The Road Less Scheduled

Aaron Defazio[1]
Fundamental AI Research Team, Meta

Xingyu (Alice) Yang[2]
Fundamental AI Research Team, Meta

Harsh Mehta
Google Research

Konstantin Mishchenko
Samsung AI Center

Ahmed Khaled
Princeton University

Ashok Cutkosky[3]
Boston University

[1]Research Co-lead [2] Engineering Co-lead [3] Senior Author

Abstract

Existing learning rate schedules that do not require specification of the optimization stopping step T are greatly out-performed by learning rate schedules that depend on T. We propose an approach that avoids the need for this stopping time by eschewing the use of schedules entirely, while exhibiting state-of-the-art performance compared to schedules across a wide family of problems ranging from convex problems to large-scale deep learning problems. Our *Schedule-Free* approach introduces no additional hyper-parameters over standard optimizers with momentum. Our method is a direct consequence of a new theory we develop that unifies scheduling and iterate averaging. An open source implementation of our method is available[1].

1 Introduction

The theory of optimization, as applied in machine learning, has been successful at providing precise, prescriptive results for many problems. However, even in the simplest setting of stochastic gradient descent (SGD) applied to convex Lipschitz functions, there are glaring gaps between what our current theory prescribes and the methods used in practice.

Consider the stochastic gradient descent (SGD) step with step size $\gamma > 0$, $z_{t+1} = z_t - \gamma g_t$ where g_t is the stochastic (sub-)gradient at step t (formally defined in Section 1.2) of a convex Lipschitz function f. Although standard practice for many classes of problems, classical convergence theory suggests that the expected loss of this z sequence is *suboptimal*, and that the Polyak-Ruppert (PR) average x of the sequence should be returned instead (Polyak, 1990; Ruppert, 1988):

$$z_{t+1} = z_t - \gamma g_t$$
$$x_{t+1} = (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1},$$

where using $c_{t+1} = 1/(t+1)$ results in $x_t = \frac{1}{T}\sum_{t=1}^{T} z_t$. Despite their theoretical optimality, PR averages give much worse results in practice than using the last-iterate of SGD (Figure 2) — a folk-law result in the field of optimization, and a large theory-practice gap that is often attributed to the mismatch between this simplified problem class and the complexity of problems addressed in practice.

Recently, Zamani and Glineur (2023) and Defazio et al. (2023) showed that the exact worst-case optimal rates can be achieved via carefully chosen learning rate sequences (also known as *schedules*)

[1] https://github.com/facebookresearch/schedule_free

Preprint. Under review.

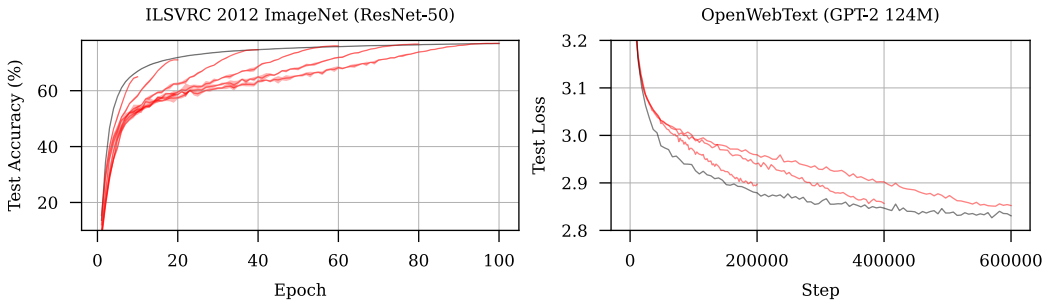

Figure 1: Schedule-Free methods (black) closely track the Pareto frontier of loss v.s. training time in a single run. Both Schedule-Free SGD (left) and AdamW (right) match or exceed the performance of cosine learning rate schedules of varying lengths (red).

alone, without the use of averaging. This result suggests that schedules have, in some sense, the same role to play as PR averaging in optimization. However, schedules have a critical disadvantage: they require setting the optimization stopping time T in advance.

Motivated by the theory-practice gap for Polyak-Ruppert averaging, we ask the following question:

> Do there exist iterate averaging approaches that match the empirical performance of learning rate schedules, without sacrificing theoretical guarantees?

Polyak-Ruppert averaging vs Schedulers

By developing a new link between averaging and learning rate sequences, we introduce a new approach to averaging that maintains the worst-case convergence rate theory of PR averaging, while matching and often exceeding the performance of schedule-based approaches – firmly answering this question in the affirmative.

1.1 Summary of Results

- Our approach does not require the stopping time T to be known or set in advance. It closely tracks the Pareto frontier of loss versus training time during a single training run (Figure 1), while requiring *no additional hyper-parameters* over the base SGD or Adam optimizer.
- Our approach uses an alternative form of momentum. This form has appealing theoretical properties: it is **worst case optimal for any choice of the momentum parameter in the convex Lipschitz setting**, a property that does not hold for traditional momentum.
- Our key theoretical result is a new *online-to-batch* conversion theorem, which establishes the optimality of our method while also unifying several existing online-to-batch theorems.
- We perform, to our knowledge, **one of the largest and most comprehensive machine learning optimization algorithm evaluations** to date, consisting of 28 problems, ranging from logistic regression to large-scale deep learning problems. Schedule-Free methods show strong performance, matching or out-performing heavily-tuned cosine schedules.

Wait how can the non-worst case be better than optimal?

1.2 Notation

Consider the stochastic convex minimization $\min_{x\in\mathbb{R}^d} f(x) = \mathbb{E}_\zeta[f(x,\zeta)]$, where each $f(x,\zeta)$ is Lipschitz and convex in x, and the expectation is taken over the random variable ζ. With a slight abuse of notation, we assume we are given, at time step t and any point y that we choose, an arbitrary sub-gradient $\nabla f(y,\zeta_t)$ from the sub-differential of f.

I think `x` is the online equal-weighted average of the weights (`z`) and `C` (zeta) is the input?

2 Method

We propose the following method, which we call Schedule-Free SGD:

`y` is how much we value the immediate weights w/ the latest gradient change vs valuing the moving average of the weights

$$
\begin{aligned}
y_t &= (1-\beta)z_t + \beta x_t, \\
z_{t+1} &= z_t - \gamma \nabla f(y_t, \zeta_t), \\
x_{t+1} &= (1-c_{t+1})\, x_t + c_{t+1} z_{t+1},
\end{aligned}
$$

I think `z` is the weights that are immediately changed with gradients at each time step `t`. (weights + gradient)

This is the Polyak-Ruppert averaging

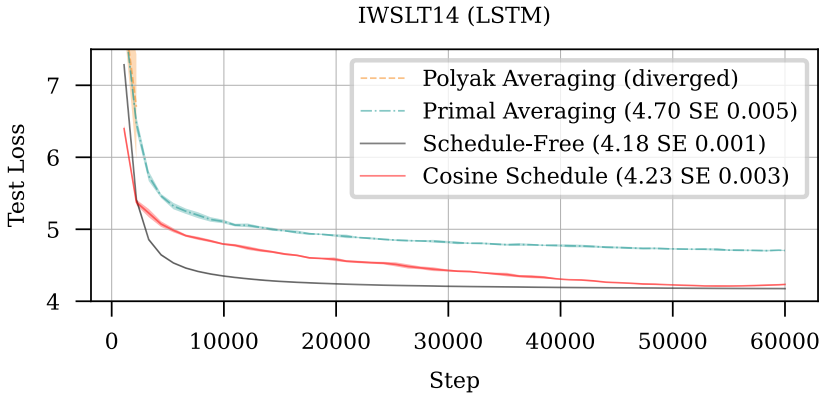

Figure 2: Schedule-Free learning converges faster than classical averaging approaches, often outperforming tuned schedules.

where $c_{t+1} = 1/(t+1)$ and $z_1 = x_1$ is the initial point. Note that with this weighting, the x sequence is just an online equal-weighted average of the z sequence. This method has a momentum parameter β that interpolates between Polyak-Ruppert averaging ($\beta = 0$) and Primal averaging ($\beta = 1$). *Primal averaging* (Nesterov and Shikhman, 2015; Tao et al., 2018; Cutkosky, 2019; Kavis et al., 2019; Sebbouh et al., 2021; Defazio and Gower, 2021; Defazio and Jelassi, 2022), is an approach where the gradient is evaluated at the averaged point x, instead of z:

$$z_{t+1} = z_t - \gamma \nabla f(x_t, \zeta_t)$$
$$x_{t+1} = (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1},$$

gradient of the function weighted by some discount gamma (learning rate)

weights are updated such that they are immediately changed w/ gradients, and don't use the weighted average of the weight changes, but as time (`t`) progresses, it values more the weighted average of the weights and doesn't update from the gradients as much.

this approach maintains the worst-case optimality of PR averaging but is generally considered to converge too slowly to be practical (Figure 2). The advantage of our interpolation is that we get the best of both worlds. We can achieve the fast convergence of Polyak-Ruppert averaging (since the z sequence moves much quicker than the x sequence), while still keeping some coupling between the returned sequence x and the gradient-evaluation locations y, which increases stability (Figure 2). Values of β similar to standard momentum values $\beta \approx 0.9$ appear to work well in practice. We will use the notation $\alpha = 1 - \beta$ when convenient.

In this formulation, $\beta = 0.9$ gives the practical advantages of momentum, dampening the immediate impact of large gradients, resulting in more stable training. To see this, notice that the immediate effect of the gradient g_t at step t is to introduce $(1-\beta)g_t = 0.1g_t$ into the iterate sequence y. This is similar to exponential-moving-average (EMA) momentum, where also $(1-\beta)g_t$ is added into the iterate sequence on step t. However, here the remainder of g_t is very slowly added into y over time, via its place in the average x, whereas with an EMA with $\beta = 0.9$, the majority of the gradient is incorporated within the next 10 steps. So from this viewpoint, the Schedule-Free updates can be seen as a version of momentum that has the same immediate effect, but with a greater delay for adding in the remainder of the gradient. This form of momentum (by interpolation) also has a striking advantage: it does not result in any theoretical slowdown; it gives the optimal worst case (Nesterov, 2013) convergence for the non-smooth convex setting (including constants), for any choice of momentum β between 0 and 1 inclusive:

Theorem 1. *Suppose F is a convex function, and $\zeta_1, \ldots, \zeta_T$ is an i.i.d. sequence of random variables such that $F = \mathbb{E}[f(x, \zeta)]$ for some function f that is G-Lipschitz in x. For any minimizer $x_\star$, define $D = \|x_1 - x_\star\|$ and $\gamma = D/(G\sqrt{T})$. Then for any $\beta \in [0, 1]$, Schedule-Free SGD ensures:*

$$\mathbb{E}[F(x_T) - F(x_\star)] \leq \frac{DG}{\sqrt{T}}$$

The `DG` numerator essentially says that the gradient change for a function is bounded, thus it will converge

`x*` are the weights that minimize the loss I believe (optimal weights)
`D` is the euclidean (manhattan?) distance between the initial and optimal weights
`F` is the expectation of that output from a network
`C` (zeta) is in input sequence
`T` is the last time step

What is `G`? I think it's the Lipschitz constant (meaning the maximum gradient/change multiplier a function can achieve)

In contrast, exponential-moving-average momentum in the non-smooth setting actually hurts the theoretical worst-case convergence rate. The Schedule-Free approach maintains the advantages of momentum (Sutskever et al., 2013) without the potential worst-case slow-down.

2.1 General Theory

The method analyzed in Theorem 1 is actually a special-case of a more general result that incorporates arbitrary online optimization algorithms rather than only SGD, as well as arbitrary time-varying sequences of β_t. The proof is provided in Appendix A.

