}}{\sqrt{\sum_{k=0}^{n} d_k^2 \omega_k}} \left(2 + \log\left(1 + \sum_{k=0}^{n} \omega_k^2\right)\right) \\ &\leq \frac{2d_{\hat{n}} \sqrt{2(n+1)^p} DG}{\sqrt{d_{\hat{n}}^2 \sum_{k=\hat{n}}^{n} \omega_k}} \left(2 + (2p+1)\log(n+2)\right) \\ &\leq \frac{4\sqrt{p+1} DG}{\sqrt{n+1}} \left(2 + (2p+1)\log(n+2)\right) = \mathcal{O}\left(\frac{DG(p+1)^{3/2}\log(n+1)}{\sqrt{n+1}}\right), \end{aligned}$$

which matches our claim. □

Notice that the bound in Corollary 2 does not depend on D/d_0. This is only possible asymptotically for a large enough k and a similar bound without weights was presented by Defazio & Mishchenko (2023).

A.4. GD smooth analysis

We will need the following technical result to show convergence on smooth functions.

Lemma 5. *If numbers $a > 1, b > 0$ are such that $a \le b(1 + \log a)$, then it also holds $a \le 2b(1 + \log b)$.*

Proof. Firstly, let us plug-in the upper bound into itself:

$$a \le b(1 + \log a) \le b(1 + \log(b(1 + \log a))) = b(1 + \log b + \log(1 + \log a)).$$

Therefore, it is enough to show that $\log(1 + \log a) \le 1 + \log b$. Notice that for any $a > 1$ it holds $(1 + \log a)^2 \le 2a$, so $1 + \log a \le \frac{2a}{1+\log a} \le 2b$ and $\log(1 + \log a) \le \log(2b) < 1 + \log b$. $\square$

Theorem 7 (Same as Theorem 3)**.** *Assume f is L-smooth, and set $G = 0$ and $\omega_0 = \cdots = \omega_n = 1$ in Algorithm 1. Then, we get the following convergence guarantee:*

$$f(\overline{x}_t) - f_* = \mathcal{O}\left(\frac{\log_{2+}\left(\frac{D}{d_0}\right) \log^2_{2+}\left(\frac{LD^2}{d_0\|g_0\|}\right)}{n}\right),$$

where we used $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2} \sum_{k=0}^n d_k^2 x_k$ and $t = \arg\min_{k \le n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^k d_i^2}}$.

Proof. Recall that we have established the upper bound

$$\begin{aligned}\sum_{k=0}^n \eta_k (f(x_k) - f_*) &\le 2Dd_{n+1} + \frac{D}{d_{n+1}} \sum_{k=0}^n \eta_k^2 \|g_k\|^2 \\ &= 2Dd_{n+1} + \frac{D}{d_{n+1}} \sum_{k=0}^n \frac{d_k^4}{\sum_{i=0}^k d_i^2 \|g_i\|^2} \|g_k\|^2.\end{aligned}$$

Firstly, let us work on the right-hand side. Since we do not assume that $\|g_k\|$ is bounded in the smooth case, we keep the gradients in the upper bound:

$$\begin{aligned}\sum_{k=0}^n \frac{d_k^4 \|g_k\|^2}{\sum_{i=0}^k d_i^2 \|g_i\|^2} &= d_0^2 + \sum_{k=1}^n \frac{d_k^4 \|g_k\|^2}{\sum_{i=0}^k d_i^2 \|g_i\|^2} \\ &\le d_0^2 + d_n^2 \sum_{k=1}^n \frac{d_k^2 \|g_k\|^2}{\sum_{i=0}^k d_i^2 \|g_i\|^2} \\ &\overset{(5)}{\le} d_0^2 + d_n^2 \log\Big(d_0^2 \|g_0\|^2 + \sum_{k=1}^n d_k^2 \|g_k\|^2\Big) - \log(d_0^2 \|g_0\|^2) \\ &= d_0^2 + d_n^2 \log\Big(1 + \sum_{k=1}^n \frac{d_k^2 \|g_k\|^2}{d_0^2 \|g_0\|^2}\Big) \\ &= d_0^2 + d_n^2 \log\Big(\sum_{k=0}^n \frac{d_k^2 \|g_k\|^2}{d_0^2 \|g_0\|^2}\Big).\end{aligned}$$

Secondly, we shall work on the left-hand side of the first inequality. The smoothness assumption implies $\|g_i\|^2 \le 2L(f(x_i) - f_*)$, so

$$\eta_k = \frac{d_k^2}{\sqrt{\sum_{i=0}^k d_i^2 \|g_i\|^2}} \ge \frac{d_k^2}{\sqrt{2L}\sqrt{\sum_{i=0}^k d_i^2 (f(x_i) - f_*)}}.$$

