Learning-Rate-Free Learning by D-Adaptation

Aaron Defazio
Meta AI, Fundamental AI Research (FAIR) team

Konstantin Mishchenko
Samsung AI Center

Abstract

The speed of gradient descent for convex Lipschitz functions is highly dependent on the choice of learning rate. Setting the learning rate to achieve the optimal convergence rate requires knowing the distance D from the initial point to the solution set. In this work, we describe a single-loop method, with no back-tracking or line searches, which does not require knowledge of D yet asymptotically achieves the optimal rate of convergence for the complexity class of convex Lipschitz functions. Our approach is the first parameter-free method for this class without additional multiplicative log factors in the convergence rate. We present extensive experiments for SGD and Adam variants of our method, where the method automatically matches hand-tuned learning rates across more than a dozen diverse machine learning problems, including large-scale vision and language problems. Our method is practical, efficient and requires no additional function value or gradient evaluations each step. An open-source implementation is available[1].

1. Introduction

We consider the problem of unconstrained convex minimization,

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

where f has Lipschitz constant G and a non-empty set of minimizers. The standard approach to solving it is the subgradient method that, starting at a point x_0, produces new iterates following the update rule:

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

where $g_k \in \partial f(x_k)$ is a subgradient of f. The *learning rate* γ_k, also known as the *step size*, is the main quantity controlling if and how fast the method converges. If the learning rate sequence is chosen too large, the method might oscillate around the solution, whereas small values lead to very slow progress.

Setting γ_k optimally requires knowledge of the distance to a solution. In particular, denote x_* to be any minimizer of f, D to be the associated distance $D = \|x_0 - x_*\|$, and f_* to be the optimal value, $f_* = f(x_*)$. Then, using the step size

$$\gamma_k = \frac{D}{G\sqrt{n}},$$

1. https://github.com/facebookresearch/dadaptation

Algorithm 1 Dual Averaging with D-Adaptation

Input: d_0, x_0
$s_0 = 0$, $g_0 \in \partial f(x_0)$, $\gamma_0 = 1/\|g_0\|$
If $g_0 = 0$, exit with $\hat{x}_n = x_0$
for $k = 0$ **to** n **do**
  $g_k \in \partial f(x_k)$
  $s_{k+1} = s_k + d_k g_k$
  $$\gamma_{k+1} = \frac{1}{\sqrt{\sum_{i=0}^{k} \|g_i\|^2}}$$
  $$\hat{d}_{k+1} = \frac{\gamma_{k+1} \|s_{k+1}\|^2 - \sum_{i=0}^{k} \gamma_i d_i^2 \|g_i\|^2}{2 \|s_{k+1}\|}$$
  $d_{k+1} = \max\left(d_k, \hat{d}_{k+1}\right)$
  $x_{k+1} = x_0 - \gamma_{k+1} s_{k+1}$
end for
Return $\hat{x}_n = \frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k x_k$

the average iterate $\hat{x}_n$ converges in terms of function value at an inverse square-root rate:

$$f(\hat{x}_n) - f_* = \mathcal{O}(DG/\sqrt{n}).$$

This rate is worst-case optimal for this complexity class [20]. Knowledge of the constant G can be removed using AdaGrad-Norm step sizes [10, 29, 31],

$$\gamma_k = \frac{D}{\sqrt{\sum_{i=0}^{k} \|g_i\|^2}},$$

together with projection onto the D-ball around the origin. In the (typical) case where we don't have knowledge of D, we can start with loose lower and upper bounds d_0 and $d_{\max}$, and perform a hyper-parameter grid search on a log-spaced scale, with the rate:

$$f(x_n) - f_* = \mathcal{O}\left(\frac{DG \log(d_{\max}/d_0)}{\sqrt{n+1}}\right).$$

In most machine learning applications this grid search is the current standard practice.

In this work we take a different approach. We describe a modification of dual averaging that achieves the optimal rate, for sufficiently large n, by maintaining and updating a lower bound on D. Using this lower bound is provably sufficient to achieve the optimal rate of convergence, with no additional log factors, avoiding the need for hyper-parameter grid searches.

2. Algorithm

The algorithm we propose is Algorithm 1. It is a modification of the AdaGrad step size applied to weighted dual averaging, together with our key innovation: D lower bounding. At each step, we construct a lower bound $\hat{d}_k$ on D using empirical quantities. If this bound is better (i.e. larger) than our current best bound d_k of D, we use $d_k = \hat{d}_k$ in subsequent steps.

To construct the lower bound, we show that a weighted sum of the function values is bounded above as:

$$\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \leq D \left\|s_{n+1}\right\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} d_k^2 \left\|g_k\right\|^2 - \frac{\gamma_{n+1}}{2} \left\|s_{n+1}\right\|^2.$$

There are two key differences from the classical bound:

$$\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \leq \frac{1}{2}\gamma_{n+1}^{-1} D^2 + \sum_{k=0}^{n} \frac{\gamma_k}{2} d_k^2 \left\|g_k\right\|^2$$

Firstly, we are able to gain an additional negative term $-\frac{1}{2}\gamma_{n+1}\left\|s_{n+1}\right\|^2$. Secondly, we replace the typical D^2 error term with $D\left\|s_{n+1}\right\|$, following the idea of Carmon and Hinder [2]. This bound is tighter than the classical bound, and equivalent when $D = \left\|x_0 - x_{n+1}\right\|$, since:

$$D\left\|s_{n+1}\right\| - \frac{1}{2}\gamma_{n+1}\left\|s_{n+1}\right\|^2 = \frac{1}{2}\gamma_{n+1}^{-1}\left(D^2 - \left(D - \left\|x_0 - x_{n+1}\right\|\right)^2\right) \leq \frac{1}{2}\gamma_{n+1}^{-1} D^2.$$

From our bound, using the fact that

$$\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \geq 0,$$

we have:

$$0 \leq D\left\|s_{n+1}\right\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} d_k^2 \left\|g_k\right\|^2 - \frac{\gamma_{n+1}}{2}\left\|s_{n+1}\right\|^2,$$

which can be rearranged to yield a lower bound on D, involving only known quantities:

$$D \geq \hat{d}_{n+1} = \frac{\gamma_{n+1}\left\|s_{n+1}\right\|^2 - \sum_{k=0}^{n} \gamma_k d_k^2 \left\|g_k\right\|^2}{2\left\|s_{n+1}\right\|}.$$

This bound is potentially vacuous if $\left\|s_{n+1}\right\|^2$ is small in comparison to $\sum_{k=0}^{n} \gamma_k d_k^2 \left\|g_k\right\|^2$. This only occurs once the algorithm is making fast-enough progress that bound adjustment is not necessary at that time.

Theorem 1 *For a convex G-Lipschitz function f, Algorithm 1 returns a point $\hat{x}_n$ such that:*

$$f(\hat{x}_n) - f(x_*) = \mathcal{O}\left(\frac{DG}{\sqrt{n+1}}\right),$$

as $n \to \infty$, where $D = \left\|x_0 - x_*\right\|$ for any x_* in the set of minimizers of f, as long as $d_0 \leq D$.