Theorem 2. *Let F be a convex function. Let $\zeta_1, \dots, \zeta_T$ be an iid sequence such that $F(x) = \mathbb{E}_\zeta[f(x,\zeta)]$. Let $z_1, \dots, z_T$ be arbitrary vectors and let $w_1, \dots, w_T$ and $\beta_1, \dots, \beta_T$ be arbitrary numbers in $[0,1]$ such that z_t, w_t and β_t are independent of $\zeta_t, \dots, \zeta_T$. Set:*

$$x_t = \frac{\sum_{i=1}^t w_i z_i}{\sum_{i=1}^t w_i} = x_{t-1}\underbrace{\left(1 - \frac{w_t}{\sum_{i=1}^t w_i}\right)}_{\triangleq 1-c_t} + \underbrace{\frac{w_t}{\sum_{i=1}^t w_i}}_{\triangleq c_t} z_t$$

`w` is the weight of the timestep

(above is equivalent to `1-c_t`)

(above is equivalent to `c_t`)

`y` is the "target" to get the function gradients from

$$y_t = \beta_t x_t + (1-\beta_t) z_t$$
$$g_t = \nabla f(y_t, \zeta_t)$$

Then we have for all $x_\star$:

$$\mathbb{E}[F(x_T) - F(x_\star)] \leq \frac{\mathbb{E}[\sum_{t=1}^T w_t \langle g_t, z_t - x_\star\rangle]}{\sum_{i=1}^T w_i}.$$

<-- this is the `regret`

To recover Theorem 1 from the above result, notice that the algorithm analyzed by Theorem 1 is captured by Theorem 2 with $w_t = 1$, β_t a constant β and $z_{t+1} = z_t - \gamma g_t$ for all t. Next, observe that the sequence $z_1, \dots, z_T$ is performing online gradient descent (Zinkevich, 2003), for which it is well-known that the *regret* $\sum_{t=1}^T \langle g_t, z_t - x_\star\rangle$ (appearing in the numerator of our result) is bounded by $DG\sqrt{T}$ and so the result of Theorem 1 immediately follows.

The regret is the principle object of study in online convex optimization (Hazan, 2022; Orabona, 2019). Viewed in this light, Theorem 2 provides a way to convert an online convex optimization algorithm into a stochastic optimization algorithm: it is a form of *online-to-batch conversion* (Cesa-Bianchi et al., 2004). Classical online-to-batch conversions are a standard technique for obtaining convergence bounds for many stochastic optimization algorithms, including stochastic gradient descent (Zinkevich, 2003), AdaGrad (Duchi et al., 2011), AMSGrad (Reddi et al., 2018), and Adam (Kingma and Ba, 2014). All of these algorithms can be analyzed as online convex optimization algorithms: they provide bounds on the regret $\sum_{t=1}^T \langle g_t, z_t - x_\star\rangle$ rather than direct convergence guarantees. It is then necessary (although sometimes left unstated) to convert these regret bounds into stochastic convergence guarantees via an online-to-batch conversion. Our result provides a more versatile method for effecting this conversion.

Theorem 2 actually provides a "grand unification" of a number of different online-to-batch conversions that have been proposed over the years. Most of these conversion methods were first developed specifically to provide convergence analysis for SGD (or some variant such as dual averaging or mirror descent), and then generalized into techniques that apply to any online convex optimization algorithm. For example, the classical Polyak averaging method can be generalized to form the "standard" online-to-batch conversion of Cesa-Bianchi et al. (2004), and is immediately recovered from Theorem 2 by setting $w_t = 1$ and $\beta_t = 0$ for all t. More recently Nesterov and Shikhman (2015); Tao et al. (2018) derived an alternative to Polyak averaging that was later generalized to work with arbitrarily online convex optimization algorithms by Cutkosky (2019); Kavis et al. (2019), and then observed to actually be equivalent to the heavy-ball momentum by Defazio (2020); Defazio and Gower (2021); Defazio and Jelassi (2022). This method is recovered by our Theorem 2 by setting $w_t = 1$ and $\beta_t = 1$ for all t. Finally, very recently Zamani and Glineur (2023) discovered that gradient descent with a *linear decay* stepsize provides a last-iterate convergence guarantee, which was again generalized to an online-to-batch conversion by Defazio et al. (2023). This final result is *also* recovered by Theorem 2 by setting $w_t = 1$ and $\beta_t = \frac{t}{T}$ (see Appendix B).

In Appendix C, we give a further tightening of Theorem 2 – it can be improved to an equality by precisely tracking additional terms that appear on the right-hand-side. This tightened version can be used to show convergence rate results for smooth losses, both with and without strong-convexity. As an example application, we show that schedule-free optimistic-gradient methods (Rakhlin and

Sridharan, 2013) converge with accelerated rates:

$$\mathbb{E}[F(x_T) - F(x_\star)] = O\left(\frac{D^2 L}{T^2} + \frac{D\sigma}{\sqrt{T}}\right).$$

2.2 On Large Learning Rates

Under classical worst-case convergence theory, the optimal choice of γ for a fixed duration training time T is $\gamma = D/(G\sqrt{T})$. This is the rate used in our bounds for Theorem 1 above. For *any-time* convergence (i.e. when stopping is allowed at any timestep), our proposed method can, in theory, be used with the standard learning rate sequence:

$$\gamma_t = \frac{D}{G\sqrt{t}}.$$

Is this theory perfect? It makes sense on the first glance but because it's empirically wrong, there must be more to learning rates than Theorem 1. Wonder if they're reading into the tea leaves here

However, learning rate sequences of this form have poor practical performance (Defazio et al., 2023). Instead, much larger steps of the form D/G give far better performance across virtually all problems in applications — another theory-practice mismatch that is virtually undiscussed in the literature. Existing theory suggests that this step-size is too large to give $\mathcal{O}(1/\sqrt{T})$ convergence, however, as we show below, there is a important special case where such large step sizes also give optimal rates up to constant factors.

Theorem 3. *Consider the online learning setting with bounded gradients g_t. Let $z_{t+1} = z_t - \gamma g_t$. Let $D = \|z_1 - z_*\|$ for arbitrary reference point z_* and define $G = \max_{t \leq T} \|g_t\|$. Suppose that the chosen step-size is $\gamma = D/G$, then if it holds that:*

$$\sum_{t=1}^{T} \langle g_t, z_t - z_1 \rangle \leq D \sqrt{\sum_{t=1}^{T} \|g_t\|^2}, \tag{1}$$

Bounds the regret

then:

$$\frac{1}{T}\sum_{t=1}^{T} \langle g_t, z_t - z_* \rangle = \mathcal{O}\left(\frac{D}{T}\sqrt{\sum_{t=1}^{T} \|g_t\|^2}\right).$$

Mmmm kinda losing me here. "ok there's a bound for regret that may answer why theory breaks empirically, but we have no proof for it, but we're using it as our regret bound"?????

This regret bound for SGD implies a convergence rate bound for Schedule-Free SGD by application of our online-to-batch conversion. Condition 11 involves known quantities and so can be checked during a training run (Using reference point $z_* = x_T$, and so $D = \|x_1 - x_T\|$), and we find that it holds for *every* problem we consider in our experiments in Section 4.1. More generally, the full conditions under which large learning rates can be used are not yet fully understood for stochastic problems. In the quadratic case, Bach and Moulines (2013) established that large fixed step-sizes give optimal convergence rates, and we conjecture that the success of large learning rates may be attributed to asymptotic quadratic behavior of the learning process.

I think condition is the condition above? I can't tell - not included in paper

Empirically, we find that Schedule-Free momentum enables the use of larger learning rates $\gamma > 0$ even in quadratic minimization problems $f(x) = \frac{1}{2}x^\top A x - b^\top x$. We generate 10 different such 20-dimensional problems with eigenvalues drawn log-uniformly in $[10^{-6}, 1]$. We plot the average minimal loss achieved as a function of the two parameters β and γ in Figure 3. We can see that when the learning rate we use is small, what value of β we choose has little to no effect on the convergence of the algorithm. However, when γ is large, choosing $\beta < 1$ becomes crucial to achieving convergence.

3 Related Work

The proposed method has a striking resemblance to Nesterov's accelerated method (Nesterov, 1983, 2013) for L-smooth functions, which can be written in the AC-SA form (Lan, 2012):

$$\begin{aligned} y_t &= (1 - c_{t+1})x_t + c_{t+1} z_t \\ z_{t+1} &= z_t - \frac{k+1}{2L}\nabla f(y_t) \\ x_{t+1} &= (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1}, \end{aligned}$$

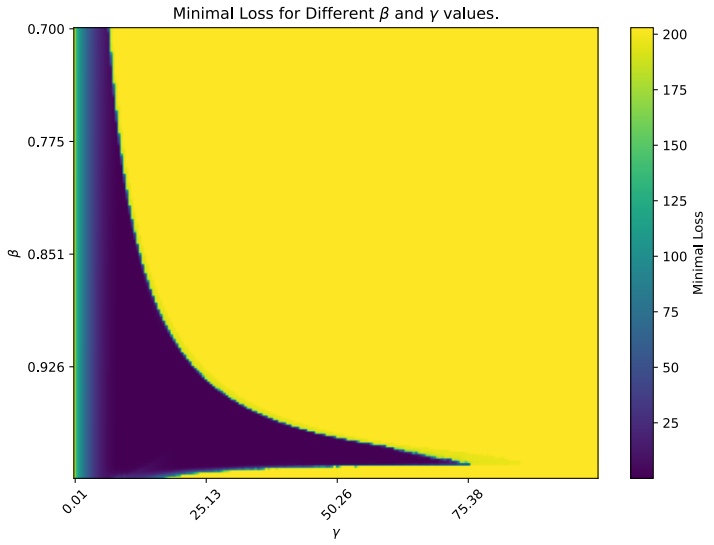

Figure 3: Incorporating the momentum parameter β allows for convergence despite using larger learning rates γ on quadratic problems. Dark region indicates convergence.

where $c_{t+1} = 2/(t+2)$. The averaging constant, and more generally

$$c_{t+1} = \frac{r+1}{t+r+1}, \tag{2}$$

for any real $r > -1$ is equivalent to the weighted average (Shamir and Zhang, 2013; Defazio and Gower, 2021) $x_t \propto \sum_{t=1}^{T} t^{\bar{r}} z_t$, where $t^{\bar{r}}$ represents the rth factorial power of t. Our framework is compatible with factorial power averages without sacrificing theoretical guarantees.

Our approach differs from conventional accelerated methods by using a different weight for the y_t and x_t interpolations. We use a constant weight for y_t and a decreasing weight for x_t. Accelerated methods for strongly-convex problems use a constant weight for both, and those for non-strongly convex use an decreasing weight for both, so our approach doesn't directly correspond to either class of accelerated method. Accelerated methods also use a much larger step size for the z_t sequence than our approach.

The use of equal-weighted averages is less common than the use of exponential weighting in the practical deep learning optimization literature. Exponential moving averages (EMA) of the iterate sequence are used in the popular Lookahead optimizer (Zhang et al., 2019). In the case of SGD, it performs $i = 1 \dots k$ inner steps:

$$z_{t,i} = z_{t,i-1} - \gamma \nabla f(z_{t,i-1})$$

followed by an outer step:

$$x_t = x_{t-1} + \alpha \left(z_{t,k} - x_{t-1} \right).$$

The inner optimizer then starts at $z_{t+1,0} = x_{t-1}$. The Lookahead method can be seen as the EMA version of primal averaging, just as exponential weight averaging is the EMA version of Polyak-Ruppert averaging.

Tail averaging, either using an exponential moving average or an equal-weighted average, is a common 'folk-law' technique that often yields a practical improvement. For instance, this kind of averaging is used without citation by the influential work of Szegedy et al. (2016): "Model evaluations are performed using a running average of the parameters computed over time.", and by Vaswani et al. (2017): "...averaged the last 20 checkpoints". Tail averages are typically "Polyak-Ruppert" style averaging as the average is not used for gradient evaluations during training.

This tail averaging was used in the original transformer paper

More sophisticated tail averaging approaches such as Stochastic Weight Averaging (Izmailov et al., 2018) and LAtest Weight Averaging (Kaddour, 2022; Sanyal et al., 2023) combine averaging with large or cyclic learning rates. They are not a replacement for scheduling, instead they aim to improve

final test metrics. They generally introduce additional hyper-parameters to tune, and require additional memory. It is possible to use SWA and LAWA on top of our approach, potentially giving further gains.

Within optimization theory, tail averages can be used to improve the convergence rate for stochastic non-smooth SGD in the strongly convex setting from $\mathcal{O}(\log(T)/T)$ to $\mathcal{O}(1/T)$(Rakhlin et al., 2012), although at the expense of worse constants compared to using weighted averages of the whole sequence (Lacoste-Julien et al., 2012).

cosine schedules

Portes et al. (2022) use cyclic learning rate schedules with increasing cycle periods to give a method that explores multiple points along the Pareto frontier of training time vs eval performance. Each point at the end of a cycle is an approximation to the model from a tuned schedule ending at that time. Our method gives the entire frontier, rather than just a few points along the path. In addition, our method matches or improves upon best known schedules, whereas the "... cyclic trade-off curve underestimated the standard trade-off curve by a margin of 0.5% validation accuracy" (Portes et al., 2022).