Combining this lower bound with the first part of Proposition 1 with $a_k = d_k^2(f(x_k) - f_*)$ gives

$$\sum_{k=0}^{n} \eta_k(f(x_k) - f_*) \geq \sum_{k=0}^{n} \frac{d_k^2(f(x_k) - f_*)}{\sqrt{2L}\sqrt{\sum_{i=0}^{k} d_i^2(f(x_i) - f_*)}} \geq \frac{1}{\sqrt{2L}}\sqrt{\sum_{k=0}^{n} d_k^2(f(x_k) - f_*)}.$$

All together, the bounds above yield

$$\frac{1}{\sqrt{2L}}\sqrt{\sum_{k=0}^{n} d_k^2(f(x_k) - f_*)} \leq 2Dd_{n+1} + \frac{D}{d_{n+1}}d_0^2 + \frac{D}{d_{n+1}}d_n^2 \log\Big(\sum_{k=0}^{n} \frac{d_k^2\|g_k\|^2}{d_0^2\|g_0\|^2}\Big).$$

Since $\frac{D}{d_{n+1}}d_0^2 \leq \frac{D}{d_{n+1}}d_n^2 \leq \frac{D}{d_{n+1}}d_{n+1}^2 = Dd_{n+1}$, we have

$$\begin{aligned}\frac{1}{\sqrt{2L}}\sqrt{\sum_{k=0}^{n} d_k^2(f(x_k) - f_*)} &\leq Dd_{n+1}\left(3 + \log\Big(\sum_{k=0}^{n} \frac{d_k^2\|g_k\|^2}{d_0^2\|g_0\|^2}\Big)\right) \\ &\overset{(3)}{\leq} Dd_{n+1}\left(3 + \log\Big(\frac{2L}{d_0^2\|g_0\|^2}\sum_{k=0}^{n} d_k^2(f(x_k) - f_*)\Big)\right).\end{aligned}$$

Denote $F_n = \sqrt{\frac{2L}{d_0^2\|g_0\|^2}\sum_{k=0}^{n} d_k^2(f(x_k) - f_*)}$ and notice that the bounds above imply $F_n \geq 1$. Then, we can rewrite the last bound above as

$$\frac{d_0\|g_0\|}{2L}F_n \leq Dd_{n+1}(3 + \log F_n^2) = Dd_{n+1}(3 + 2\log F_n) < 3Dd_{n+1}(1 + \log F_n).$$

Applying Lemma 5 with $a = F_n$ and $b = \frac{6LDd_{n+1}}{d_0\|g_0\|}$, we get

$$F_n \leq \frac{12LDd_{n+1}}{d_0\|g_0\|}\Big(1 + \log\Big(\frac{6LDd_{n+1}}{d_0\|g_0\|}\Big)\Big) \leq \frac{12LDd_{n+1}}{d_0\|g_0\|}\Big(1 + \log\Big(\frac{6LD^2}{d_0\|g_0\|}\Big)\Big).$$

Squaring both sides and rearranging yields

$$\sum_{k=0}^{n} d_k^2(f(x_k) - f_*) \leq 72LD^2 d_{n+1}^2\Big(1 + \log\Big(\frac{6LD^2}{d_0\|g_0\|}\Big)\Big)^2.$$

Now we can apply Jensen's inequality:

$$f(\overline{x}_n) - f_* \leq \frac{1}{\sum_{k=0}^{n} d_k^2}\sum_{k=0}^{n} d_k^2(f(x_k) - f_*) \leq \frac{d_{n+1}^2}{\sum_{k=0}^{n} d_k^2}72LD^2\Big(1 + \log\Big(\frac{6LD^2}{d_0\|g_0\|}\Big)\Big)^2.$$

Taking the minimum of the right-hand side over n, we can replace the first ratio in the right-hand side with $4\log_{2+}(D^2/d_0^2) = 8\log_{2+}(D^2/d_0)$. □

A.5. DA analysis

Lemma 6. *Considering Algorithm 2, we have*

$$\|s_{n+1}\| \leq \frac{2d_{n+1}}{\gamma_{n+1}} + \frac{\sum_{k=0}^{n} \gamma_k d_k^4\|g_k\|^2}{2d_{n+1}}.$$

Proof. When studying Dual Averaging, we need to introduce an extra sequence that lower bounds $\overline{d}_n$:

$$\overline{d}_{n+1} \overset{\text{def}}{=} \frac{\gamma_{n+1}\|s_{n+1}\|^2 - \sum_{k=0}^{n} \gamma_k d_k^4 \|g_k\|^2}{2\|s_{n+1}\|}.$$