The above result is asymptotic due to the potential of worst-case functions. For any fixed choice of n, a function could be constructed such that Algorithm 1 run for n steps has a dependence on d_0. In the next theorem, we prove a non-asymptotic bound that is worse only by a factor of $\log_2(D/d_0)$. This guarantee is significantly better than using the subgradient method with step size proportional to d_0, which would incur an extra factor of D/d_0.

Theorem 2 *Consider Algorithm 1 run for $n \geq \log_2(D/d_0)$ steps with the step size modified to be*

$$\gamma_{k+1} = \frac{1}{\sqrt{G^2 + \sum_{i=0}^{k} \|g_i\|^2}}. \tag{1}$$

If we return the point $\hat{x}_t = \frac{1}{\sum_{k=0}^{t} d_k} \sum_{k=0}^{t} d_k x_k$ where t is chosen to be

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

then

$$f(\hat{x}_t) - f_* \leq 8 \frac{\log_2(D/d_0)}{n+1} D \sqrt{\sum_{k=0}^{t} \|g_k\|^2} \leq 8 \frac{DG \log_2(D/d_0)\sqrt{t+1}}{n+1}.$$

The worst-case behavior occurs when d_k grows exponentially from d_0, but slowly, only reaching D at the last step. For this reason, the worst case construction requires knowledge of the stopping time n. The modification to the step size can be avoided at the cost of having an extra term, namely we would have the following guarantee for the same iterate $\hat{x}_t$:

$$f(\hat{x}_t) - f_* \leq \frac{8DG \log_2(D/d_0)}{\sqrt{n+1}} + \frac{4DG^2 \log_2(D/d_0)}{(n+1)\|g_0\|}.$$

Notice that, unlike the bound in the theorem above, it also depends on the initial gradient norm $\|g_0\|$.

3. D-Adapted AdaGrad

The D-Adaptation technique can be applied on top of the coordinate-wise scaling variant of AdaGrad with appropriate modifications. Algorithm 2 presents this method. This variant estimates the distance to the solution in the ℓ_∞-norm instead of the Euclidean norm, $D_\infty = \|x_0 - x_*\|_\infty$. The theory for AdaGrad without D-Adaptation also uses the same norm to measure the distance to solution, so this modification is natural, and results in the same adaptive convergence rate as AdaGrad up to constant factors *without* requiring knowledge of D_∞.

Theorem 3 *For a convex p-dimensional function with $G_\infty = \max_x \|\nabla f(x)\|_\infty$, D-Adapted AdaGrad (Algorithm 2) returns a point $\hat{x}_n$ such that*

$$f(\hat{x}_n) - f_* = \mathcal{O}\left(\frac{\|a_{n+1}\|_1 D_\infty}{n+1}\right) = \mathcal{O}\left(\frac{pG_\infty D_\infty}{\sqrt{n+1}}\right),$$

as $n \to \infty$, where $D_\infty = \|x_0 - x_*\|_\infty$ for any x_* in the set of minimizers of f, as long as $d_0 \leq D_\infty$.

Similarly to Theorem 2, we could achieve the same result up to higher order terms without using G_∞ in the initialization of a_0.

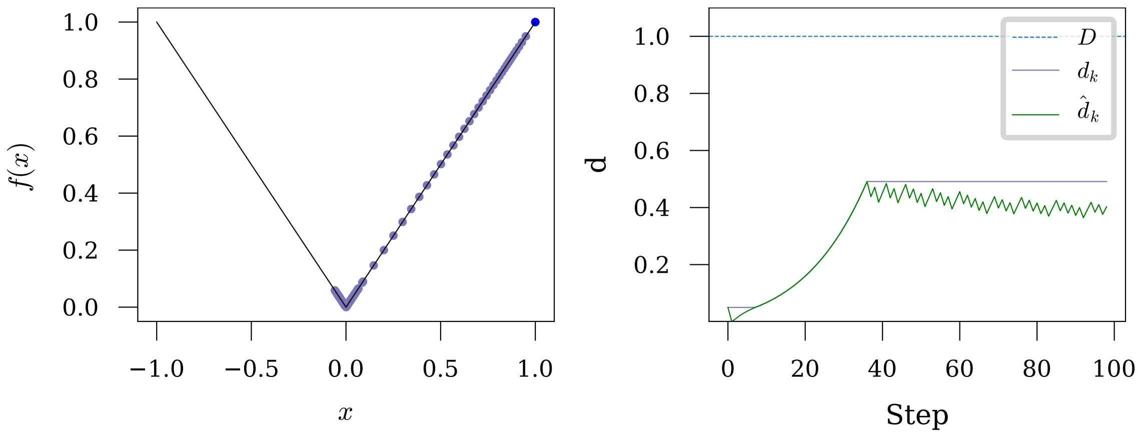

Figure 1: Toy problem illustrating the estimate of D over time, $f(x) = |x|$. $x_0 = 1.0$ is shown as a blue dot on the left plot, and the following iterates are shown in purple.

Algorithm 2 D-Adapted AdaGrad

Input: x_0, d_0 (default 10^{-6}), G_∞
$s_0 = 0$, $a_0 = G_\infty$
for $k = 0$ **to** n **do**
  $g_k \in \partial f(x_k, \xi_k)$
  $s_{k+1} = s_k + d_k g_k$
  $a_{k+1}^2 = a_k^2 + g_k^2$
  $A_{k+1} = \text{diag}(a_{k+1})$

$$\hat{d}_{k+1} = \frac{\|s_{k+1}\|^2_{A_{k+1}^{-1}} - \sum_{i=0}^{k} d_i^2 \|g_i\|^2_{A_i^{-1}}}{2\|s_{k+1}\|_1}$$

  $d_{k+1} = \max(d_k, \hat{d}_{k+1})$
  $x_{k+1} = x_0 - A_{k+1}^{-1} s_{k+1}$
end for
Return $\hat{x}_n = \frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k x_k$

4. Discussion

Figure 1 depicts the behavior of D-Adaptation on a toy problem - minimizing an absolute value function starting at $x_0 = 1.0$. Here d_0 is started at 0.1, below the known D value of 1.0. This example illustrates the growth of d_k towards D. The value of d_k typically doesn't asymptotically approach D, as this is not guaranteed nor required by our theory. Instead, we shown in Theorem 19 that under a mild assumption, d is asymptotically greater than or equal to $D/3$. The lower bound $\hat{d}_k$ will often start to decrease, and even go negative, once d_k is large enough. Negative values of $\hat{d}_k$ were seen in most of the experiments in Section 7.

The numerator of the D bound is not tight, it can be replaced with a larger inner product quantity:

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

The inner product between the step direction s and the gradient g is a quantity known as the (negative) hyper-gradient [1, 4, 7, 11, 23]. In classical applications of the hyper-gradient, the learning rate is increased when the gradient points in the same direction as the previous step, and it is decreased otherwise. In essence, the hyper-gradient indicates if the current learning rate is too large or to small. An additional hyper-learning rate parameter is needed to control the rate of change of the learning rate, whereas our approach requires no extra parameters beyond the initial d_0.

In our approach, the hyper-gradient quantity is used to provide an actual estimate of the *magnitude* of the optimal learning rate (or more precisely a lower bound), which is far more information than just a directional signal of too-large or too-small. This is important for instance when a learning rate schedule is being used, as we can anneal the learning rate down over time, without the hyper-gradient responding by pushing the learning rate back up. This is also useful during learning rate warmup, as we are able to build an estimate of D during the warmup, which is not possible when using a classical hyper-gradient approach.