4 Experiments

To validate the effectiveness of our method, we performed a large-scale comparison across multiple domains (computer vision, language, and categorical data) and covering a range of small scale to large-scale experiments (logistic regression to large language model training). Details of the implementation of our method for SGD and Adam used in the experiments are in Section 4.4.

4.1 Deep Learning Problems

For our deep learning experiments, we evaluated Schedule-Free learning on a set benchmark tasks that are commonly used in the optimization research literature:

CIFAR10 A Wide ResNet (16-8) architecture (Zagoruyko and Komodakis, 2016) on the CIFAR10 image classification dataset.

CIFAR100 A DenseNet (Huang et al., 2017) architecture on the CIFAR-100 (100-class) classification dataset.

SVHN A deep ResNet architecture (3-96) on the Street View House Numbers (SVHN) dataset.

ImageNet A standard ResNet-50 architecture (He et al., 2016) on the ILSVRC 2012 ImageNet (Russakovsky et al., 2015) classification dataset.

Not transformer? The original transformer was evaluated on this dataset. I think the original LSTM maybe was eval'd on it as well

IWSLT14 A LSTM architecture (Wiseman and Rush, 2016) on the IWSLT14 German-English translation dataset (Cettolo et al., 2014).

DLRM The DLRM (Naumov et al., 2019) architecture on the Criteo Kaggle Display Advertising dataset (Jean-Baptiste Tien, 2014).

MRI A stacked U-Net architecture (Sriram et al., 2020) on the fastMRI dataset (Zbontar et al., 2018).

MAE Fine-tuning a pretrained Masked Autoencoder (He et al., 2021) ViT (patch16-512d-8b) on the ILSVRC 2012 ImageNet dataset.

NanoGPT A 124M parameter GPT-2 (Radford et al., 2019) style decoder-only transformer on the OpenWebText dataset (Gokaslan and Cohen, 2019).

For each problem, both the baseline and the Schedule-Free method were tuned by sweeping both the weight decay and learning rate on a grid. We also swept β over two values, 0.9 and 0.98. Final hyper-parameters are listed in the Appendix. Schedule-Free SGD was used for CIFAR10, CIFAR100, SVHN and ImageNet, and Schedule-Free AdamW was used for the remaining tasks. We further include a step-wise schedule as a comparison on problems where step-wise schedules are customary.

Our approach shows very strong performance (Figure 4) out-performing existing state-of-the-art cosine schedules on CIFAR-10, CIFAR-100, SVHN, IWSLT-14 (Figure 2) and OpenWebText GPT-2 problems, as well as the state-of-the-art Linear Decay schedules on the fastMRI and Criteo DLRM tasks. On the remaining two problems, MAE finetuning and ImageNet ResNet-50 training, it ties with the existing best schedules.

In general, the optimal learning rates for the Schedule-Free variants were larger than the optimal values for the base optimizers. The ability to use larger learning rates without diverging may be a contributing factor to the faster convergence of Schedule-Free methods. The β parameter works well

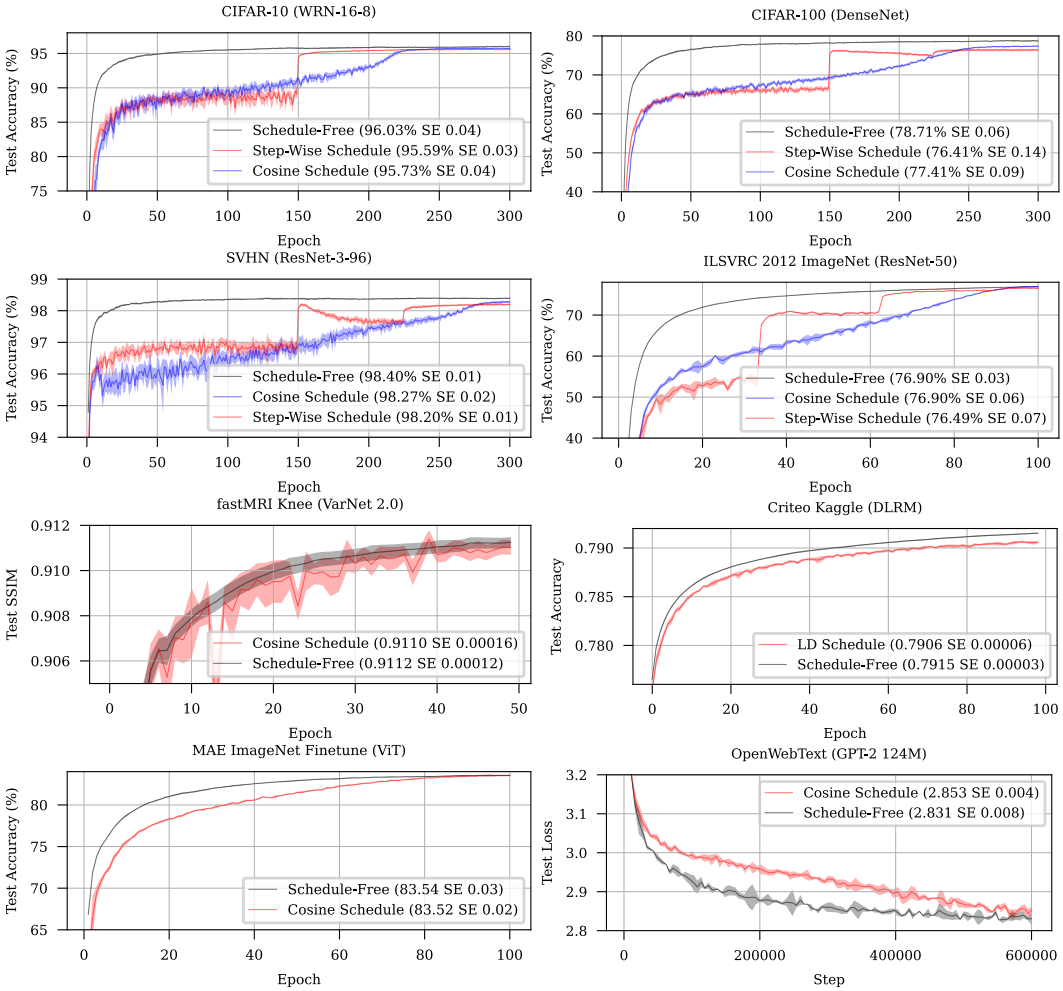

Figure 4: Deep Learning Experiments

at the default value of 0.9 for all problems except NanoGPT, where the loss started to increase rapidly when 0.9 was used. The larger $\beta = 0.98$ value in our sweep was stable.

4.2 MLCommons Algorithmic Efficiency benchmark

The AlgoPerf challenge (Dahl et al., 2023) is designed to be a large-scale and comprehensive benchmark for deep learning optimization algorithms, covering major data domains and architectures. It includes Transformers, ConvNets and U-Net models across image, language, graph and speech domains, and contains 8 problems total. We evaluated Schedule-Free AdamW following the competition guidelines, comparing against NAdamW, the competition reference Algorithm, running 10 seeds of each. As this is a time-to-target competition, traditional error bars are not appropriate so we instead plot all 10 seeds separately. Note that we excluded one benchmark problem, ResNet-50 training, as neither AdamW nor NAdamW can hit the target accuracy on that task. The remaining tasks are:

WMT A Encoder-Decoder Transformer Model on the WMT17 German-to-english translation task (Bojar et al., 2017).

VIT A S/16 Vision Transformer (Dehghani et al., 2023) model on the ILSVRC 2012 ImageNet classification task (Russakovsky et al., 2015).

FASTMRI The reference U-Net architecture from the fastMRI challenge Knee MRI dataset (Zbontar et al., 2018).

CONFORMER A Conformer (Gulati et al., 2020) Speech Recognition model on the LibriSpeech ASR dataset (Panayotov et al., 2015).

Algorithm 1 Schedule-Free AdamW

Input: x_1, learning rate γ, decay λ, warmup steps T_{warmup}, $\beta_1, \beta_2, \epsilon$
$z_1 = x_1$
$v_0 = 0$
for $t = 1$ **to** T **do**
  $y_t = (1 - \beta_1)z_t + \beta_1 x_t$
  $g_t \in \partial f(y_t, \zeta_t)$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\hat{v}_t = v_t / (1 - \beta_2^t)$
  $\gamma_t = \gamma \min(1, t/T_{\text{warmup}})$
  $z_{t+1} = z_t - \gamma_t g_t / (\sqrt{\hat{v}_t} + \epsilon) - \gamma_t \lambda y_t$
  $c_{t+1} = \frac{\gamma_t^2}{\sum_{i=1}^{t} \gamma_i^2}$
  $x_{t+1} = (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1}$
end for
Return x_{T+1}

OGBG A Graph-Neural Network in the style of Battaglia et al. (2018) on a Molecular property prediction task from the Open Graph Benchmark (Hu et al., 2020) suite (PubChem BioAssay data).

CRITEO Clickthrough-rate prediction on the criteo 1B dataset (Criteo, 2022) using the Deep Learning Recommendation Model (DLRM) architecture.

DEEPSPEECH The Deep Speech model on the LibriSpeech ASR dataset.

The self-tuning track restricts participants to provide a single set of hyper-parameters to use for all 8 problems. Given the large number of problems, this gives performance representative of a good default configuration.

Schedule-Free AdamW performs well across all considered tasks, out-performing the baseline on the WMT, VIT, FASTMRI and OGBG training, while tying on the Conformer and Criteo workloads, and marginally under-performing on the DeepSpeech workload. We attribute the performance on the Conformer and DeepSpeech tasks to their use of batch-norm - the AlgoPerf setup doesn't easily allow us to update the BN running statistics on the x sequence, which is necessary with our method to get the best performance (See Section 4.4).

4.3 Convex Problems

We validated the Schedule-Free learning approach on a set of standard logistic regression problems from the LibSVM repository. For each problem, and each method separately, we performed a full learning rate sweep on a power-of-two grid, and plotted mean and standard-error of the final train accuracy from 10 seeds using the best learning rate found.

Schedule-Free learning out-performs both averaging approaches and the state-of-the-art linear decay (LD) schedule baseline (Figure 6). It converges faster on all but 1 of 12 problems, has higher accuracy on 6 of the problems, and ties the baseline on the remaining problems. This demonstrates that the performance advantages of Schedule-Free methods are not limited to non-convex problems.

4.4 Implementation Concerns

The Schedule-Free variant of a method typically has the same memory requirements as the base method. For instance, Schedule-Free SGD requires no extra memory over standard SGD with momentum. Whereas SGDM tracks the current point x and the momentum buffer m, we track x and z. The quantity y does not need to be stored, as it can be computed directly from the latest values of x and z. This is the case for AdamW also, see Algorithm 1.