Let us show that $\hat{d}_{n+1} \geq \overline{d}_{n+1}$ by comparing their numerators:

$$\begin{aligned}
\hat{d}_{n+1}\|s_{n+1}\| &= \sum_{k=0}^{n} d_k^2 \langle g_k, x_0 - x_k \rangle = \sum_{k=0}^{n} d_k^2 \gamma_k \langle g_k, s_k \rangle = \sum_{k=0}^{n} \gamma_k \langle s_{k+1} - s_k, s_k \rangle \\
&= \sum_{k=0}^{n} \frac{\gamma_k}{2} \left[\|s_{k+1}\|^2 - \|s_{k+1} - s_k\|^2 - \|s_k\|^2 \right] \\
&= \frac{\gamma_n}{2} \|s_{n+1}\|^2 + \frac{1}{2} \sum_{k=0}^{n} (\gamma_k - \gamma_{k+1}) \|s_{k+1}\|^2 - \frac{1}{2} \sum_{k=0}^{n} \gamma_k d_k^4 \|g_k\|^2 \\
&\overset{\gamma_k \geq \gamma_{k+1}}{\geq} \frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 - \frac{1}{2} \sum_{k=0}^{n} \gamma_k d_k^4 \|g_k\|^2 \\
&= \overline{d}_{n+1}\|s_{n+1}\|.
\end{aligned}$$

Using the definition of $\overline{d}_{n+1}$, and the property $\overline{d}_{n+1} \leq \hat{d}_{n+1} \leq d_{n+1}$, we derive

$$\frac{\gamma_{n+1}}{2} \left\| s_{n+1} \right\|^2 - \frac{1}{2} \sum_{k=0}^{n} \gamma_k d_k^4 \left\| g_k \right\|^2 = \overline{d}_{n+1} \left\| s_{n+1} \right\| \leq d_{n+1} \left\| s_{n+1} \right\|.$$

Using inequality $2\alpha\beta \leq \alpha^2 + \beta^2$ with $\alpha^2 = \frac{2d_{n+1}^2}{\gamma_{n+1}}$ and $\beta^2 = \frac{\gamma_{n+1}}{2}\|s_{n+1}\|^2$ and then the bound above, we establish

$$\begin{aligned}
2d_{n+1}\|s_{n+1}\| = 2\alpha\beta \leq \alpha^2 + \beta^2 &= \frac{2d_{n+1}^2}{\gamma_{n+1}} + \frac{\gamma_{n+1}}{2}\|s_{n+1}\|^2 \\
&\leq \frac{2d_{n+1}^2}{\gamma_{n+1}} + d_{n+1}\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^{n} \gamma_k d_k^4 \|g_k\|^2.
\end{aligned}$$

Rearranging the terms, we obtain

$$d_{n+1}\|s_{n+1}\| \leq \frac{2d_{n+1}^2}{\gamma_{n+1}} + \frac{1}{2}\sum_{k=0}^{n} \gamma_k d_k^4 \|g_k\|^2.$$

It remains to divide both sides by d_{n+1}. □

Lemma 7. *The Dual Averaging algorithm (Algorithm 2) satisfies*

$$\sum_{k=0}^{n} d_k^2 (f(x_k) - f_*) \leq (D - \hat{d}_{n+1}) \|s_{n+1}\|. \tag{9}$$

Proof. Summing inequality $f(x_k) - f_* \leq \langle g_k, x_k - x_* \rangle$ with weights d_k^2, we get

$$\sum_{k=0}^{n} d_k^2 (f(x_k) - f_*) \leq \sum_{k=0}^{n} d_k^2 \langle g_k, x_k - x_* \rangle = \sum_{k=0}^{n} d_k^2 \langle g_k, x_0 - x_* \rangle + \sum_{k=0}^{n} d_k^2 \langle g_k, x_k - x_0 \rangle.$$

Using Cauchy-Schwarz on the first product in the right-hand side and then telescoping the second sum, we obtain

$$\begin{aligned}
\sum_{k=0}^{n} d_k^2 (f(x_k) - f_*) &\leq \|s_{n+1}\| \|x_0 - x_*\| + \sum_{k=0}^{n} d_k^2 \langle g_k, x_k - x_0 \rangle \\
&= \|s_{n+1}\| D - \hat{d}_{n+1} \|s_{n+1}\|.
\end{aligned}$$

□

Next, we restate and prove Theorem 2:

Theorem 8 (Same as Theorem 2)**.** *For Algorithm 2, it holds that:*

$$f(\overline{x}_t) - f_* \le \frac{4GD}{\sqrt{n}}\sqrt{\log_2\Big(\frac{2D}{d_0}\Big)},$$

where $t = \arg\min_{k\le n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^k d_i^2}}$.