Our analysis applies to a very restricted problem setting of convex Lipschitz functions. In Carmon and Hinder [2], an approach for the same setting is extended to the stochastic setting in high probability. The same extension may also be applicable here.

Our algorithm requires an initial lower bound d_0 on D. The value of d_0 does not appear in the convergence rate bound for the asymptotic setting as its contribution goes to zero as $k \to \infty$, and hence is suppressed when big-$\mathcal{O}$ notation is used. In practice very small values can be used, as d_k will grow exponentially with k when d_0 is extremely small.

5. Related Work

There are a number of techniques for optimizing Lipschitz functions that achieve independence of problem parameters. We review the major classes of approaches below.

5.1. Polyak step size

We can trade the requirement of knowledge of D to knowledge of f_*, by using the Polyak step size[24]:

$$\gamma_k = \frac{f(x_k) - f_*}{\|g_k\|^2}.$$

This gives the optimal rate of convergence without any additional log factors. Using estimates or approximations of f_* tend to result in unstable convergence, however a restarting scheme that maintains lower bounds on f_* can be shown to converge within a multiplicative log factor of the optimal rate [13].

5.2. Exact line searches

The following method relying on an exact line search also gives the optimal rate, without requiring any knowldge of problem parameters [9, 12]:

$$\begin{aligned}
s_{k+1} &= s_k + g_k, \\
\gamma_{k+1} &= \arg\min f_{k+1}\left(\frac{k+1}{k+2}x_k + \frac{1}{k+2}\left(z_0 - \gamma_{k+1}s_{k+1}\right)\right), \\
z_{k+1} &= z_0 - \gamma_{k+1}s_{k+1}, \\
x_{k+1} &= \frac{k+1}{k+2}x_k + \frac{1}{k+2}z_{k+1}.
\end{aligned}$$

Relaxing this exact line search to an approximate line search without an assumption of smoothness is non-trivial, and will potentially introduce additional dependencies on problem constants.

5.3. Bisection

Instead of running subgradient descent on every grid-point on a log spaced grid from d_0 to $d_{\max}$, we can use more sophisticated techniques to instead run a bisection algorithm on the same grid, resulting in a $\log\log$, rather than *log* dependence on $d_{\max}/d_0$ [2]:

$$f(x_n) - f_* = \mathcal{O}\left(\frac{DG\log\log(d_{\max}/d_0)}{\sqrt{n+1}}\right),$$

This can be further improved by estimating $d_{\max}$, which allows us to replace $d_{\max}$ with D in this bound.

5.4. Coin-betting

If we assume knowledge of G but not D, coin betting approaches can be used. Coin-betting [22] is normally analyzed in the online-convex optimization framework, which is more general than our setting and for that class, coin-betting methods achieve optimal regret among methods without knowledge of D, which is a log-factor worse than the best possible regret with knowledge of D [21]:

$$\text{Regret}_n = \mathcal{O}\left(DG\sqrt{(n+1)\log\left(1+D\right)}\right).$$

Using online to batch conversion gives a rate of convergence in function value of

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

5.5. Reward Doubling

Streeter and McMahan [30]'s reward-doubling technique for online learning is perhaps the most similar approach to ours. In the 1D setting, they track the sum of the quantity $x_k g_k$ and compare it to the learning rate η times $\bar{H}$, a pre-specified hyper-parameter upper bounding on the total sum of squares of the gradients. Whenever the reward sum exceeds $\eta\bar{H}$, they double the step size and reset the optimizer state, starting again from x_0. They obtain similar rates to the coin betting approach.

Algorithm 3 SGD with D-Adaptation

Input: x_0,
d_0 (default 10^{-6}),
γ_k (default 1.0).
$s_0 = 0$
If $g_0 = 0$, exit with $\hat{x}_{n+1} = x_0$
for $k = 0$ **to** n **do**
$g_k \in \partial f(x_k, \xi_k)$
$\lambda_k = \frac{d_k \gamma_k}{\|g_0\|}$
$s_{k+1} = s_k + \lambda_k g_k$
$x_{k+1} = x_k - \lambda_k g_k$
$\hat{d}_{k+1} = \frac{\|s_{k+1}\|^2 - \sum_{i=0}^{k} \lambda_i^2 \|g_i\|^2}{\|s_{k+1}\|}$
$d_{k+1} = \max\left(d_k, \hat{d}_{k+1}\right)$
end for

Algorithm 4 Adam with D-Adaptation

Input: x_0,
d_0 (default 10^{-6}),
γ_k (default 1.0),
β_1, β_2 (default 0.9, 0.999).
$s_0 = 0, m_0 = 0, v_0 = 0, r_0 = 0$
If $g_0 = 0$, exit with $\hat{x}_{n+1} = x_0$
for $k = 0$ **to** n **do**
$g_k \in \partial f(x_k, \xi_k)$
$m_{k+1} = \beta_1 m_k + (1 - \beta_1) d_k \gamma_k g_k$
$v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$
$A_{k+1} = \sqrt{v_{k+1}} + \epsilon$
$x_{k+1} = x_k - A_{k+1}^{-1} m_{k+1}$
Learning rate update
$s_{k+1} = \beta_2 s_k + (1 - \beta_2) d_k \gamma_k g_k$
$r_{k+1} = \beta_2 r_k + (1 - \beta_2) d_k^2 \gamma_k^2 \|g_i\|_{A_{k+1}^{-1}}^2$
$\hat{d}_{k+1} = \frac{\|s_{k+1}\|_{A_{k+1}^{-1}}^2 / (1 - \beta_2) - r_{k+1}}{\|s_{k+1}\|_1}$
$d_{k+1} = \max\left(d_k, \hat{d}_{k+1}\right)$
end for

6. Machine Learning Applications

It is straightforward to adapt the D-Adaptation technique to stochastic optimization, although the theory no longer directly supports this case. Algorithm 3 and 4 are versions of D-Adaptation for SGD and Adam respectively. Both of the two methods solve the stochastic optimization problem,

$$\min_{x \in \mathbb{R}^p} \mathbb{E}[f(x, \xi)]$$

using stochastic subgradients $g_k \in \partial f(x_k, \xi_k)$.

Compared to Algorithm 1, we remove the factor of 2 from the D bound in Algorithms 3 and 4. This improves the practical performance of the method, and is allowed by the theory, as it is equivalent to multiplying the step size by 2 everywhere. For Adam, further modifications are needed:

- The norms are now weighted instead of unweighted.
- Since s_k is now updated by an exponential moving average, a correction factor of $(1 - \beta_2)$ in the D bound is needed to keep everything at the same scale.
- No bias correction is included as it doesn't appear necessary based on our experiments. The implicit learning rate warm-up of D-Adaptation has a similar effect.

We include an optional γ_k constant sequence as input to the algorithms. This sequence should be set following a learning rate schedule if one is needed for the problem. This schedule should consider 1.0 as the base value, increase towards 1.0 during warm-up (if needed), and decrease from 1 during learning rate annealing. Typically the same schedule can be used as would normally be used without D-Adaptation.