Our method requires extra code to handle models where batch norm is used. This is due to the fact that BatchNorm layers maintain a running_mean and running_var to track batch statistics which is calculated at y. For model evaluation, these buffers need to be updated to match the statistics on the x sequence. This can be done by evaluating a small number of training batches using x right before

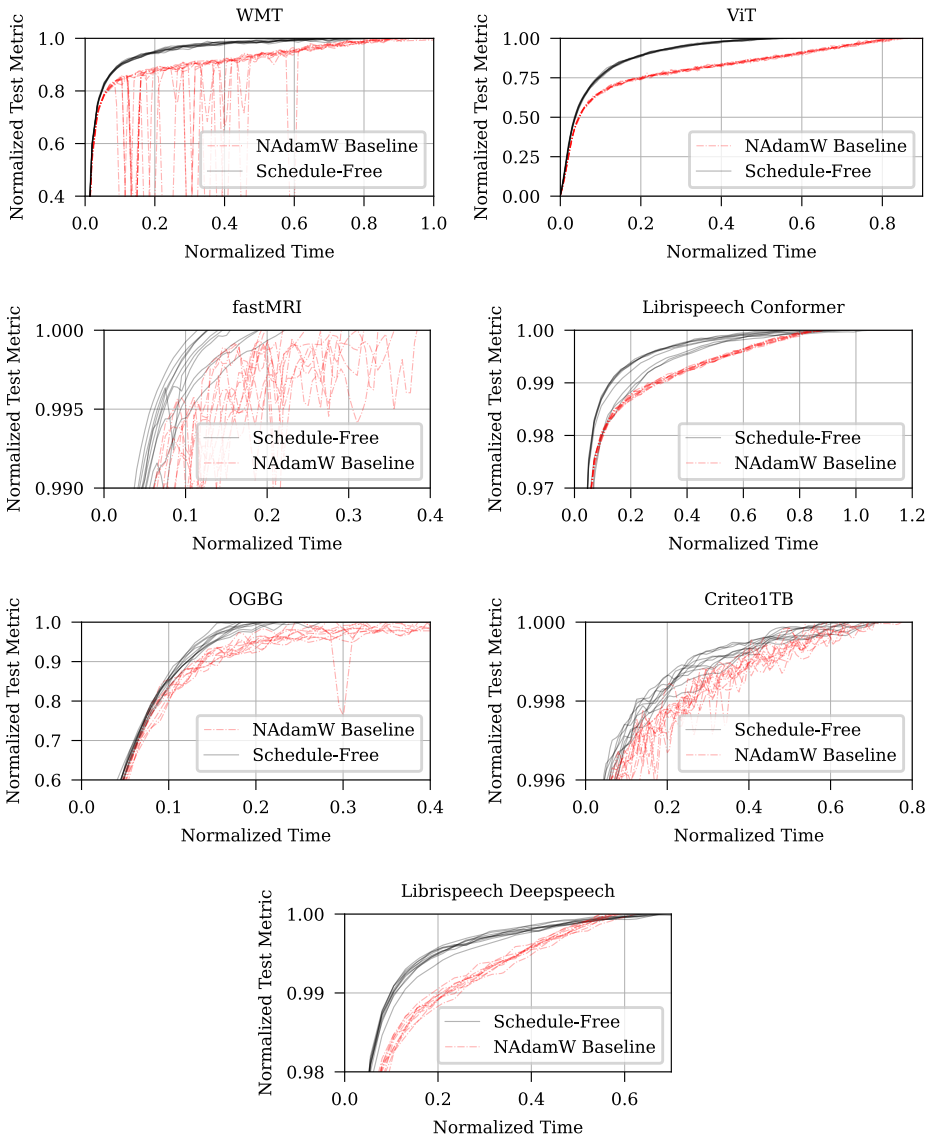

Figure 5: Schedule-Free Adam compared to target-setting baseline on the Algoperf competition self-tuning track.

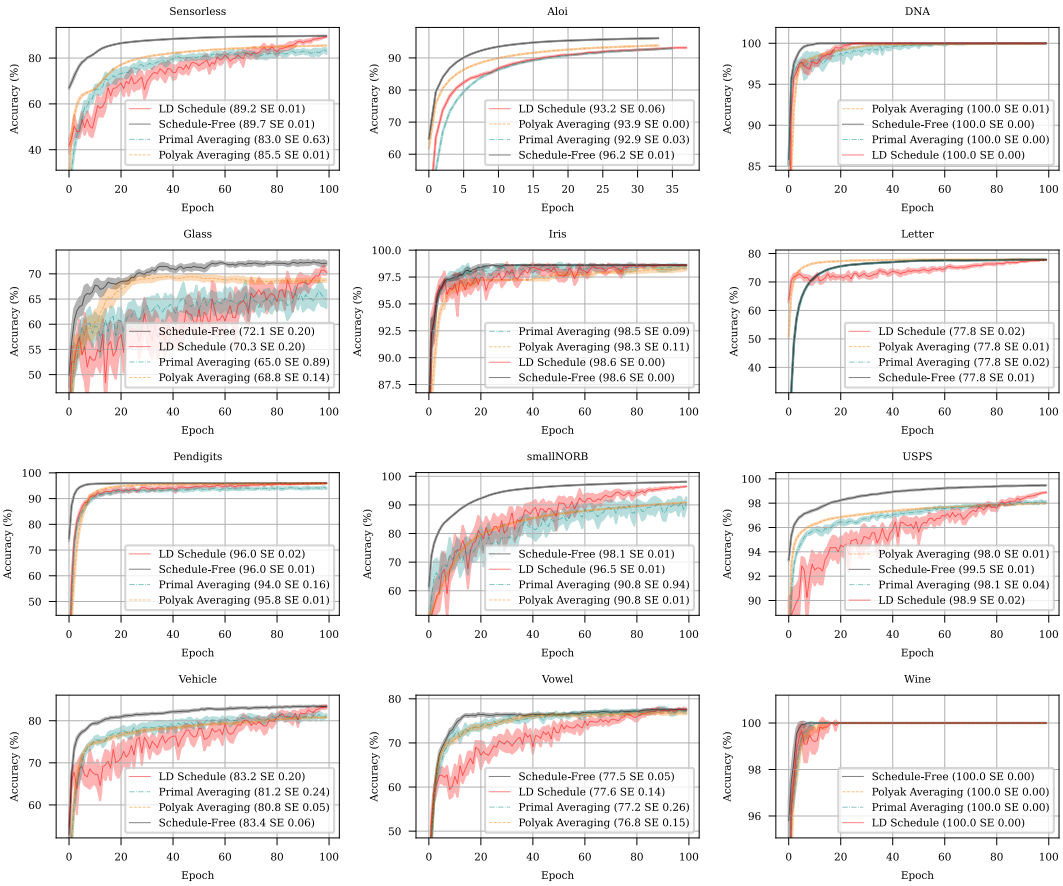

Figure 6: Stochastic logistic regression experiments.

each eval. More sophisticated approaches such as PreciseBN (Wu and Johnson, 2021) can also be used.

Learning rate warmup is still necessary for our method. We found that performance was greatly improved by using a weighted c_t sequence when warmup is used, weighted by the square of the LR $\gamma_t = \gamma \min(1, t/T_{\text{warmup}})$ used during warmup:

$$c_{t+1} = \frac{\gamma_t^2}{\sum_{i=1}^{t} \gamma_i^2} \tag{3}$$

This sequence decreases at $1/t$ after the learning rate warmup, and is shifted by one from the indexing used in Theorem 2. This sequence is motivated by Theorem 2's weighting sequences, which suggest weights proportional to polynomials of the learning rate.

Weight decay for Schedule-Free methods can be computed at either the y or z sequences. We used decay at y for our experiments, as this matches the interpretation of weight-decay as the use of an additional L2-regularizer term in the loss.

5 Conclusion

We have presented Schedule-Free learning, an optimization approach that removes the need to specify a learning rate schedule while matching or outperforming schedule-based learning. Our method has no notable memory, computation or performance limitations compared to scheduling approaches and we show via large-scale experiments that it is a viable drop-in replacement for schedules. The primary practical limitation is the need to sweep learning rate and weight decay, as the best values differ from the those used with a schedule. We provide a preliminary theoretical exploration of the method, but further theory is needed to fully understand the method.

6 Contributions

Aaron Defazio discovered the method, led research experimentation and proved initial versions of Theorems 1 and 7, with experimental/theoretical contributions by Alice Yang. Alice Yang led the development of the research codebase. Ashok Cutkosky proved key results including Theorem 2 and led the theoretical investigation of the method. Ahmed Khaled developed preliminary theory for obtaining accelerated rates which was later supplanted by Theorem 2, and investigated the utility of β with large learning rates for quadratics. Additional derivations by Konstantin Mishchenko and Harsh Mehta are included in appendix sections. Discussions between Aaron Defazio, Ashok Cutkosky, Konstantin Mishchenko, Harsh Mehta, and Ahmed Khaled over the last year contributed to this scientific discovery.

References

Bach, F. and Moulines, E. (2013). Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the 2017 Conference on Machine Translation (WMT17)*.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. (2014). Report on the 11th IWSLT evaluation campaign. In *IWSLT*.

Chiang, C.-K., Yang, T., Lee, C.-J., Mahdavi, M., Lu, C.-J., Jin, R., and Zhu, S. (2012). Online optimization with gradual variations. In *Conference on Learning Theory*, pages 6–1. JMLR Workshop and Conference Proceedings.

Criteo (2022). Criteo 1TB click logs dataset. https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/.

Cutkosky, A. (2019). Anytime online-to-batch, optimism and acceleration. In *International conference on machine learning*, pages 1446–1454. PMLR.

Dahl, G. E., Schneider, F., Nado, Z., Agarwal, N., Sastry, C. S., Hennig, P., Medapati, S., Eschenhagen, R., Kasimbeg, P., Suo, D., Bae, J., Gilmer, J., Peirson, A. L., Khan, B., Anil, R., Rabbat, M., Krishnan, S., Snider, D., Amid, E., Chen, K., Maddison, C. J., Vasudev, R., Badura, M., Garg, A., and Mattson, P. (2023). Benchmarking Neural Network Training Algorithms.

Defazio, A. (2020). Momentum via primal averaging: Theoretical insights and learning rate schedules for non-convex optimization.

Defazio, A., Cutkosky, A., Mehta, H., and Mishchenko, K. (2023). When, why and how much? adaptive learning rate scheduling by refinement.

Defazio, A. and Gower, R. M. (2021). The power of factorial powers: New parameter settings for (stochastic) optimization. In Balasubramanian, V. N. and Tsang, I., editors, *Proceedings of The 13th Asian Conference on Machine Learning*, volume 157 of *Proceedings of Machine Learning Research*, pages 49–64. PMLR.

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

Defazio, A. and Mishchenko, K. (2023). Learning-rate-free learning by D-adaptation. *The 40th International Conference on Machine Learning (ICML 2023)*.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme Ruiz, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., Steenkiste, S. V., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M., Gritsenko, A. A., Birodkar, V., Vasconcelos, C. N., Tay, Y., Mensink, T., Kolesnikov, A., Pavetic, F., Tran, D., Kipf, T., Lucic, M., Zhai, X., Keysers, D., Harmsen, J. J., and Houlsby, N. (2023). Scaling vision transformers to 22 billion parameters. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7480–7512. PMLR.

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

Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition.

Hazan, E. (2022). *Introduction to online convex optimization*. MIT Press.

Hazan, E. and Kale, S. (2010). Extracting certainty from uncertainty: Regret bounded by variation in costs. *Machine learning*, 80:165–188.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners. *arXiv:2111.06377*.

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

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: datasets for machine learning on graphs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Jean-Baptiste Tien, joycenv, O. C. (2014). Display advertising challenge.

Joulani, P., György, A., and Szepesvári, C. (2017). A modular analysis of adaptive (non-) convex optimization: Optimism, composite objectives, and variational bounds. In *International Conference on Algorithmic Learning Theory*, pages 681–720. PMLR.

Joulani, P., Raj, A., Gyorgy, A., and Szepesvári, C. (2020). A simpler approach to accelerated optimization: iterative averaging meets optimism. In *International conference on machine learning*, pages 4984–4993. PMLR.

Kaddour, J. (2022). Stop wasting my time! saving days of ImageNet and BERT training with latest weight averaging.

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

Kingma, D. P. and Ba, J. (2014). Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Lacoste-Julien, S., Schmidt, M., and Bach, F. (2012). A simpler approach to obtaining an $o(1/t)$ convergence rate for the projected stochastic subgradient method.

Lan, G. (2012). An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397.

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. (2019). Deep learning recommendation model for personalization and recommendation systems. *CoRR*.

Nesterov, Y. (1983). A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*.

Nesterov, Y. (2013). *Lectures on Convex Optimization*. Springer Nature.

Nesterov, Y. and Shikhman, V. (2015). Quasi-monotone subgradient methods for nonsmooth convex minimization. *Journal of Optimization Theory and Applications*, 165(3):917–940.

Orabona, F. (2019). A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Polyak, B. (1990). New stochastic approximation type procedures. *Avtomatica i Telemekhanika*, 7:98–107.

Portes, J., Blalock, D., Stephenson, C., and Frankle, J. (2022). Fast benchmarking of accuracy vs. training time with cyclic learning rates.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Technical report, OpenAI.

Rakhlin, A., Shamir, O., and Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*.

Rakhlin, A. and Sridharan, K. (2013). Online learning with predictable sequences. In *Conference on Learning Theory*, pages 993–1019. PMLR.

Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of Adam and beyond. In *International Conference on Learning Representations*.

Ruppert, D. (1988). Efficient estimations from a slowly convergent Robbins-Monro process. *Technical Report, Cornell University*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3).