Proof. Let us sum inequality $d_k^2(f(x_k) - f_*) \ge 0$ and then apply Lemma 7:

$$0 \le \sum_{k=0}^n d_k^2(f(x_k) - f_*) \overset{(9)}{\le} (D - \hat{d}_{n+1})\|s_{n+1}\|.$$

Clearly, this implies that $\hat{d}_{n+1} \le D$, and by induction it follows that $d_{n+1} \le D$ as well. Now let us upper bound the functional values:

$$\begin{aligned}
\sum_{k=0}^n d_k^2(f(x_k) - f_*) &\overset{(9)}{\le} D\|s_{n+1}\| - \sum_{k=0}^n \gamma_k d_k^2 \langle g_k, s_k\rangle \\
&= D\|s_{n+1}\| - \sum_{k=0}^n \gamma_k \langle s_{k+1} - s_k, s_k\rangle \\
&= D\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^n \gamma_k \left(\|s_{k+1} - s_k\|^2 + \|s_k\|^2 - \|s_{k+1}\|^2\right) \\
&= D\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^n \gamma_k \|s_{k+1} - s_k\|^2 + \frac{1}{2}\sum_{k=0}^n (\gamma_k - \gamma_{k-1})\|s_k\|^2 - \frac{\gamma_n}{2}\|s_{n+1}\|^2.
\end{aligned}$$

We can drop the last two terms since $\gamma_k \le \gamma_{k-1}$:

$$\begin{aligned}
\sum_{k=0}^n d_k^2(f(x_k) - f_*) &\le D\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^n \gamma_k \|s_{k+1} - s_k\|^2 \\
&= D\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^n \gamma_k d_k^4 \|g_k\|^2.
\end{aligned}$$

The first term in the right-hand side is readily bounded by Lemma 6:

$$\begin{aligned}
\sum_{k=0}^n d_k^2(f(x_k) - f_*) &\le D\|s_{n+1}\| + \frac{1}{2}\sum_{k=0}^n \gamma_k d_k^4 \|g_k\|^2 \\
&\le \frac{2Dd_{n+1}}{\gamma_{n+1}} + \frac{D}{2d_{n+1}}\sum_{k=0}^n \gamma_k d_k^4 \|g_k\|^2 + \frac{1}{2}\sum_{k=0}^n \gamma_k d_k^4 \|g_k\|^2 \\
&\overset{d_k \le d_n \le d_{n+1}}{\le} \frac{2Dd_{n+1}}{\gamma_{n+1}} + Dd_n \sum_{k=0}^n \gamma_k d_k^2 \|g_k\|^2.
\end{aligned}$$

Algorithm 4 Prodigy (Coordinate-wise Dual Averaging version)

Input: $d_0 > 0, x_0, G_\infty \ge 0$; $s_0 = 0 \in \mathbb{R}^p$, $\mathbf{1} = (1, \dots, 1)^\top \in \mathbb{R}^p$
for $k = 0$ **to** n **do**
  $g_k \in \partial f(x_k)$
  $s_{k+1} = s_k + d_k^2 g_k$
  $\hat{d}_{k+1} = \dfrac{\sum_{i=0}^k d_i^2 \langle g_i, x_0 - x_i \rangle}{\|s_{k+1}\|_1}$
  $d_{k+1} = \max(d_k, \hat{d}_{k+1})$
  $a_{k+1}^2 = d_{k+1}^2 G_\infty^2 \mathbf{1} + \sum_{i=0}^k d_i^2 g_i^2$ ▷ Coordinate-wise square
  $\mathbf{A}_{k+1} = \mathrm{diag}(a_{k+1})$
  $x_{k+1} = x_0 - \mathbf{A}_{k+1}^{-1} s_{k+1}$
end for
Return $\overline{x}_n = \frac{1}{\sum_{k=0}^n d_k^2} \sum_{k=0}^n d_k^2 x_k$

Then, apply Proposition 1:

$$\begin{aligned}
\sum_{k=0}^n d_k^2 (f(x_k) - f_*) &\le \frac{2D}{\gamma_{n+1}} + D d_n \sum_{k=0}^n \gamma_k d_k^2 \|g_k\|^2 \\
&= \frac{2D}{\gamma_{n+1}} + D d_n \sum_{k=0}^n \frac{1}{\sqrt{d_k^2 G^2 + \sum_{i=0}^{k-1} d_i^2 \|g_i\|^2}} d_k^2 \|g_k\|^2 \\
&\le \frac{2D}{\gamma_{n+1}} + D d_n \sum_{k=0}^n \frac{1}{\sqrt{d_k^2 \|g_k\|^2 + \sum_{i=0}^{k-1} d_i^2 \|g_i\|^2}} d_k^2 \|g_k\|^2 \\
&\overset{(4)}{\le} \frac{2D}{\gamma_{n+1}} + 2 D d_n \sqrt{\sum_{k=0}^n d_k^2 \|g_k\|^2}.
\end{aligned}$$