7. Experimental Results

We compared our D-Adapted variants of Adam and SGD on a range of machine learning problems to demonstrate their effectiveness in practice. For the deep learning problems, we varied both the models and datasets to illustrate the effectiveness of D-Adaptation across a wide range of situations. In each case we used the standard learning rate schedule typically used for the problem, with the *base* learning rate set by D-Adaptation. Full hyper-parameter settings for each problem are included in the Appendix. We plot the mean of multiple seeds, with the error bars in each plot indicating a range of 2 standard errors from the mean. The number of seeds used for each problem is listed in the Appendix.

7.1. Convex Problems

For our convex experiments, we considered logistic regression applied to 5 commonly used benchmark problems from the LIBSVM repository. In each case, we consider 100 epochs of training, with a stage-wise schedule with 10-fold decreases at 60, 80, and 95 epochs. No weight decay was used, and batch-size 16 was applied for each problem. All other hyper-parameters were set to their defaults. The learning rate for Adam was chosen as the value that gave the highest accuracy after a grid search. D-Adaptation matches or exceeds the performance of a grid-search based learning rate on all 5 problems, to within 0.1% accuracy.

7.2. Convolutional Image Classification

For a convolutional image classification benchmark, we used the three most common datasets used for optimization method testing: CIFAR10, CIFAR100 [17] and ImageNet 2012 [27]. We varied the architectures to show the flexibility of D-Adaptation, using a Wide-Resnet [36], a DenseNet [15] and a vanilla ResNet model [14] respectively. D-Adaptation matches or exceeds the baseline learning rates on each problem.

7.3. LSTM Recurrent Neural Networks

The IWSLT14 German-to-English dataset [3] is a common choice for benchmarking machine translation models. We trained an LSTM model [33] commonly used for this problem. The standard training procedure includes an inverse-square-root learning rate schedule, which we used for both the baseline and for D-Adaptation. Our model achieves comparable performance to the baseline training regimen without any need to tune the learning rate.

7.4. Masked Language Modelling

Bidirectional Encoder Representations from Transformers (BERT) is a popular approach to pretraining transformer models [6]. We use the 110M parameter RoBERTA variant [18] of BERT for our experiments. This model size provides a large and realistic test problem for D-Adaptation. We

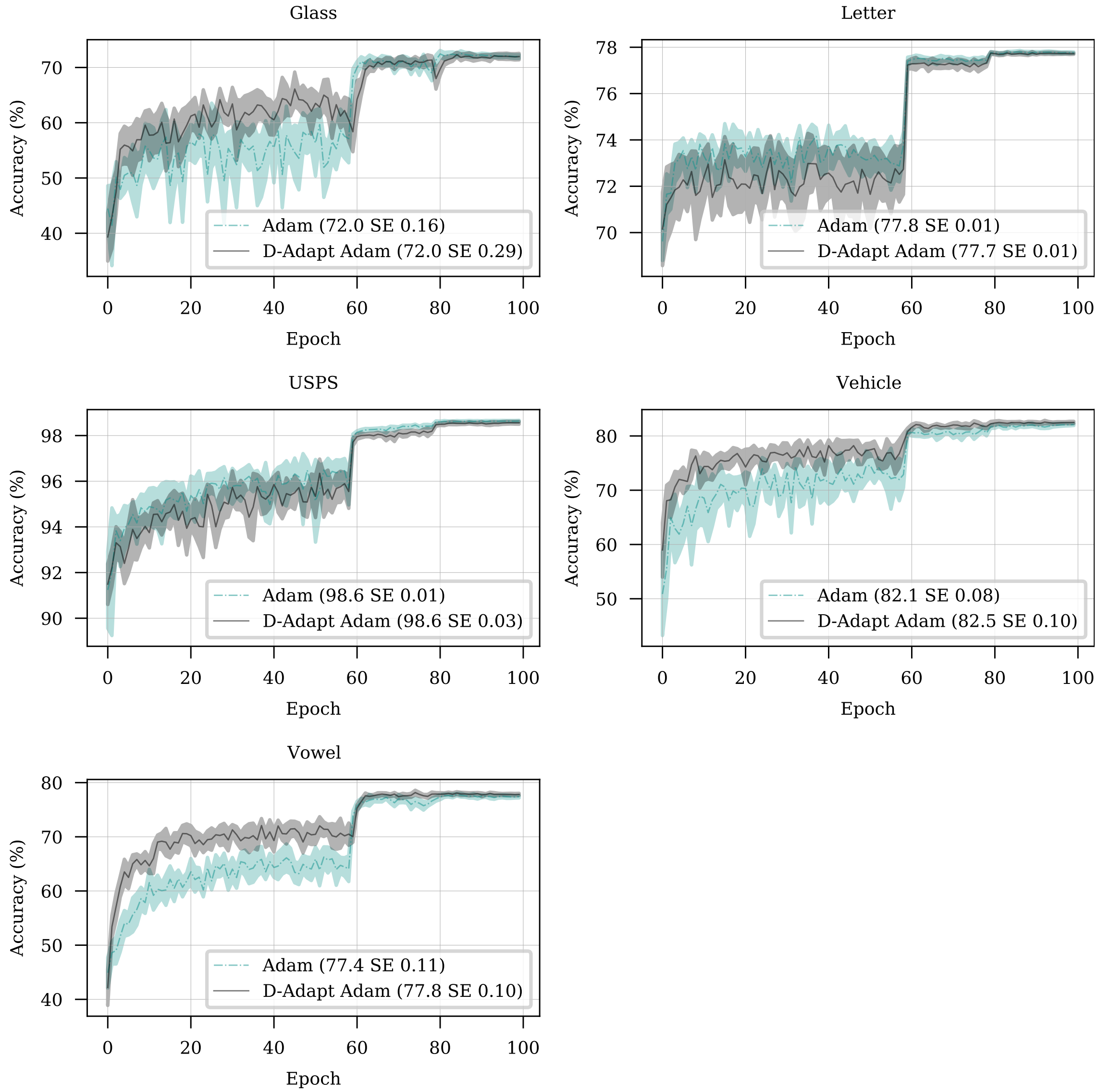

Figure 2: Logistic Regression experiments.