Sanyal, S., Neerkaje, A., Kaddour, J., Kumar, A., and Sanghavi, S. (2023). Early weight averaging meets high learning rates for LLM pre-training.

Sebbouh, O., Gower, R. M., and Defazio, A. (2021). On the (asymptotic) convergence of stochastic gradient descent and stochastic heavy ball. In *Conference on Learning Theory, COLT 2021*, Proceedings of Machine Learning Research. PMLR.

Shamir, O. and Zhang, T. (2013). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on Machine Learning*.

Sriram, A., Zbontar, J., Murrell, T., Defazio, A., Zitnick, C. L., Yakubova, N., Knoll, F., and Johnson, P. (2020). End-to-end variational networks for accelerated MRI reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. JMLR.org.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Tao, W., Pan, Z., Wu, G., and Tao, Q. (2018). Primal averaging: A new gradient evaluation step to attain the optimal individual convergence. *IEEE Transactions on Cybernetics*, PP:1–11.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Wu, Y. and Johnson, J. (2021). Rethinking "batch" in batchnorm.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Zamani, M. and Glineur, F. (2023). Exact convergence rate of the last iterate in subgradient methods.

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

Zhang, M., Lucas, J., Ba, J., and Hinton, G. E. (2019). Lookahead optimizer: k steps forward, 1 step back. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, pages 928–935.

A Proof of Theorem 2

Theorem 2. *Let F be a convex function. Let $\zeta_1, \dots, \zeta_T$ be an iid sequence such that $F(x) = \mathbb{E}_\zeta[f(x,\zeta)]$. Let $z_1, \dots, z_T$ be arbitrary vectors and let $w_1, \dots, w_T$ and $\beta_1, \dots, \beta_T$ be arbitrary numbers in $[0,1]$ such that z_t, w_t and β_t are independent of $\zeta_t, \dots, \zeta_T$. Set:*

$$x_t = \frac{\sum_{i=1}^t w_i z_i}{\sum_{i=1}^t w_i} = x_{t-1} \underbrace{\left(1 - \frac{w_t}{\sum_{i=1}^t w_i}\right)}_{\triangleq 1-c_t} + \underbrace{\frac{w_t}{\sum_{i=1}^t w_i}}_{\triangleq c_t} z_t$$

$$y_t = \beta_t x_t + (1-\beta_t) z_t$$

$$g_t = \nabla f(y_t, \zeta_t)$$

Then we have for all $x_\star$:

$$\mathbb{E}[F(x_T) - F(x_\star)] \leq \frac{\mathbb{E}[\sum_{t=1}^T w_t \langle g_t, z_t - x_\star\rangle]}{\sum_{i=1}^T w_i}.$$

Proof. Throughout this proof, we will use the notation $w_{1:t} = \sum_{i=1}^t w_i$. The result is established by showing the following identity:

$$w_{1:t} F(x_t) - w_{1:t-1} F(x_{t-1}) - w_t F(x_\star) \leq w_t \langle \nabla F(y_t), z_t - x_\star\rangle. \tag{4}$$

Where here $\nabla F(y_t)$ indicates a subgradient of F at y_t with $\mathbb{E}[g_t|z_t] = \nabla F(y_t)$. Given the identity (4), we sum over all t from 1 to T. Then the LHS will telescope to obtain:

$$w_{1:T}(F(x_T) - F(x_\star)) \leq \sum_{t=1}^T w_t \langle \nabla F(y_t), z_t - x_\star\rangle,$$

from which the conclusion immediately follows since $\mathbb{E}[g_t|z_t] = \nabla F(y_t)$. So, let us establish (4). To do so, it will help to observe the following identities:

$$w_t z_t = w_{1:t} x_t - w_{1:t-1} x_{t-1}$$

$$w_{1:t-1}(x_t - x_{t-1}) = w_t(z_t - x_t) \tag{5}$$

$$z_t - y_t = \frac{\beta_t}{1-\beta_t}(y_t - x_t). \tag{6}$$

Now, setting $\nabla F(x_t)$ to be an arbitrary subgradient of F at x_t, we have:

$$\begin{aligned}
&w_{1:t} F(x_t) - w_{1:t-1} F(x_{t-1}) - w_t F(x_\star) \\
&= w_{1:t-1}(F(x_t) - F(x_{t-1})) + w_t(F(x_t) - F(x_\star)) \\
&\leq w_{1:t-1}\langle \nabla F(x_t), x_t - x_{t-1}\rangle + w_t(F(x_t) - F(x_\star))
\end{aligned}$$

using (5):

$$\begin{aligned}
&= w_t \langle \nabla F(x_t), z_t - x_t\rangle + w_t(F(x_t) - F(x_\star)) \\
&= w_t \langle \nabla F(x_t), z_t - x_t\rangle + w_t(F(x_t) - F(y_t)) + w_t(F(y_t) - F(x_\star)) \\
&\leq w_t \langle \nabla F(x_t), z_t - x_t\rangle + w_t\langle \nabla F(x_t), x_t - y_t\rangle + w_t\langle \nabla F(y_t), y_t - x_\star\rangle \\
&= w_t \langle \nabla F(x_t) - \nabla F(y_t), z_t - y_t\rangle + w_t\langle \nabla F(y_t), z_t - x_\star\rangle
\end{aligned}$$

using (6):

$$= w_t \frac{\beta_t}{1-\beta_t}\langle \nabla F(x_t) - \nabla F(y_t), y_t - x_t\rangle + w_t\langle \nabla F(y_t), z_t - x_\star\rangle$$

Finally, recall that any convex function satisfies $\langle \nabla F(b) - \nabla F(a), a - b\rangle \leq 0$ for all a, b. This classical fact can be established by adding the following two subgradient identities:

$$F(a) \geq F(b) + \langle \nabla F(b), a - b\rangle,$$
$$F(b) \geq F(a) + \langle \nabla F(a), b - a\rangle.$$

Then, since $\beta_t \in [0,1]$, we have $w_t \frac{\beta_t}{1-\beta_t}\langle \nabla F(x_t) - \nabla F(y_t), y_t - x_t\rangle \leq 0$, which establishes the desired identity (4). □

B Recovering Prior Conversions, and Connections to Momentum

The following recursions provide an equivalent update to our main algorithm that casts the update in a more "momentum-like" form.

Theorem 4. *Under the same assumptions and notation as Theorem 2, set:*

$$\begin{aligned}\Delta_t &= z_{t+1} - z_t,\\ m_t &= x_{t+1} - x_t,\\ u_t &= y_{t+1} - y_t.\end{aligned}$$

Then:

$$\begin{aligned}m_t &= \frac{w_{t+1}w_{1:t-1}}{w_t w_{1:t+1}} m_{t-1} + \frac{w_{t+1}}{w_{1:t+1}}\Delta_t\\ u_t &= \left(\beta_t + (\beta_t - \beta_{t+1})\frac{w_{1:t}}{w_{t+1}}\right) m_t + (1-\beta_t)\Delta_t\end{aligned}$$

Here u_t is playing the role of the "update vector", as the sequence of points y_t are where we will be evaluating gradients. The Δ_t value can be interpreted as a "base update" value: for the case that the z_t sequence is specified by SGD (as in Theorem 1), $\Delta_t = -\eta g_t$. Thus, the update can be interpreted as a momentum term m_t, plus an extra "push" in the direction of Δ_t scaled by $1-\beta_t$.

Proof. Let's solve for m_t in terms of previous values:

$$\begin{aligned}m_t &= x_{t+1} - x_t\\ &= \frac{w_{t+1}}{w_{1:t+1}}(z_{t+1} - x_t)\\ &= \frac{w_{t+1}}{w_{1:t+1}}(\Delta_t + z_t - x_t)\\ &= \frac{w_{t+1}}{w_{1:t+1}}(\Delta_t + \frac{w_{1:t-1}}{w_t}(x_t - x_{t-1}))\\ &= \frac{w_{t+1}w_{1:t-1}}{w_t w_{1:t+1}} m_{t-1} + \frac{w_{t+1}}{w_{1:t+1}}\Delta_t.\end{aligned}$$

Now let's solve for u_t:

$$\begin{aligned}u_t &= \beta_{t+1}x_{t+1} + (1-\beta_{t+1})z_{t+1} - \beta_t x_t - (1-\beta_t)z_t\\ &= \beta_t m_t + (1-\beta_t)\Delta_t + (\beta_t - \beta_{t+1})(z_{t+1} - x_{t+1})\\ &= \beta_t m_t + (1-\beta_t)\Delta_t + (\beta_t - \beta_{t+1})\frac{w_{1:t}}{w_{t+1}}(x_{t+1} - x_t)\\ &= \beta_t m_t + (1-\beta_t)\Delta_t + (\beta_t - \beta_{t+1})\frac{w_{1:t}}{w_{t+1}} m_t\\ &= \left(\beta_t + (\beta_t - \beta_{t+1})\frac{w_{1:t}}{w_{t+1}}\right) m_t + (1-\beta_t)\Delta_t\end{aligned}$$

□

In the special case that $w_t = 1$ for all t, the updates simplify to:

$$\begin{aligned}m_t &= \frac{t-1}{t+1} m_{t-1} + \frac{1}{t+1}\Delta_t\\ u_t &= (\beta_t + t(\beta_t - \beta_{t+1}))\, m_t + (1-\beta_t)\Delta_t.\end{aligned}$$

In the special case that $\beta_t = \beta$ for all t, the update for u_t simplifies to:

$$u_t = \beta m_t + (1-\beta)\Delta_t.$$

From this, it is clear that if $\beta = 1$ and $w_t = 1$, then we recover the standard Polyak momentum with a time-varying momentum factor $m_t = \frac{t-1}{t+1} m_{t-1} + \frac{1}{t+1}\Delta_t$, while it $\beta = 0$, then we have ordinary SGD without momentum.

B.1 Recovering Linear Decay

Let's take a look at the update for $u_t = y_{t+1} - y_t$ in the special case that $w_t = 1$ for all t:

$$u_t = (\beta_t + t(\beta_t - \beta_{t+1}))\, m_t + (1 - \beta_t)\Delta_t.$$

Let us define $\alpha_t = 1 - \beta_t$. Then we can re-write this update as:

$$u_t = (\alpha_t + t(\alpha_{t+1} - \alpha_t))\, m_t + \alpha_t \Delta_t.$$

It looks like we might be able to set α_t such that the coefficient of m_t vanishes. In this case, α_t would play the role of a "schedule" as the update would just be $u_t = \alpha_t \Delta_t$. Solving the recursion we get:

$$\begin{aligned} \alpha_t - 1 &= t(\alpha_{t+1} - \alpha_t), \\ \alpha_{t+1} &= \frac{(t+1)\alpha_t - 1}{t}. \end{aligned}$$

Amazingly, this recursion is satisfied by $\alpha_t = \frac{T-t}{T}$, which is the linear decay schedule! Notably, this schedule has $\alpha_T = 0$, which in turn implies that $y_T = x_T$, so that the last iterate of our algorithm is x_T, for which Theorem 2 provides a convergence guarantee.

The recursion is also satisfied by $\alpha_t = 1$ for all t (which recovers standard Polyak-Ruppert averaging). Notably, this recursion shows that α_1 will determine all subsequent α values. The values will decease linearly to zero, and then they will try to go negative, which is not allowed. So the linear decay schedule is the value of α_1 that is "just barely" allowed since it hits zero at α_T.

In general with arbitrary w_t, the recursion is:

$$1 - \alpha_t + (\alpha_{t+1} - \alpha_t)\frac{w_{1:t}}{w_{t+1}} = 0.$$

If we insist that $\alpha_T = 0$ (so that $y_T = x_T$ and we get a "last iterate" guarantee), then solving the recursion yields:

$$\alpha_t = \frac{w_{t+1:T}}{w_{1:T}},$$

which exactly recovers the main result of Defazio et al. (2023).

C Generalizing Theorem 2 via Bregman Divergences

Here, we provide a generalized version of Theorem 2 in the style of Joulani et al. (2020). This result employs Bregman divergences to tighten the inequality of Theorem 2 to an equality.