Let us now bound each gradient norm using $\|g_k\| \le G$:

$$\sum_{k=0}^n d_k^2 (f(x_k) - f_*) \le 4 D d_{n+1} \sqrt{\sum_{k=0}^n d_k^2 \|g_k\|^2} \le 4 G D d_{n+1} \sqrt{\sum_{k=0}^n d_k^2}.$$

Thus, we get the following convergence rate:

$$\begin{aligned}
f(\overline{x}_t) - f_* &\le \frac{4 G D d_{t+1} \sqrt{\sum_{k=0}^t d_k^2}}{\sum_{k=0}^t d_k^2} = \frac{4 G D d_{t+1}}{\sqrt{\sum_{k=0}^t d_k^2}} = \min_{t' < n} \frac{4 G D d_{t'+1}}{\sqrt{\sum_{k=0}^{t'} d_k^2}} \\
&\le \frac{4 G D}{\sqrt{n}} \sqrt{\log_{2+}\left(\frac{D}{d_0}\right)}.
\end{aligned}$$

□

A.6. Coordinate-wise Prodigy

Here we study Algorithm 4. The theory in this section follows closely the analysis in Section A.5. There are only a few minor differences such as the use of weighted norms, which we define as $\langle x, y \rangle_{\mathbf{A}^{-1}} = x^\top \mathbf{A}^{-1} y$ for any matrix $\mathbf{A} \succcurlyeq 0$. In addition, we use ℓ_∞ norm for the distance term and for the gradients, see the assumption below.

Assumption 3. *The gradients are upper bounded coordinate-wise:* $\|g_k\|_\infty \le G_\infty$.

We begin with the analogue of Lemma 6:

Lemma 8. *It holds for the iterates of Algorithm 4:*

$$\|s_{n+1}\|_1 \le 2d_{n+1}\|a_{n+1}\|_1 + \frac{1}{2d_{n+1}}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}}.$$

Proof. As in the proof of Lemma 6, let us introduce an extra sequence $\overline{d}_n$:

$$\overline{d}_{n+1} \overset{\text{def}}{=} \frac{\|s_{n+1}\|^2_{\mathbf{A}_{n+1}^{-1}} - \sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}}}{2\|s_{n+1}\|_1}.$$

The next step is to show that $\hat{d}_{n+1} \ge \overline{d}_{n+1}$ by comparing the numerators:

$$\begin{aligned}
\hat{d}_{n+1}\|s_{n+1}\|_1 &= \sum_{k=0}^{n} d_k^2 \langle g_k, x_0 - x_k\rangle = \sum_{k=0}^{n} d_k^2 \langle g_k, s_k\rangle_{\mathbf{A}_k^{-1}} = \sum_{k=0}^{n} \langle s_{k+1} - s_k, s_k\rangle_{\mathbf{A}_k^{-1}} \\
&= \sum_{k=0}^{n} \frac{1}{2}\left[\|s_{k+1}\|^2_{\mathbf{A}_k^{-1}} - \|s_{k+1} - s_k\|^2_{\mathbf{A}_k^{-1}} - \|s_k\|^2_{\mathbf{A}_k^{-1}}\right] \\
&= \frac{1}{2}\|s_{n+1}\|^2_{\mathbf{A}_n^{-1}} + \frac{1}{2}\sum_{k=0}^{n} \|s_{k+1}\|^2_{\mathbf{A}_k^{-1} - \mathbf{A}_{k+1}^{-1}} - \frac{1}{2}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} \\
&\overset{\mathbf{A}_k^{-1} \succcurlyeq \mathbf{A}_{k+1}^{-1}}{\ge} \frac{1}{2}\|s_{n+1}\|^2_{\mathbf{A}_{n+1}^{-1}} - \frac{1}{2}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} \\
&= \overline{d}_{n+1}\|s_{n+1}\|_1.
\end{aligned}$$

Using the definition of $\overline{d}_{n+1}$, and the property $\overline{d}_{n+1} \le \hat{d}_{n+1} \le d_{n+1}$, we derive

$$\frac{1}{2}\|s_{n+1}\|^2_{\mathbf{A}_{n+1}^{-1}} - \frac{1}{2}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} = \overline{d}_{n+1}\|s_{n+1}\|_1 \le d_{n+1}\|s_{n+1}\|_1.$$