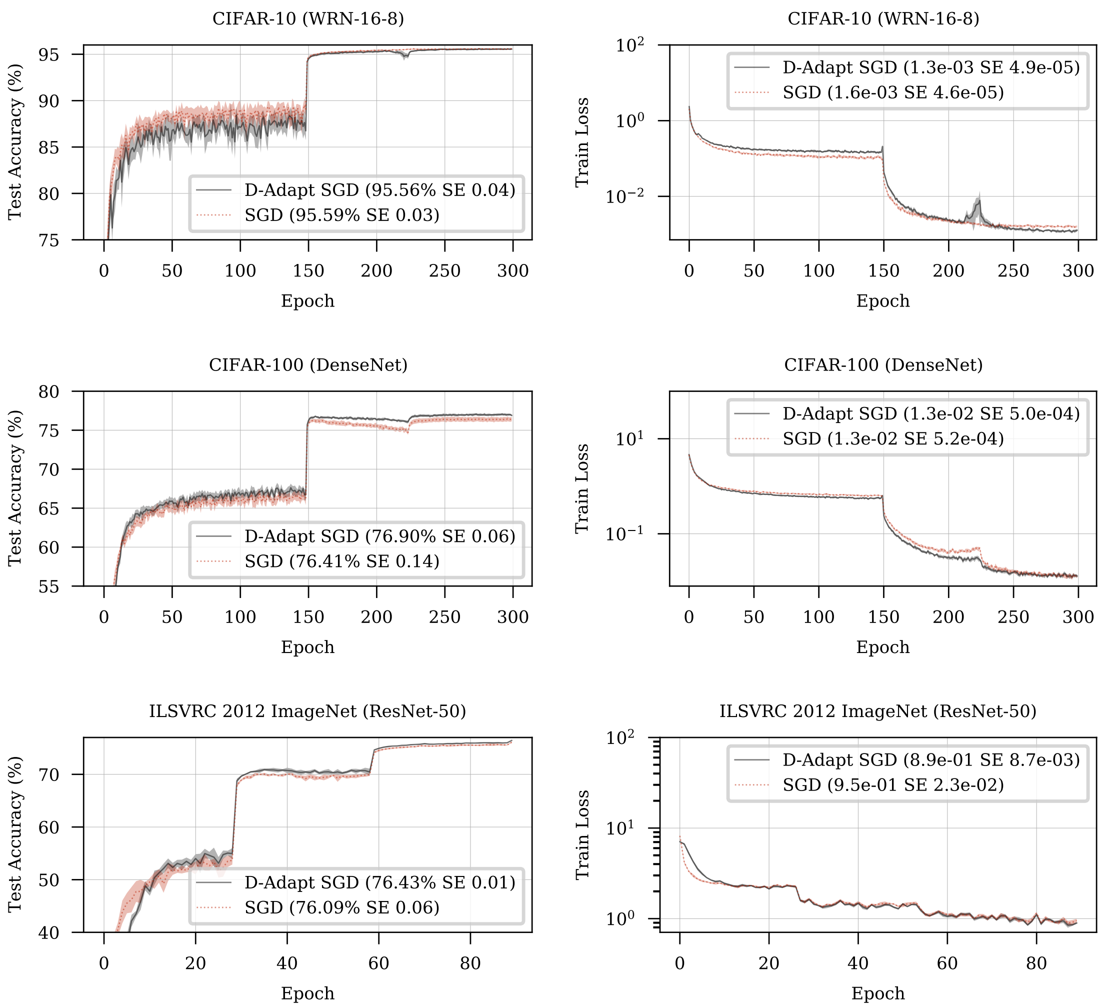

Figure 3: Image Classification experiments.

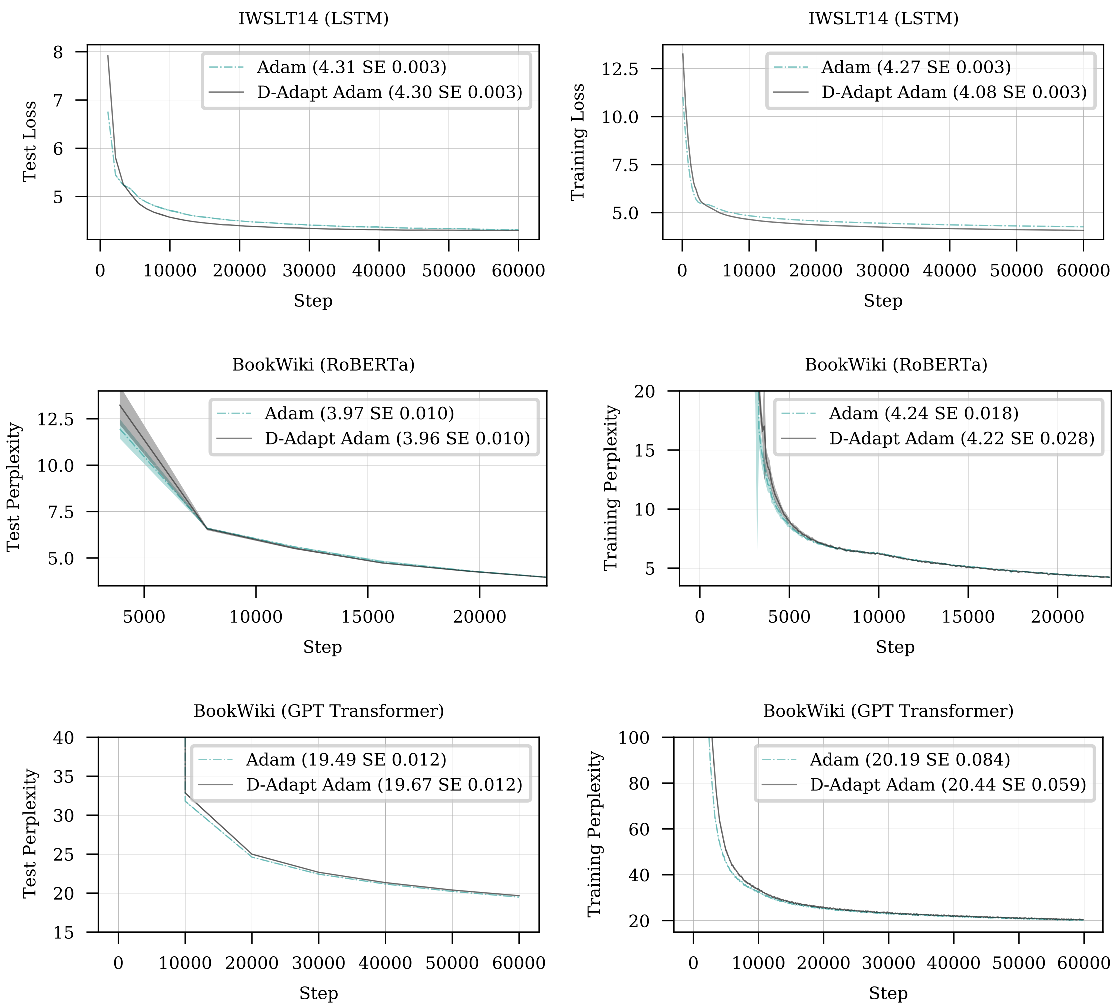

Figure 4: Natural Language Processing experiments.

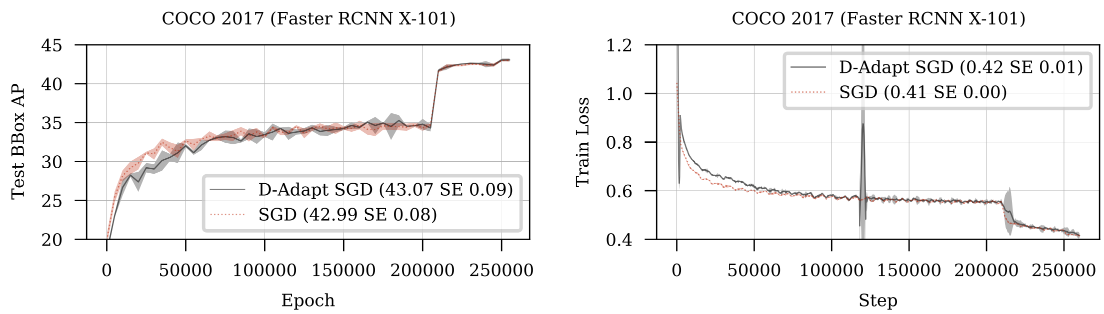

Figure 5: A Faster RCNN object detector trained on COCO 2017.

train on the Book-Wiki corpus (combining books from Zhu et al. [38] and a snapshot of Wikipedia). D-Adaptation again matches the baseline in test-set perplexity.