Theorem 5. *Let F be a convex function. Let $\zeta_1, \dots, \zeta_T$ be a sequence of i.i.d. random variables, and let g be a function such that $\mathbb{E}[g(x, \zeta_t)] \in \partial F(x)$ for all x and t. Let $z_1, \dots, z_T$ be arbitrary vectors and let $w_1, \dots, w_T$ and $\alpha_1, \dots, \alpha_T$ be arbitrary non-negative real numbers with $\alpha_t \leq 1$ such that z_t, w_t and α_t are independent of $\zeta_t, \dots, \zeta_T$. Define the Bregman divergence of F as $B_F(a, b) = F(a) - F(b) - \langle \nabla F(b), a - b \rangle$[2]. Set:*

$$\begin{aligned} x_t &= \frac{\sum_{i=1}^t w_i z_i}{\sum_{i=1}^t w_i} = x_{t-1}\left(1 - \frac{w_t}{\sum_{i=1}^t w_i}\right) + \frac{w_i}{\sum_{i=1}^t w_i} z_t \\ y_t &= (1 - \alpha_t)x_t + \alpha_t z_t \\ g_t &= g(y_t, \zeta_t). \end{aligned}$$

Define the "compressed sum" notation: $w_{1:t} = \sum_{i=1}^t w_i$, with $w_{1:0} = 0$.

[2] if F is not differentiable, then by abuse of notation define $\nabla F(b) = \mathbb{E}[g(b, \zeta)]$, which is a particular choice of subgradient of F.

Then we have for all $x_\star$:

$$\begin{aligned}\mathbb{E}[F(x_T) - F(x_\star)] = \mathbb{E}\left[\frac{\sum_{t=1}^T w_t \langle g_t, z_t - x_\star\rangle}{w_{1:T}}\right] \\ - \mathbb{E}\left[\frac{\sum_{t=1}^T \frac{w_t}{\alpha_t} B_F(y_t, x_t) + \frac{w_t(1-\alpha_t)}{\alpha_t} B_F(x_t, y_t)}{w_{1:T}}\right] \\ - \mathbb{E}\left[\frac{\sum_{t=1}^T w_{1:t-1} B_F(x_{t-1}, x_t) + w_t B_F(x_\star, y_t)}{w_{1:T}}\right].\end{aligned}$$

Let's take a minute to unpack this result since it is depressingly complicated. Recall that the Bregman divergence for a convex function must be positive, and so all the subtracted Bregman divergence terms can be dropped to make the bound only looser. This recovers Theorem 2. However, in Section D, we show how to exploit the negative Bregman terms to achieve accelerated rates when F is smooth, and in Section E we show how to exploit the negative Bregman terms to achieve faster rates when F is strongly-convex.

Proof. The proof is nearly the same as that of Theorem 2. The only difference is that we keep track of all the error terms in the inequalities via Bregman divergences.

Throughout this proof, we use $\nabla F(x)$ to indicate $\mathbb{E}_\zeta[g(x,\zeta)]$. When F is differentiable, this is simply the ordinary gradient at x. When F is non-differentiable, this reprents a specific choice of subgradient at x.

Recall that any convex function satisfies $\langle \nabla F(b) - \nabla F(a), a - b\rangle = -B_F(a,b) - B_F(b,a)$ for all a, b. This classical fact can be established by adding the following two subgradient identities:

$$\begin{aligned}F(a) &= F(b) + \langle \nabla F(b), a - b\rangle + B_F(a,b) \\ F(b) &= F(a) + \langle \nabla F(a), b - a\rangle + B_F(b,a) \\ \langle \nabla F(b) - \nabla F(a), a - b\rangle &= -B_F(a,b) - B_F(b,a). \end{aligned} \tag{7}$$

The Theorem is established by showing the following identity:

$$\begin{aligned}w_{1:t}F(x_t) - w_{1:t-1}F(x_{t-1}) - w_t F(x_\star) = w_t\langle \nabla F(y_t), z_t - x_\star\rangle \\ - \frac{w_t}{\alpha_t} B_F(y_t, x_t) - \frac{w_t(1-\alpha_t)}{\alpha_t} B_F(x_t, y_t) \\ - w_{1:t-1} B_F(x_{t-1}, x_t) - w_t B_F(x_\star, y_t).\end{aligned} \tag{8}$$

Given the identity (8), we sum over all t from 1 to T. Then the LHS will telescope to obtain:

$$\begin{aligned}w_{1:T}(F(x_T) - F(x_\star)) = \sum_{t=1}^T w_t\langle \nabla F(y_t), z_t - x_\star\rangle \\ - \sum_{t=1}^T \frac{w_t}{\alpha_t} B_F(y_t, x_t) - \frac{w_t(1-\alpha_t)}{\alpha_t} B_F(x_t, y_t) \\ - \sum_{t=1}^T w_{1:t-1} B_F(x_{t-1}, x_t) - w_t B_F(x_\star, y_t),\end{aligned}$$

from which the conclusion immediately follows since $\mathbb{E}[g_t | g_1, \dots, g_{t-1}] = \mathbb{E}[\nabla F(y_t) | g_1, \dots, g_{t-1}]$. So, let us establish (4). To do so, it will help to observe the following identities:

$$\begin{aligned}w_t z_t &= w_{1:t}x_t - w_{1:t-1}x_{t-1} \\ w_{1:t-1}(x_t - x_{t-1}) &= w_t(z_t - x_t) \qquad (9) \\ z_t - y_t &= \frac{1-\alpha_t}{\alpha_t}(y_t - x_t). \qquad (10)\end{aligned}$$

So, we have:

$$\begin{aligned}
&w_{1:t}F(x_t) - w_{1:t-1}F(x_{t-1}) - w_tF(x_\star)\\
&= w_{1:t-1}(F(x_t) - F(x_{t-1}) + w_t(F(x_T) - F(x_\star))\\
&= w_{1:t-1}\langle \nabla F(x_t), x_t - x_{t-1}\rangle + w_t(F(x_t) - F(x_\star))\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t)
\end{aligned}$$

using (9):

$$\begin{aligned}
&= w_t\langle \nabla F(x_t), z_t - x_t\rangle + w_t(F(x_t) - F(x_\star)) - w_{1:t-1}B_F(x_{t-1}, x_t)\\
&= w_t\langle \nabla F(x_t), z_t - x_t\rangle + w_t(F(x_t) - F(y_t)) + w_t(F(y_t) - F(x_\star))\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t)\\
&= w_t\langle \nabla F(x_t), z_t - x_t\rangle + w_t\langle \nabla F(x_t), x_t - y_t\rangle + w_t\langle \nabla F(y_t), y_t - x_\star\rangle\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t) - w_tB_F(y_t, x_t) - w_tB_F(x_\star, y_t)\\
&= w_t\langle \nabla F(x_t) - \nabla F(y_t), z_t - y_t\rangle + w_t\langle \nabla F(y_t), z_t - x_\star\rangle\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t) - w_tB_F(y_t, x_t) - w_tB_F(x_\star, y_t)
\end{aligned}$$

using (10):

$$\begin{aligned}
&= w_t\frac{1-\alpha_t}{\alpha_t}\langle \nabla F(x_t) - \nabla F(y_t), y_t - x_t\rangle + w_t\langle \nabla F(y_t), z_t - x_\star\rangle\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t) - w_tB_F(y_t, x_t) - w_tB_F(x_\star, y_t)
\end{aligned}$$

using (7):

$$\begin{aligned}
&= w_t\langle \nabla F(y_t), z_t - x_\star\rangle\\
&\quad - w_t\frac{1-\alpha_t}{\alpha_t}(B_F(x_t, y_t) + B_F(y_t, x_t))\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t) - w_tB_F(y_t, x_t) - w_tB_F(x_\star, y_t)\\
&= w_t\langle \nabla F(y_t), z_t - x_\star\rangle\\
&\quad - \frac{w_t}{\alpha_t}B_F(y_t, x_t) - \frac{w_t(1-\alpha_t)}{\alpha_t}B_F(x_t, y_t)\\
&\quad - w_{1:t-1}B_F(x_{t-1}, x_t) - w_tB_F(x_\star, y_t).
\end{aligned}$$

□

D Acceleration

In this section, we show that by instantiating our framework with an *optimistic* online learning algorithm (Rakhlin and Sridharan, 2013), we achieve accelerated convergence guarantees. Our results match those available in the prior literature (Kavis et al., 2019; Joulani et al., 2020). Our approach is inspired by Joulani et al. (2020),: their method is based upon a version of Theorem 5 for the special case that $\alpha_t = 0$. Our result simply extends their analysis to $\alpha_t = O(1/t)$.

First, we establish an important technical Corollary that simplifies Theorem 5 in the case that F is smooth and α_t is sufficiently small.

Corollary 1. *Under the same conditions as Theorem 5, suppose additionally that F is L-smooth and suppose $\alpha_t \leq \frac{w_t}{10w_{1:t}}$ for all t. Then we have for all $x_\star$:*

$$\begin{aligned}
\mathbb{E}[F(x_T) - F(x_\star)] \leq \mathbb{E}\left[\frac{\sum_{t=1}^T w_t\langle g_t, z_t - x_\star\rangle}{w_{1:T}}\right]\\
- \mathbb{E}\left[\frac{\sum_{t=1}^T w_{1:t-1}\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2}{6Lw_{1:T}}\right],
\end{aligned}$$

where above the value of y_0 is arbitrary (since the coefficient is $w_{1:0} = 0$).

Proof. The key thing is to observe that smoothness implies $B_F(a,b) \geq 2L\|\nabla F(a) - \nabla F(b)\|^2$. The rest of the argument is straightforward manipulation of the terms in Theorem 5:

$$-\frac{w_t}{\alpha_t}B_F(y_t,x_t) - \frac{w_t(1-\alpha_t)}{\alpha_t}B_F(x_t,y_t) \leq -\frac{w_t(2-\alpha_t)}{2L\alpha_t}\|\nabla F(x_t) - \nabla F(y_t)\|^2$$
$$-w_{1:t-1}B_F(x_{t-1},x_t) - w_tB_F(x_\star,y_t) \leq -\frac{w_{1:t-1}}{2L}\|\nabla F(x_t) - \nabla F(x_{t-1})\|^2.$$

Next, observe that for any vectors a, b, c, for any $\lambda > 0$:

$$\begin{aligned}-\|a+b+c\|^2 &= -\|a\|^2 - \|b\|^2 - \|c\|^2 - 2\langle a,b\rangle - 2\langle b,c\rangle - 2\langle a,c\rangle \\ &\leq -(1-2/\lambda)\|a\|^2 + (2\lambda-1)(\|b\|^2 + \|c\|^2),\end{aligned}$$

where we have used Young's inequality: $|\langle v,w\rangle| \leq \frac{\|v\|^2}{2\lambda} + \frac{\lambda\|w\|^2}{2}$. Therefore, setting $\lambda_t = 3$ we obtain:

$$\begin{aligned}&- w_{1:t-1}B_F(x_{t-1},x_t) - w_tB_F(x_\star,y_t) \\ &\leq -\frac{w_{1:t-1}}{6L}\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2 \\ &\quad + \frac{5w_{1:t-1}}{2L}(\|\nabla F(x_t) - \nabla F(y_t)\|^2 + \|\nabla F(x_{t-1}) - \nabla F(y_{t-1})\|^2).\end{aligned}$$

Now, since $\alpha_t \leq \frac{w_t}{10w_{1:t}} \leq 1$, we obtain:

$$\begin{aligned}&-\frac{w_t}{\alpha_t}B_F(y_t,x_t) - \frac{w_t(1-\alpha_t)}{\alpha_t}B_F(x_t,y_t) - w_{1:t-1}B_F(x_{t-1},x_t) - w_tB_F(x_\star,y_t) \\ &\quad\leq -\frac{w_{1:t-1}}{6L}\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2 \\ &\quad - \frac{5w_{1:t}}{2L}\|\nabla F(x_t) - \nabla F(y_t)\|^2 + -\frac{5w_{1:t-1}}{2L}\|\nabla F(x_{t-1}) - \nabla F(y_{t-1})\|^2.\end{aligned}$$

Now summing over t from 1 to T (and dropping one negative term), the sum telescopes to:

$$\sum_{t=1}^{T} -\frac{w_{1:t-1}}{6L}\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2.$$

The result now follows from Theorem 5. □

Now, we consider the case that z_t is given by an optimistic mirror descent algorithm:

Corollary 2. *Suppose F is L-smooth. Define $g_0 = 0$ and suppose also that for some D satisfying $D \geq \|y_1 - x_\star\|$:*

$$\sum_{t=1}^{T} w_t\langle g_t, z_t - x_\star\rangle \leq D\sqrt{\sum_{t=1}^{T} w_t^2\|g_t - g_{t-1}\|^2}.$$

Finally, suppose $\mathbb{E}[\|g_t - g_{t-1}\|^2] \leq \|\nabla F(y_t) - \nabla F(y_{t-1})\|^2 + \sigma_t^2$ for some constants $\sigma_1,\ldots,\sigma_T$ (these are just variance bounds on the stochastic gradient oracle). Then with $w_t = t$ and $\alpha_t \leq \frac{1}{5(t-1)}$, we have:

$$\begin{aligned}\mathbb{E}[F(x_T) - F(x_\star)] &\leq \frac{14D^2L}{T(T+1)} + \frac{2D\sqrt{\sum_{t=1}^{T} t^2\sigma_t^2}}{T(T+1)} \\ &= O\left(\frac{D^2L}{T^2} + \frac{D\sigma}{\sqrt{T}}\right),\end{aligned}$$

where σ is uniform upper-bound on σ_t. Note that the algorithm does not need to know L or σ.