Using inequality $2\alpha\beta \le \alpha^2 + \beta^2$ with $\alpha^2 = 2d_{n+1}^2 a_{(n+1)i}$ and $\beta^2 = \frac{1}{2a_{(n+1)i}} s^2_{(n+1)i}$ for $i = 1, \ldots, p$ and then the bound above, we establish

$$\begin{aligned}
2d_{n+1}\|s_{n+1}\|_1 &= \sum_{i=1}^{p} d_{n+1}|s_{(n+1)i}| \le \sum_{i=1}^{p}\left(2d_{n+1}^2 a_{(n+1)i} + \frac{1}{2a_{(n+1)i}} s^2_{(n+1)i}\right) \\
&= 2d_{n+1}^2\|a_{n+1}\|_1 + \frac{1}{2}\|s_{n+1}\|_{\mathbf{A}_{n+1}^{-1}} \\
&\le 2d_{n+1}^2\|a_{n+1}\|_1 + d_{n+1}\|s_{n+1}\|_1 + \frac{1}{2}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}}.
\end{aligned}$$

Rearranging the terms, we get

$$d_{n+1}\|s_{n+1}\|_1 \le 2d_{n+1}^2\|a_{n+1}\|_1 + \frac{1}{2}\sum_{k=0}^{n} d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}}.$$

It remains to divide both sides by d_{n+1}. □

The next lemma is similar to Lemma 8 except that it uses ℓ_∞ norm for the distance to a solution and ℓ_1 norm for the weighted gradient sum s_n.

Lemma 9. *The coordinate-wise version of Prodigy (Algorithm 4) satisfies*

$$\sum_{k=0}^{n} d_k^2 (f(x_k) - f_*) \le (D_\infty - \hat{d}_{n+1})\|s_{n+1}\|_1, \tag{10}$$

where $D_\infty = \|x_0 - x_*\|_\infty$.

Proof. Summing inequality $f(x_k) - f_* \le \langle g_k, x_k - x_* \rangle$ with weights d_k^2, we get

$$\sum_{k=0}^n d_k^2 (f(x_k) - f_*) \le \sum_{k=0}^n d_k^2 \langle g_k, x_k - x_* \rangle = \sum_{k=0}^n d_k^2 \langle g_k, x_0 - x_* \rangle + \sum_{k=0}^n d_k^2 \langle g_k, x_k - x_0 \rangle.$$

Using Hölder's inequality on the first product in the right-hand side and then telescoping the second sum, we obtain

$$\begin{aligned}\sum_{k=0}^n d_k^2 (f(x_k) - f_*) &\le \|s_{n+1}\|_1 \|x_0 - x_*\|_\infty + \sum_{k=0}^n d_k^2 \langle g_k, x_k - x_0 \rangle \\ &= \|s_{n+1}\|_1 D_\infty - \hat{d}_{n+1} \|s_{n+1}\|.\end{aligned}$$

The use of ℓ_1 norm for the term s_{n+1} above is motivated by the fact that it naturally arises in other parts of the theory. $\square$

Theorem 9. *Algorithm 4 converges with the rate*

$$f(\overline{x}_t) - f_* \le \frac{4pG_\infty D_\infty}{\sqrt{n}} \sqrt{\log_{2+}\left(\frac{D_\infty}{d_0}\right)},$$

where $t = \arg\min_{k \le n} \frac{d_{k+1}}{\sqrt{\sum_{i=0}^k d_i^2}}$.

Proof. From Lemma 9, we get

$$0 \le \sum_{k=0}^n d_k^2 (f(x_k) - f_*) \overset{(10)}{\le} (D_\infty - \hat{d}_{n+1}) \|s_{n+1}\|_1,$$

so we can prove by induction that $d_{n+1} \le D_\infty$. Using the same bounds as before, we get for the average iterate

$$\begin{aligned}\sum_{k=0}^n d_k^2 (f(x_k) - f_*) &\le D_\infty \|s_{n+1}\|_1 - \sum_{k=0}^n d_k^2 \langle g_k, x_0 - x_k \rangle \\ &= D_\infty \|s_{n+1}\|_1 + \frac{1}{2} \sum_{k=0}^n d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} + \frac{1}{2} \sum_{k=0}^n \|s_k\|^2_{\mathbf{A}_k^{-1} - \mathbf{A}_{k+1}^{-1}} - \frac{1}{2} \|s_{n+1}\|^2_{\mathbf{A}_{n+1}^{-1}} \\ &\le D_\infty \|s_{n+1}\|_1 + \frac{1}{2} \sum_{k=0}^n d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}}.\end{aligned}$$