7.5. Auto-regressive Language Modelling

For our experiments on auto-regressive language modelling, we used the original GPT decoder-only transformer architecture [25]. This model is small enough to train on a single machine, unlike the larger GPT-2/3 models. Its architecture is representative of other large language models. We trained on the large Book-Wiki corpus. D-Adaptation is comparable to the baseline with only a negligible perplexity difference.

7.6. Object Detection

The COCO 2017 object detection task is a popular benchmark in computer vision. We trained as Faster-RCNN [26] model as implemented in Detectron2 [34]. For the backbone model, we used a pretrained ResNeXt-101-32x8d [35], the largest model available in Detectron2 for this purpose. Our initial experiments showed D-Adaptation overfitting. We identified that the default decay of 0.0001 in the code-base was not optimized for this backbone model, and increasing it to 0.00015 improved the test set accuracy for both the baseline (42.67 to 42.99) and D-adapted versions (41.92 to 43.07), matching the published result of 43 for this problem.

7.7. Vision Transformers

Vision transformers [8] are a recently developed approach to image classification that differ significantly from the image classification approaches in Section 7.2. They are closer to the state-of-the-art than ResNet models, and require significantly more resources to train to high accuracy. Vision Transformers continue to improve past the 90 epochs traditionally used for ResNet models, and 300 epochs of training is the standard. Vision transformers require adaptive optimizers such as Adam to train, and avoid the overfitting problem seen when using Adam on ResNet models by using multiple additional types of regularization. We use the `vit_tiny_patch16_224` model in the *PyTorch Image Models* framework [32] as it is small enough to train on 8 GPUs. The standard training pipeline uses a cosine learning rate schedule.

This is an example of a situation where D-Adaptation under-performs the baseline learning rate. After careful examination, we believe the cosine learning rate schedule may be causing this issue.

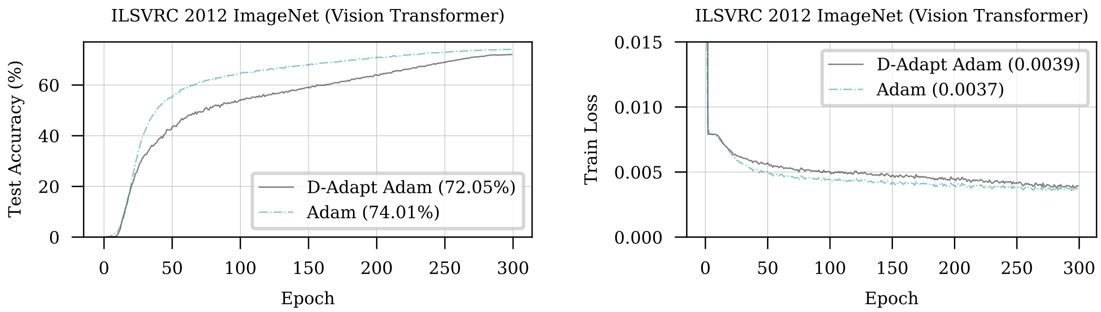

Figure 6: A vision transformer trained on ImageNet.

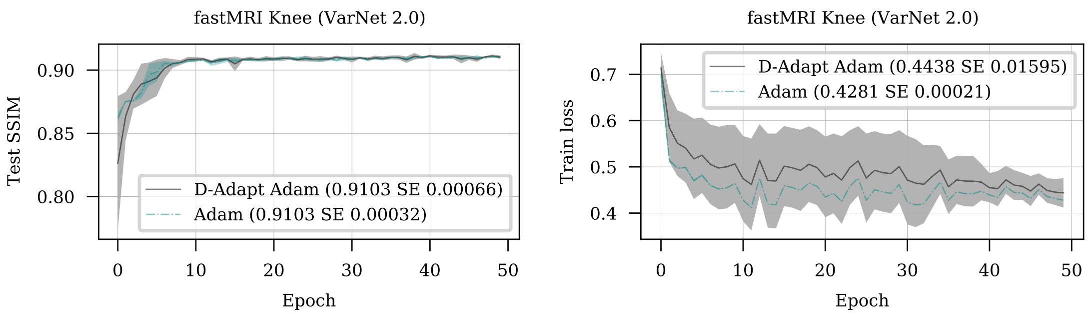

Figure 7: VarNet 2.0 model trained on the fastMRI Knee dataset.

D-Adaptation chooses a learning rate approximately twice the standard rate used for Adam on this problem. The cosine schedule decreases the learning rate less aggressively than other schedules early on, which may explain the performance gap.

7.8. fastMRI

The fastMRI Knee Dataset [37] is a large-scale release of raw MRI data. The reconstruction task consists of producing a 2-dimensional, grey-scale image of the anatomy from the raw sensor data, under varying under-sampling regimes. We trained a VarNet 2.0 [28] model, a strong baseline model on this dataset, using the code and training setup released by Meta [5, 16]. We again match the highly tuned baseline learning rate with D-Adaptation.

7.9. Recommendation Systems

The Criteo Kaggle Display Advertising dataset[2] is a large, sparse dataset of user click-through events. The DLRM [19] model is a common benchmark for this problem, representative of personalization and recommendation systems used in industry. Our method closely matches the performance of the tuned baseline learning rate.

2. https://www.kaggle.com/c/criteo-display-ad-challenge

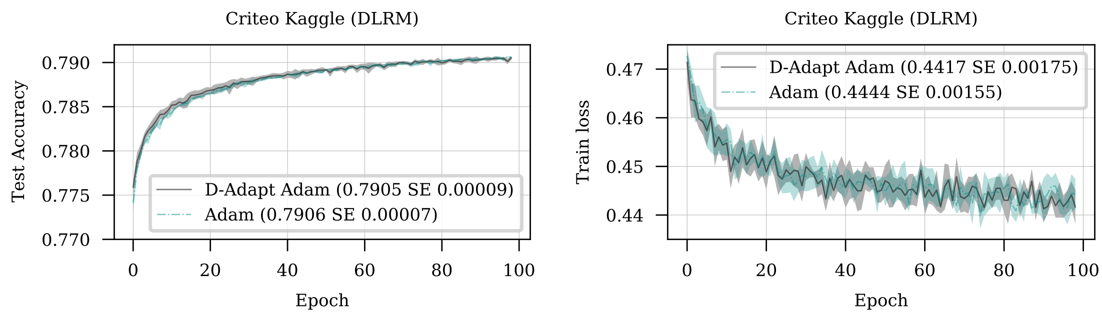

Figure 8: DLRM recommendation model on the Criteo Click-Through-Rate prediction problem.

8. Conclusion

We have presented a simple approach to achieving parameter free learning of convex Lipshitz functions, by constructing successively better lower bounds on the key unknown quantity: the distance to solution $\|x_0 - x_*\|$. Our approach for constructing these lower bounds may be of independent interest. Our method is also highly practical, demonstrating excellent performance across a range of large and diverse machine learning problems.

Acknowledgements

We would like to thank Ashok Cutkosky for suggesting substantial simplifications to the proof of Lemma 5.

References

[1] Yoshua Bengio. Gradient-based optimization of hyperparameters. In *Neural Computation*, 2000.