Algorithms producing z sequences obtaining the guarantee stated here are called "optimistic online learning algorithms".

Proof. Applying Corollary 1, we obtain immediately:

$$\begin{aligned}
&\frac{T(T+1)}{2}\mathbb{E}[F(x_T) - F(x_\star)] \\
&\leq \mathbb{E}\left[D\sqrt{\sum_{t=1}^{T} t^2\|g_t - g_{t-1}\|^2} - \sum_{t=1}^{T}\frac{(t-1)t}{12L}\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2\right] \\
&\leq D\sqrt{\sum_{t=1}^{T} t^2\mathbb{E}\left[\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2\right] + t^2\sigma_t^2} + \frac{\|\nabla F(y_1)\|^2}{24L} \\
&\quad - \frac{1}{24L}\sum_{t=1}^{T} t^2\mathbb{E}\left[\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2\right] \\
&\leq D\sqrt{\sum_{t=1}^{T} t^2\mathbb{E}\left[\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2\right]} + D\sqrt{\sum_{t=1}^{T} t^2\sigma_t^2} + \frac{\|\nabla F(y_1)\|^2}{24L} \\
&\quad - \frac{1}{24L}\sum_{t=1}^{T} t^2\mathbb{E}\left[\|\nabla F(y_t) - \nabla F(y_{t-1})\|^2\right]
\end{aligned}$$

Using the identity $A\sqrt{C} - BC \leq \frac{A^2}{4B}$:

$$\begin{aligned}
&\leq 6D^2L + \frac{L\|y_1 - x_\star\|^2}{24} + D\sqrt{\sum_{t=1}^{T} t^2\sigma_t^2} \\
&\leq 7D^2L + D\sqrt{\sum_{t=1}^{T} t^2\sigma_t^2}.
\end{aligned}$$

Divide by $\frac{T(T+1)}{2}$ to conclude the result. □

D.1 An Optimistic Regret Bound

In this section we provide an algorithm that achieves the optimistic regret bound required for our acceleration result Corollary 2. This algorithm is a mild variation on the established literature (Rakhlin and Sridharan, 2013; Chiang et al., 2012; Hazan and Kale, 2010; Joulani et al., 2017) to slightly improve a technical dependence on the maximum gradient value.

Lemma 1. *For a sequence of vectors $g_1, \ldots, g_T$, set $\eta_t = \frac{D}{\sqrt{2\sum_{i=1}^{t}\|g_i - g_{i-1}\|^2}}$ with $g_0 = 0$, define $m_t = \max_{i\leq t}\|g_i - g_{i-1}\|$ and define the sequence of vectors z_t, z_t' and $\tilde{g}_t$ by the recursions:*

$$\begin{aligned}
z_1 &= z_1' = 0 \\
\tilde{g}_t &= g_{t-1} + \min\left(m_{t-1}, \|g_t - g_{t-1}\|\right)\frac{g_t - g_{t-1}}{\|g_t - g_{t-1}\|} \\
\eta_t &= \frac{D}{\sqrt{m_t^2 + \sum_{i=1}^{t}\|\tilde{g}_i - g_{i-1}\|^2}} \\
z_{t+1}' &= \Pi_{\|z_{t+1}'\|\leq D} z_t' - \eta_t\tilde{g}_t \\
z_{t+1} &= \Pi_{\|z_{t+1}\|\leq D} z_{t+1}' - \eta_t g_t.
\end{aligned}$$

Then:

$$\sum_{t=1}^{T}\langle g_t, z_t - x_\star\rangle \leq 7D\sqrt{2\sum_{t=1}^{T}\|g_t - g_{t-1}\|^2}.$$

Proof. For purposes of notation, define $g_0 = 0$ and $z'_0 = 0$. Further, observe that:

$$\begin{aligned}
\|\tilde{g}_t - g_{t-1}\| &\leq m_{t-1} \\
\|\tilde{g}_t - g_{t-1}\| &\leq \|g_t - g_{t-1}\| \\
\|\tilde{g}_t - g_t\| &= m_t - m_{t-1} \\
\eta_t &\leq \frac{D}{\sqrt{\sum_{i=1}^{t+1} \|\tilde{g}_i - g_{i-1}\|^2}} \\
\frac{1}{\eta_T} &\leq \frac{\sqrt{2\sum_{t=1}^{T} \|g_t - g_{t-1}\|^2}}{D}.
\end{aligned}$$

Next, notice that $z'_{t+1} = \text{argmin}_{\|z\| \leq D} \langle \tilde{g}_t, z \rangle + \frac{1}{2\eta_t}\|z - z'_t\|^2$. Therefore since $\|x_\star\| \leq D$, by first order optimality conditions:

$$\begin{aligned}
\left\langle \tilde{g}_t + \frac{z'_{t+1} - z'_t}{\eta_t}, z'_{t+1} - x_\star \right\rangle &\leq 0 \\
\langle \tilde{g}_t, z'_{t+1} - x_\star \rangle &\leq \frac{1}{\eta_t} \langle z'_t - z'_{t+1}, z'_{t+1} - x_\star \rangle \\
&= \frac{\|z'_t - x_\star\|^2}{2\eta_t} - \frac{\|z'_{t+1} - x_\star\|^2}{2\eta_t} - \frac{\|z'_{t+1} - z'_t\|^2}{2\eta_t}.
\end{aligned}$$

Similarly, we have $z_t = \text{argmin}_{\|z\| \leq D} \langle g_{t-1}, z \rangle + \frac{1}{2\eta_{t-1}}\|z - z'_t\|^2$. From this we have:

$$\begin{aligned}
\left\langle g_{t-1} + \frac{z_t - z'_t}{\eta_{t-1}}, z_t - z'_{t+1} \right\rangle &\leq 0 \\
\langle g_{t-1}, z_t - z'_{t+1} \rangle &\leq \frac{\|z'_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_t\|^2}{2\eta_{t-1}} \\
\langle \tilde{g}_t, z_t - z'_{t+1} \rangle &\leq \frac{\|z'_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_t\|^2}{2\eta_{t-1}} \\
&\quad + \langle \tilde{g}_t - g_{t-1}, z_t - z'_{t+1} \rangle
\end{aligned}$$

by Young's inequality:

$$\begin{aligned}
&\leq \frac{\|z'_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_{t+1}\|^2}{2\eta_{t-1}} - \frac{\|z_t - z'_t\|^2}{2\eta_{t-1}} \\
&\quad + \frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} + \frac{\|z_t - z'_{t+1}\|^2}{2\eta_{t-1}} \\
&\leq \frac{\|z'_t - z'_{t+1}\|^2}{2\eta_{t-1}} + \frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2}.
\end{aligned}$$

So, combining these facts (and noticing that $\eta_{t-1} \geq \eta_t$:)

$$\begin{aligned}
\langle \tilde{g}_t, z_t - x_\star \rangle &\leq \frac{\|z'_t - x_\star\|^2}{2\eta_t} - \frac{\|z'_{t+1} - x_\star\|^2}{2\eta_t} + \frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} \\
\langle g_t, z_t - x_\star \rangle &\leq \frac{\|z'_t - x_\star\|^2}{2\eta_t} - \frac{\|z'_{t+1} - x_\star\|^2}{2\eta_t} + \frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} + \langle g_t - \tilde{g}_t, z_t - x_\star \rangle \\
&\leq \frac{\|z'_t - x_\star\|^2}{2\eta_t} - \frac{\|z'_{t+1} - x_\star\|^2}{2\eta_t} + \frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} + 2D(m_t - m_{t-1}).
\end{aligned}$$

So, we have:

$$\begin{aligned}
\sum_{t=1}^{T}\langle g_t, z_t - x_\star\rangle &\le 2Dm_T + \frac{\|z_1' - x_\star\|^2}{2\eta_1} + \sum_{t=2}^{T}\frac{\|z_t' - x_\star\|^2}{2}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) \\
&\quad + \sum_{t=1}^{T}\frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} \\
&\le 2Dm_T + 4D^2/\eta_T + \sum_{t=1}^{T}\frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} \\
&\le 6D^2/\eta_T + \sum_{t=1}^{T}\frac{\eta_{t-1}\|\tilde{g}_t - g_{t-1}\|^2}{2} \\
&\le 6D^2/\eta_T + \sum_{t=1}^{T}\frac{D\|\tilde{g}_t - g_{t-1}\|^2}{2\sqrt{\sum_{i=1}^{t}\|\tilde{g}_i - g_{i-1}\|^2}} \\
&\le 6D^2/\eta_T + D\sqrt{\sum_{t=1}^{T}\|\tilde{g}_t - g_{t-1}\|^2} \\
&\le 7D\sqrt{2\sum_{t=1}^{T}\|g_t - g_{t-1}\|^2}.
\end{aligned}$$

□

E Strongly Convex Losses

Suppose that the expected loss F is actually known to be μ-strongly convex. Then we'd like to have a convergence guarantee of $O(1/\mu T)$. This is achieved in Theorem 6 below.

Theorem 6. *Under the same assumptions as Theorem 5, define $\ell_t(z) = \langle g_t, z\rangle + \frac{\mu}{2}\|y_t - z\|^2$. Define the "regret" of the sequence z_t as:*

$$\textit{Regret}_\ell(x_\star) = \sum_{t=1}^{T} w_t(\ell_t(z_t) - \ell_t(x_\star)).$$

Then we have for $x_\star = \operatorname{argmin} F$:

$$\mathbb{E}[F(x_T) - F(x_\star)] \le \mathbb{E}\left[\frac{\textit{Regret}_\ell(x_\star) - \sum_{t=1}^{T}\frac{w_t\mu}{2}\|z_t - y_t\|^2}{w_{1:T}}\right].$$

In particular, suppose $\|x_\star\| \le D$ for some known bound D and $\|g_t\| \le G$ for all t for some G so long as $\|y_t\| \le D$. Then if we define $w_t = t$ for all t and set z_t by:

$$z_{t+1} = \Pi_{\|z\|\le D}\left[z_t - \frac{2(g_t + \mu(z_t - y_t))}{\mu(t+1)}\right].$$

then we have:

$$\mathbb{E}[F(x_T) - F(x_\star)] \le \frac{2(G + 2\mu D)^2}{\mu(T+1)}.$$

Proof. From Theorem 5, we have:

$$\mathbb{E}[F(x_T) - F(x_\star)] \le \mathbb{E}\left[\frac{\sum_{t=1}^{T} w_t\langle g_t, z_t - x_\star\rangle}{w_{1:T}} - \frac{\sum_{t=1}^{T} w_t B_F(x_\star, y_t)}{w_{1:T}}\right].$$

Now, since F is μ-strongly convex, we have $B_F(x_\star, y_t) \geq \frac{\mu}{2}\|y_t - x_\star\|^2$. Further, we have:

$$\sum_{t=1}^{T} w_t \langle g_t, z_t - x_\star \rangle = \sum_{t=1}^{T} w_t(\ell_t(z_t) - \ell_t(x_\star)) - \frac{w_t\mu}{2}\|z_t - y_t\|^2 + \frac{w_t\mu}{2}\|x_\star - y_t\|^2.$$

From this we obtain the desired result:

$$\mathbb{E}[F(x_T) - F(x_\star)] \leq \mathbb{E}\left[\frac{\text{Regret}_\ell(x_\star) - \sum_{t=1}^{T}\frac{w_t\mu}{2}\|z_t - y_t\|^2}{w_{1:T}}\right].$$