Let us plug in the bound from Lemma 8:

$$\begin{aligned}\sum_{k=0}^n d_k^2 (f(x_k) - f_*) &\le 2D_\infty d_{n+1} \|a_{n+1}\|_1 + \frac{D_\infty}{2d_{n+1}} \sum_{k=0}^n d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} + \frac{1}{2} \sum_{k=0}^n d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} \\ &\overset{d_{n+1} \le D_\infty}{\le} 2D_\infty d_{n+1} \|a_{n+1}\|_1 + \frac{D_\infty}{d_{n+1}} \sum_{k=0}^n d_k^4 \|g_k\|^2_{\mathbf{A}_k^{-1}} \\ &\overset{d_k \le d_n}{\le} 2D_\infty d_{n+1} \|a_{n+1}\|_1 + \frac{D_\infty}{d_{n+1}} d_n^2 \sum_{k=0}^n d_k^2 \|g_k\|^2_{\mathbf{A}_k^{-1}}.\end{aligned}$$

We now apply Proposition 1, and use $g_{kj}^2 \le G_\infty^2$:

$$\begin{aligned}\sum_{k=0}^n d_k^2 (f(x_k) - f_*) &\le 2D_\infty d_{n+1} \|a_{n+1}\|_1 + \frac{D_\infty}{d_{n+1}} d_n^2 \sum_{j=1}^p \sum_{k=0}^n \frac{d_k^2 g_{kj}^2}{\sqrt{d_k^2 G_\infty^2 + \sum_{i=0}^{k-1} d_i^2 g_{ij}^2}} \\ &\le 2D_\infty d_{n+1} \|a_{n+1}\|_1 + \frac{2D_\infty}{d_{n+1}} d_n^2 \sum_{j=1}^p \sqrt{\sum_{k=0}^n d_k^2 g_{kj}^2} \\ &\le 4D_\infty d_{n+1} p G_\infty \sqrt{\sum_{k=0}^n d_k^2}.\end{aligned}$$

Using Lemma 1, we get the rate for $t = \arg\min_{t' \le n} \frac{d_{t'+1}}{\sqrt{\sum_{k=0}^{t'} d_k^2}}$:

$$f(\overline{x}_t) - f_* \le \frac{4pG_\infty D_\infty}{\sqrt{n}} \sqrt{\log_{2+}\left(\frac{D_\infty}{d_0}\right)}.$$

□

B. Lower Complexity Theory

Theorem 10. *Consider any exponentially bounded algorithm for minimizing a convex G-Lipschitz function starting from x_0, which has no knowledge of problem constants G and D. There exists a fixed gradient oracle such that any sequence of $x_{1,\dots}, x_n$, there exists a convex Lipschitz problem f with $G = 1$ and $\|x_0 - x_*\| \le D$ for all minimizing points x_*, consistent with the gradient oracle such that:*

$$\min_{k \le n} f(x_k) - f_* \ge \frac{DG\sqrt{\log_2(D/x_1)}}{2\sqrt{n+1}}.$$

Proof. We consider the construction of a 1D oracle for this problem. Our oracle returns $g_0 = -1$ and $f(x_k) = -x_k$ for all queries. Without loss of generality we assume that $x_k > 0$ for all $k \ge 1$, and $G = 1$.

For each step $k \ge 1$ we define:

$$x_* = 2^{n+1} x_1,$$

and thus $D = |x_0 - x_*| = 2^{k+1} x_1$. and our construction uses the following function value and gradient sequence

$$f(x) = |x - x_*| + x_*.$$

Note that for all query points x, the gradient is negative, and only the left arm of the absolute value function is seen by the algorithm, so the function appears linear for all test points. Using this construction, we have:

$$\begin{aligned}
\min_{k \le n} [f(x_k) - f_*] &= \min_{k \le n} (x_* - x_k) \\
&= 2^{n+1} x_1 - \max_{k \le n} x_k \\
&\ge 2 \cdot 2^n x_1 - 2^n x_1 \\
&= 2^n x_1 \\
&= \frac{1}{2} D_n.
\end{aligned}$$

Now note that:

$$\sqrt{\log_2(D_n/x_1)} = \sqrt{\log_2(2^{n+1})} = \sqrt{n+1}.$$

Combining these two results:

$$\min_{k \le n} f(x_k) - f_* \ge \frac{1}{2} D = \frac{1}{2} DG = \frac{\frac{1}{2} DG\sqrt{\log_2(D/x_1)}}{\sqrt{n+1}}.$$

□

Theorem 11. *D-Adaptation, DoG and Prodigy are exponentially bounded.*

Proof. Consider the D lower bound from D-Adaptation:

$$\hat{d}_{n+1} = \frac{\sum_{k=0}^{n} d_k \gamma_k \langle g_k, s_k \rangle}{\|s_{n+1}\|},$$

with:

$$s_{n+1} = \sum_{k=0}^{n} d_k g_k.$$

Recall that

$$\sum_{k=0}^{n} d_k \gamma_k \langle g_k, s_k \rangle \leq \gamma_{n+1} \|s_{n+1}\|^2 .$$