[2] Yair Carmon and Oliver Hinder. Making SGD parameter-free. *arXiv preprint arXiv:2205.02160*, 2022.

[3] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *IWSLT*, 2014.

[4] Kartik Chandra, Audrey Xie, Jonathan Ragan-Kelley, and Erik Meijer. Gradient descent: The ultimate optimizer. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

[5] Aaron Defazio. Offset sampling improves deep learning based accelerated mri reconstructions by exploiting symmetry. *arXiv preprint arXiv:1912.01101*, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Lingustics*, 2019.

[7] Justin Domke. Generic methods for optimization-based modeling. In *Fifteenth International Conference on Artificial Intelligence and Statistics*, 2012.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[9] Yoel Drori and Adrien B. Taylor. Efficient first-order methods for convex minimization: a constructive approach. *Mathematical Programming*, 2020.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[11] Matthias Feurer and Frank Hutter. *Automated Machine Learning*, chapter Hyperparameter Optimization. Springer International Publishing, 2019.

[12] Baptiste Goujaud, Adrien Taylor, and Aymeric Dieuleveut. Optimal first-order methods for convex functions with a quadratic upper bound. Technical report, INRIA, 2022.

[13] Elad Hazan and Sham M. Kakade. Revisiting the polyak step size. Technical report, Google AI Princeton, 2019.

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

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. doi: 10.1109/CVPR.2017.243.

[16] Florian Knoll, Jure Zbontar, Anuroop Sriram, Matthew J. Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzalv, Adriana Romero, Michael Rabbat, Pascal Vincent, James Pinkerton, Duo Wang, Nafissa Yakubova, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1):e190007, 2020. doi: 10.1148/ryai.2020190007. URL https://doi.org/10.1148/ryai.2020190007. PMID: 32076662.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[19] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019. URL `https://arxiv.org/abs/1906.00091`.

[20] Yurii Nesterov. *Lectures on Convex Optimization*. Springer Nature, 2018.

[21] Francesco Orabona. A modern introduction to online learning. Technical report, Boston University, 2019.

[22] Francesco Orabona and Tatiana Tommasi. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[23] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746. PMLR, 2016.

[24] Boris T. Polyak. *Introduction to optimization*. Optimization Software, Inc., 1987.

[25] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2019.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[28] Anuroop Sriram, Jure Zbontar, Tullie Murrell, Aaron Defazio, C. Lawrence Zitnick, Nafissa Yakubova, Florian Knoll, and Patricia Johnson. End-to-end variational networks for accelerated MRI reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 64–73. Springer, 2020.

[29] Matthew Streeter and H. Brendan McMahan. Less regret via online conditioning, 2010. URL `https://arxiv.org/abs/1002.4862`.

[30] Matthew Streeter and H. Brendan McMahan. No-regret algorithms for unconstrained online convex optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, Red Hook, NY, USA, 2012. Curran Associates Inc.

[31] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686, 2019.

[32] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[33] Sam Wiseman and Alexander M. Rush. 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, 2016.

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. doi: 10.1109/CVPR.2017.634.

[36] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

[37] Jure Zbontar, Florian Knoll, Anuroop Sriram, Matthew J. Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, et al. fastMRI: An open dataset and benchmarks for accelerated MRI. *arXiv preprint arXiv:1811.08839*, 2018.

[38] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. doi: 10.1109/ICCV.2015.11. URL https://doi.org/10.1109/ICCV.2015.11.

Appendix A. Core Theory

Here, we are going to consider a more general form of Algorithm 1 with arbitrary positive weights λ_k that do not have to be equal to d_k. In particular, we will study the update rule

$$s_{n+1} = s_n + \lambda_n g_n \qquad \text{and} \qquad \hat{d}_{n+1} = \frac{\gamma_{n+1}\|s_{n+1}\|^2 - \sum_{k=0}^{n} \gamma_k \lambda_k^2 \|g_k\|^2}{2\|s_{n+1}\|}.$$

Later in the proofs, we will set $\lambda_k = d_k$, but most intermediate results are applicable with other choices of λ_k as well.

Lemma 4 *The inner product $\gamma_k \lambda_k \langle g_k, s_k \rangle$ is a key quantity that occurs in our theory. We can bound the sum of these inner products over time by considering the following expansion:*

$$-\sum_{k=0}^{n} \gamma_k \lambda_k \langle g_k, s_k \rangle = -\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \|g_k\|^2 + \frac{1}{2} \sum_{k=0}^{n} (\gamma_{k+1} - \gamma_k) \|s_{k+1}\|^2 .$$

This simplifies when the weighting sequence is flat:

$$-\gamma_{n+1} \sum_{k=0}^{n} \langle g_k, s_k \rangle = -\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 + \frac{\gamma_{n+1}}{2} \sum_{k=0}^{n} \|g_k\|^2 ,$$

with λ weights:

$$-\gamma_{n+1} \sum_{k=0}^{n} \lambda_k \langle g_k, s_k \rangle = -\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 + \frac{\gamma_{n+1}}{2} \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2 .$$

Proof This is straightforward to show by induction (it's a consequence of standard DA proof techniques, where $\|s_n\|^2$ is expanded).

$$\begin{aligned}
\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 &= \frac{\gamma_n}{2} \|s_{n+1}\|^2 + \frac{1}{2} (\gamma_{n+1} - \gamma_n) \|s_{n+1}\|^2 \\
&= \frac{\gamma_n}{2} \|s_n\|^2 + \gamma_n \lambda_n \langle g_n, s_n \rangle + \frac{\gamma_n}{2} \lambda_n^2 \|g_n\|^2 + \frac{1}{2} (\gamma_{n+1} - \gamma_n) \|s_{n+1}\|^2 .
\end{aligned}$$

Therefore

$$-\gamma_n \lambda_n \langle g_n, s_n \rangle = \frac{\gamma_n}{2} \|s_n\|^2 - \frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 + \frac{\gamma_n}{2} \lambda_n^2 \|g_n\|^2 + \frac{1}{2} (\gamma_{n+1} - \gamma_n) \|s_{n+1}\|^2 .$$

Telescoping

$$-\sum_{k=0}^{n} \gamma_k \lambda_k \langle g_k, s_k \rangle = -\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \|g_k\|^2 + \frac{1}{2} \sum_{k=0}^{n} (\gamma_{k+1} - \gamma_k) \|s_{k+1}\|^2 .$$

■

Lemma 5 *The iterates of Algorithm 1 satisfy*

$$\sum_{k=0}^{n} \lambda_k (f(x_k) - f_*) \leq \|x_0 - x_*\| \|s_{n+1}\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \|g_k\|^2 - \frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 .$$

Proof Starting from convexity:

$$\begin{aligned}
\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) &\leq \sum_{k=0}^{n} \lambda_k \left\langle g_k, x_k - x_* \right\rangle \\
&= \sum_{k=0}^{n} \lambda_k \left\langle g_k, x_k - x_0 + x_0 - x_* \right\rangle \\
&= \left\langle s_{n+1}, x_0 - x_* \right\rangle + \sum_{k=0}^{n} \lambda_k \left\langle g_k, x_k - x_0 \right\rangle \\
&= \left\langle s_{n+1}, x_0 - x_* \right\rangle - \sum_{k=0}^{n} \lambda_k \gamma_k \left\langle g_k, s_k \right\rangle \\
&\leq \left\| s_{n+1} \right\| \left\| x_0 - x_* \right\| - \sum_{k=0}^{n} \lambda_k \gamma_k \left\langle g_k, s_k \right\rangle .
\end{aligned}$$