For the final statement, observe that with $w_t = t$, $w_t\ell_t(z) = t\langle g_t, z\rangle + \frac{t\mu}{2}\|z - y_t\|^2$ is $t\mu$-strongly convex. Therefore if we use learning rate $\eta_t = \frac{1}{\mu w_{1:t}} = \frac{2}{\mu t(t+1)}$, then standard analysis of projected OGD yields:

$$\begin{aligned}
\sum_{t=1}^{T} t(\ell_t(z_t) - \ell_t(x_\star)) &\leq \sum_{t=1}^{T} t\langle \nabla\ell_t(z_t), z_t - x_\star\rangle - \frac{t\mu}{2}\|z_t - x_\star\|^2 \\
&\leq \|z_1 - x_\star\|^2\left(\frac{1}{2\eta_1} - \frac{\mu}{2}\|z_t - x_\star\|^2\right) - \frac{\|z_{T+1} - x_\star\|^2}{2\eta_T} \\
&\quad + \sum_{t=2}^{T}\|z_t - x_\star\|^2\left(\frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}} - \frac{t\mu}{2}\right) + \sum_{t=1}^{T}\frac{\eta_t t^2\|\nabla\ell_t(z_t)\|^2}{2} \\
&\leq \sum_{t=1}^{T}\frac{\eta_t t^2\|\nabla\ell_t(z_t)\|^2}{2} \\
&\leq \frac{1}{\mu}\sum_{t=1}^{T}\|\nabla\ell_t(z_t)\|^2 \\
&= \frac{1}{\mu}\sum_{t=1}^{T}\|g_t + \mu(z_t - y_t)\|^2 \\
&\leq \frac{T(G + 2\mu D)^2}{\mu}.
\end{aligned}$$

where in the last inequality we have observed that since $\|z_t\| \leq D$ and y_t is a linear combination of past z values, $\|y_t\| \leq D$ as well. Finally, observing that $w_{1:T} = \frac{T(T+1)}{2}$, the result follows. □

F Large Step size convergence

Theorem 7. *Consider the online learning setting with bounded gradients g_t. Let $z_{t+1} = z_t - \gamma g_t$. Let $D = \|z_1 - z_*\|$ for arbitrary reference point z_* and define $G = \max_{t\leq T}\|g_t\|$. Suppose that the chosen step-size is $\gamma = D/G$, then if it holds that:*

$$\sum_{t=1}^{T}\langle g_t, z_t - z_1\rangle \leq D\sqrt{\sum_{t=1}^{T}\|g_t\|^2}, \tag{11}$$

then:

$$\frac{1}{T}\sum_{t=1}^{T}\langle g_t, z_t - z_*\rangle = \mathcal{O}\left(\frac{D}{T}\sqrt{\sum_{t=1}^{T}\|g_t\|^2}\right).$$

Proof. Consider SGD with fixed step size γ:

$$z_{t+1} = z_t - \gamma g_t.$$

Let

$$s_{T+1} = \sum_{t=1}^{T}\gamma g_t.$$

Recall from D-Adaptation (Defazio and Mishchenko, 2023) theory that:

$$\sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_1 \right\rangle = \frac{1}{2} \sum_{t=1}^{T} \gamma^2 \left\| g_t \right\|^2 - \frac{1}{2} \left\| s_{t+1} \right\|^2 \tag{12}$$

and:

$$\sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_* \right\rangle \leq \left\| s_{T+1} \right\| D + \sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_1 \right\rangle . \tag{13}$$

Now suppose that the regret at time T is negative. Then trivially the theorem holds:

$$\frac{1}{T} \sum_{t=1}^{T} \left\langle g_t, z_t - z_* \right\rangle \leq 0 = \mathcal{O}\left(\frac{D}{T} \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2} \right),$$

therefore, without loss of generality we may assume that $\sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_* \right\rangle \geq 0$. Then from combining Equation 13 with Equation 12 we have:

$$0 \leq -\frac{1}{2} \left\| s_{T+1} \right\|^2 + \left\| s_{T+1} \right\| D + \frac{1}{2} \sum_{t=1}^{T} \gamma^2 \left\| g_t \right\|^2 .$$

This is a quadratic equation in $\left\| s_{T+1} \right\|$ which we can solve explicitly via the quadratic formula, taking the largest root:

$$\left\| s_{T+1} \right\| \leq \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Plugging in the values $a = -\frac{1}{2}$, $b = D$, $c = \frac{1}{2} \sum_{t=1}^{T} \gamma^2 \left\| g_t \right\|^2$:

$$D \pm \sqrt{D^2 + \sum_{t=1}^{T} \gamma^2 \left\| g_t \right\|^2} \leq 2D + \sqrt{\sum_{t=1}^{T} \gamma^2 \left\| g_t \right\|^2}.$$

Therefore:

$$\left\| s_{T+1} \right\| \leq 2D + \gamma \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2}.$$

Substituting this into Equation 13:

$$\sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_* \right\rangle \leq 2D^2 + \gamma D \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2} + \sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_1 \right\rangle .$$

Therefore, if $\sum_{t=1}^{T} \left\langle g_t, z_t - z_1 \right\rangle \leq D \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2}$ then:

$$\sum_{t=1}^{T} \gamma \left\langle g_t, z_t - z_* \right\rangle \leq 2D^2 + 2\gamma D \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2}.$$

Plugging in $\gamma = D/G$:

$$\begin{aligned} \sum_{t=1}^{T} \left\langle g_t, z_t - z_* \right\rangle &\leq 2DG + 2D \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2} \\ &\leq 4D \sqrt{\sum_{t=1}^{T} \left\| g_t \right\|^2}, \end{aligned}$$

and the theorem follows. □

G Experimental Setup

G.1 Convex experiments

Each dataset is obtained from the LIBSVM repository and used without modifications.

Hyper-parameter	Value
GPUs	1×V100
Batch size	16
Epochs	100
Seeds	10
Schedule-Free β	0.9

Hyper-parameter	Value
Decay	0.0
Optimizer	Adam
β_1	0.9
β_2	0.95

G.2 CIFAR-10

We used custom training code based on the PyTorch tutorial code for this problem. Following standard data-augmentation practises, we appliyed random horizontal flips and random offset cropping down to 32x32, using reflection padding of 4 pixels. Input pixel data was normalized by centering around 0.5.

Hyper-parameter	Value
Architecture	Wide ResNet 16-8
Epochs	300
GPUs	1×V100
Batch size per GPU	128
Schedule-Free warmup	5%

Hyper-parameter	Value
Seeds	10
decay	0.0001
Momentum	0.9
Schedule-Free LR	10
Schedule-Free β	0.9

G.3 CIFAR-100

We used the same codebase as for our CIFAR-10 experiments, with the same data augmentation.

We normalized each input image using fixed mean and standard error values derived from pre-processing the data.

Hyper-parameter	Value
Architecture	DenseNet [6,12,24,16], growth rate 12
Epochs	300
GPUs	1×V100
Schedule-Free β	0.9
Schedule-Free warmup	5%

Hyper-parameter	Value
Batch size per GPU	64
Seeds	10
Decay	0.0002
Momentum	0.9
Schedule-Free LR	5

G.4 SVHN

We used the same codebase as for our CIFAR experiments, and following the same data preprocessing.

Hyper-parameter	Value
Batch size	32
Weight decay Cosine	0.0001
Weight decay Step Sched	5e-5
Baseline LR	0.1
Seeds	10

Hyper-parameter	Value
Warmup	5%
Schedule-Free decay	0.0002
Schedule-Free LR	1.0
Schedule-Free β	0.9

G.5 ImageNet

We used the same code-base as for our CIFAR-10 experiments, and applied the same preprocessing procedure. The data-augmentations consisted of PyTorch's RandomResizedCrop, cropping to 224x224 followed by random horizontal flips. Test images used a fixed resize to 256x256 followed by a center crop to 224x224.

Hyper-parameter	Value
Architecture	ResNet50
Epochs	100
GPUs	8×V100
Batch size per GPU	32
Schedule-Free Decay	0.00005
Schedule-Free LR	1.5

Hyper-parameter	Value
Seeds	5
Decay	0.0001
Momentum	0.9
Schedule-Free β	0.9
Schedule-Free warmup	5%

G.6 IWSLT14

We used the FairSeq framework [3] for our experiments. Rather than a vanilla LSTM we use the variant from Wiseman and Rush (2016) provided in the FairSeq codebase.

Hyper-parameter	Value
Architecture	lstm_wiseman_iwslt_de_en
Max Epoch	55
GPUs	1×V100
Tokens per batch	4096
Warmup steps	4000
Dropout	0.3
Label smoothing	0.1
Schedule-Free LR	0.02
Schedule-Free warmup	5%
Baseline schedule	Linear Decay

Hyper-parameter	Value
Share decoder, input, output embed	True
Float16	True
Update Frequency	1
Seeds	10
Decay	0.05
β_1, β_2	0.9, 0.98
Schedule-Free β	0.9
Baseline LR	0.01

G.7 NanoGPT

We followed the NanoGPT codebase [4] as closely as possible, matching the default batch-size, training length and schedule. Our runs replicate the stated 2.85 loss in the documentation. Disabling gradient norm clipping is crucial for the Schedule-Free runs.

Hyper-parameter	Value
Architecture	transformer_lm_gpt
Batch size per gpu	12
Max Iters	600,000
GPUs	40×V100
Tokens per sample	512
Dropout	0.0
Baseline LR	0.0005
Warmup	2,000
Schedule-Free LR	0.005
Schedule-Free β	0.98
Schedule-Free decay	0.05

Hyper-parameter	Value
Block Size	1024
Num layer	12
Num head	12
Num embd	768
Float16	True
Update Frequency	16
Seeds	5
Decay	0.1
β_1, β_2	0.9, 0.95
Gradient Clipping	0.0

G.8 MAE

Our implementation uses the offical code[5], with hyper-parameters following examples given in the repository.

[3] https://github.com/facebookresearch/fairseq
[4] https://github.com/karpathy/nanoGPT
[5] https://github.com/fairinternal/mae

Hyper-parameter	Value
Model	vit_base_patch16
Epochs	100
GPUs	32×V100
Batch Size	32
Baseline LR	5e-4
Layer Decay	0.65
Weight Decay	0.05
Schedule-Free β	0.9

Hyper-parameter	Value
Schedule-Free LR	0.0.002
Schedule-Free decay	0.05
Schedule-Free β	0.9
Drop Path	0.1
Reprob	0.25
Mixup	0.8
Cutmix	1.0

G.9 DLRM

We used a custom implementation of the DLRM model based on the publicly available code. Our optimizer uses dense gradients for implementation simplicity, although sparse-gradients using AdaGrad is a more common baseline on this problem, we consider AdaGrad variants of our scheduling approach as future work.

Hyper-parameter	Value
Iterations	300 000
Batch Size	128
Emb Dimension	16
GPUs	8×V100
Schedule-Free LR	0.0005
Schedule-Free β	0.9

Hyper-parameter	Value
Seeds	5
Decay	0.0
β_1, β_2	0.9, 0.999
Warmup	0
Baseline LR	0.0002
Baseline schedule	Linear Decay

G.10 MRI

We used the version of the the fastMRI code base at https://github.com/facebookresearch/fastMRI/tree/main/banding_removal. Note that we found that training failed using PyTorch 2 or newer, and so we ran these experiments using PyTorch 1.9.

Hyper-parameter	Value
Architecture	12 layer VarNet 2.0
Epochs	50
GPUs	8×V100
Batch size per GPU	1
Acceleration factor	4
Baseline Schedule	Linear Decay
Baseline LR	0.005

Hyper-parameter	Value
Low frequency lines	16
Mask type	Offset-1
Seeds	5
Decay	0.0
β_1, β_2	0.9, 0.999
Schedule-Free LR	0.005
Schedule-Free β	0.9

G.11 Algoperf

Our full algoperf entry is availiable at https://github.com/facebookresearch/schedule_free/tree/main/schedulefree/algoperf. The hyper-parameters used for the self-tuning track submission are listed below.

Hyper-parameter	Value
Learning Rate	0.0025
one-minus Beta1	0.1
Beta2 (default)	0.9955159689799007
Weight Decay (default)	0.08121616522670176

Hyper-parameter	Value
Dropout Rate	0.1
Warmup Percentage	2%
Label Smoothing	0.2
Polynomial in c_t average	0.75