Note also that $\gamma_{n+1} \leq \frac{1}{G}$. So:

$$d_{n+1} \leq \frac{\frac{1}{G}\|s_{n+1}\|^2}{\|s_{n+1}\|} \leq \frac{1}{G}\left\|\sum_{k=0}^{n} d_k g_k\right\| \leq \sum_{k=0}^{n} d_k.$$

So the sequence d_n is upper bounded by the sequence:

$$a_n = \begin{cases} \sum_{k=0}^{n-1} a_k & n \geq 1 \\ d_0 & n = 0 \end{cases}.$$

This sequence has the following closed form:

$$a_{n+1} = 2^n d_0 \quad \text{for } n \geq 1.$$

We can prove this by induction. The base case is by definition $a_1 = a_0$. Then

$$a_{n+1} = \sum_{k=0}^{n} a_k = \sum_{k=0}^{n-1} a_k + a_n = a_n + a_n = 2a_n = 2^n d_0.$$

Note that for both the Dual Averaging form and the GD form we have, we have:

$$\|x_{n+1} - x_0\| \leq \left\|\frac{1}{G}\sum_{k=0}^{n} d_k g_k\right\| \leq \sum_{k=0}^{n} d_k \leq d_{n+1} \leq 2^n d_0.$$

It follows that D-Adaptation is exponentially bounded. For Prodigy, note that:

$$\gamma_{n+1} \leq \frac{1}{\sqrt{d_{n+1}^2 G^2}} = \frac{1}{d_{n+1} G}.$$

Therefore

$$\begin{aligned} d_{n+1} &\leq \frac{\frac{1}{d_{n+1}G}\|s_{n+1}\|^2}{\|s_{n+1}\|} \leq \frac{1}{d_{n+1}G}\left\|\sum_{k=0}^{n} d_k^2 g_k\right\| \\ &\leq \frac{1}{d_{n+1}}\sum_{k=0}^{n} d_k^2 \\ &\leq \frac{1}{d_{n+1}}\sum_{k=0}^{n} d_k d_{n+1} \\ &= \sum_{k=0}^{n} d_k. \end{aligned}$$

The rest of the argument follows the D-Adaptation case, with:

$$\|x_{n+1} - x_0\| \leq \left\|\frac{1}{d_n G}\sum_{k=0}^{n} d_k^2 g_k\right\| \leq \sum_{k=0}^{n} d_k \leq d_{n+1} \leq 2^n d_0.$$

For DoG, recall the basic DoG step is gradient descent with step sizes:

$$\gamma_k = \frac{\bar{r}_k}{\sqrt{G^2 + \sum_{i=0}^{k} \|g_i\|^2}}.$$

Using the triangle inequality we have:

$$\begin{aligned}
\|x_{k+1} - x_0\| &= \|x_k - \gamma_k g_k - x_0\| \\
&\leq \|x_k - x_0\| + \gamma_k \|g_k\| \\
&\leq \|x_k - x_0\| + \frac{\bar{r}_k}{\sqrt{G^2}} \|g_k\| \\
&\leq \|x_k - x_0\| + \bar{r}_k \\
&\leq 2\bar{r}_k.
\end{aligned}$$

Chaining gives the result.

□

Proposition 3. *Suppose that $d_k \leq cD$ and $\gamma_k \leq d_k/G$. then:*

$$\|x_k - x_0\| \leq (2c+1)^n \|x_1 - x_0\|.$$

Proof. Without loss of generality assume that $G = 1$. Firstly, note that using the absolute value function as constructed in Theorem 5, it's clear that there is always exists a function with $D_k \leq 2\|x_k - x_*\|$ at step k consistent with the sequence of gradients seen so far. Therefore, it must hold that

$$d_k \leq cD_k \leq 2c \|x_k - x_0\|.$$

We prove the result by induction. For the base case, trivially:

$$\|x_1 - x_0\| \leq (2c+1)^1 \|x_1 - x_0\|.$$

For the inductive case:

$$\begin{aligned}
\|x_{k+1} - x_0\| &= \|x_k - \gamma_k g_k - x_0\| \\
&\leq \|x_k - x_0\| + \gamma_k \|g_k\| \\
&\leq \|x_k - x_0\| + \frac{cD_k}{G} \|g_k\| \\
&\leq \|x_k - x_0\| + cD_k \\
&\leq (2c+1) \|x_k - x_0\| \\
&\leq (2c+1)^{n+1} \|x_1 - x_0\|.
\end{aligned}$$

□