We can further simplify with:

$$-\sum_{k=0}^{n} \gamma_k \lambda_k \left\langle g_k, s_k \right\rangle = -\frac{\gamma_{n+1}}{2} \left\| s_{n+1} \right\|^2 + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \left\| g_k \right\|^2 + \frac{1}{2} \sum_{k=0}^{n} \left(\gamma_{k+1} - \gamma_k\right) \left\| s_{k+1} \right\|^2 .$$

Using the fact that $\gamma_{k+1} - \gamma_k \leq 0$ we have:

$$\begin{aligned}
\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) &\leq \left\| x_0 - x_* \right\| \left\| s_{n+1} \right\| - \sum_{k=0}^{n} \gamma_k \lambda_k \left\langle g_k, s_k \right\rangle \\
&\leq \left\| x_0 - x_* \right\| \left\| s_{n+1} \right\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \left\| g_k \right\|^2 - \frac{\gamma_{n+1}}{2} \left\| s_{n+1} \right\|^2 .
\end{aligned}$$

∎

Theorem 6 *The initial distance to solution, $D = \|x_0 - x_*\|$, can be lower bounded as follows*

$$D \geq \hat{d}_{n+1} = \frac{\gamma_{n+1} \left\| s_{n+1} \right\|^2 - \sum_{k=0}^{n} \gamma_k \lambda_k^2 \left\| g_k \right\|^2}{2 \left\| s_{n+1} \right\|}.$$

Proof The key idea is that the bound in Lemma 5,

$$\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) \leq D \left\| s_{n+1} \right\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \left\| g_k \right\|^2 - \frac{\gamma_{n+1}}{2} \left\| s_{n+1} \right\|^2 ,$$

gives some indication as to the magnitude of D in the case when the other terms on the right are negative. To proceed, we use $\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) \geq 0$, giving:

$$0 \leq D \left\| s_{n+1} \right\| + \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \left\| g_k \right\|^2 - \frac{\gamma_{n+1}}{2} \left\| s_{n+1} \right\|^2 ,$$

which we can rearrange to:

$$D\left\|s_{n+1}\right\| \geq \frac{\gamma_{n+1}}{2}\left\|s_{n+1}\right\|^2-\sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2\left\|g_k\right\|^2 .$$

Therefore:

$$D \geq \frac{\frac{\gamma_{n+1}}{2}\left\|s_{n+1}\right\|^2-\sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2\left\|g_k\right\|^2}{\left\|s_{n+1}\right\|}.$$

■

Lemma 7 *The norm of s_{n+1} is bounded by:*

$$\left\|s_{n+1}\right\| \leq \frac{2 d_{n+1}}{\gamma_{n+1}}+\frac{\sum_{k=0}^{n} \gamma_k \lambda_k^2\|g_k\|^2}{2 d_{n+1}}. \tag{2}$$

Proof Using the definition of $\hat{d}_{n+1}$ from Theorem 6, and the property $\hat{d}_{n+1} \leq d_{n+1}$, we derive

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

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

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

Rearranging the terms, we obtain

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

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

Proposition 8 *(From Streeter and McMahan [29]) The gradient error term can be bounded as:*

$$\sum_{k=0}^{n} \frac{\left\|g_k\right\|^2}{\sqrt{G^2+\sum_{i=0}^{k-1}\left\|g_i\right\|^2}} \leq 2 \sqrt{\sum_{k=0}^{n}\left\|g_k\right\|^2}. \tag{3}$$

Moreover, if $\gamma_k=\frac{1}{\sqrt{G^2+\sum_{i=0}^{k-1}\left\|g_i\right\|^2}}$, then

$$\sum_{k=0}^{n} \frac{\gamma_k}{2}\left\|g_k\right\|^2 \leq \gamma_{n+1}\left(G^2+\sum_{k=0}^{n}\left\|g_k\right\|^2\right). \tag{4}$$

Lemma 9 *It holds for Algorithm 1:*

$$\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \leq 2Dd_{n+1}\sqrt{\sum_{k=0}^{n} \|g_k\|^2} + Dd_{n+1}\sum_{k=0}^{n}\gamma_k\|g_k\|^2.$$

Proof First, recall the key bound from Lemma 5:

$$\begin{aligned}\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) &\leq D\left\|s_{n+1}\right\| - \frac{\gamma_{n+1}}{2}\left\|s_{n+1}\right\|^2 + \sum_{k=0}^{n}\frac{\gamma_k}{2}\lambda_k^2\left\|g_k\right\|^2 \\ &\leq D\left\|s_{n+1}\right\| + \sum_{k=0}^{n}\frac{\gamma_k}{2}\lambda_k^2\left\|g_k\right\|^2.\end{aligned}$$

Now let us apply the bound from Lemma 7:

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

which gives

$$\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) \leq \frac{2Dd_{n+1}}{\gamma_{n+1}} + \frac{D\sum_{k=0}^{n}\gamma_k\lambda_k^2\|g_k\|^2}{2d_{n+1}} + \sum_{k=0}^{n}\frac{\gamma_k}{2}\lambda_k^2\left\|g_k\right\|^2.$$

Using $\lambda_k = d_k \leq d_{n+1} \leq D$ and plugging in the step size, we obtain

$$\begin{aligned}\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) &\leq \frac{2Dd_{n+1}}{\gamma_{n+1}} + \frac{D\sum_{k=0}^{n}\gamma_k d_{n+1}^2\|g_k\|^2}{2d_{n+1}} + \sum_{k=0}^{n}\frac{\gamma_k}{2}d_{n+1}^2\left\|g_k\right\|^2 \\ &\leq 2Dd_{n+1}\sqrt{\sum_{k=0}^{n}\|g_k\|^2} + \frac{1}{2}Dd_{n+1}\sum_{k=0}^{n}\gamma_k\|g_k\|^2 + \frac{1}{2}Dd_{n+1}\sum_{k=0}^{n}\gamma_k\left\|g_k\right\|^2 \\ &= 2Dd_{n+1}\sqrt{\sum_{k=0}^{n}\|g_k\|^2} + Dd_{n+1}\sum_{k=0}^{n}\gamma_k\|g_k\|^2.\end{aligned}$$

This is exactly our result. ■

Theorem 10 *The average iterate $\hat{x}_n$ returned by Algorithm 1 satisfies:*

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

Proof In the case where $g_0 = 0$, $f(x_0) = f(x_*)$ and the theorem is trivially true, so we assume that $\|g_0\|^2 > 0$. We will show the result holds for some n, where we choose n sufficiently large so that a number of criteria are met:

Criterion 1: since d_k is a non-decreasing sequence upper bounded by D, there must exist some $\hat{n}$ such that after $\hat{n}$ steps, $d_k \geq \frac{1}{2}d_{n+1}$ for all $k, n \geq \hat{n}$. We take $n \geq 2\hat{n}$.

Criterion 2: since we assume the bound $\|g_k\|^2 \le G^2$, there must exist some r such that $\|g_n\|^2 \le \sum_{k=0}^{n-1} \|g_k\|^2$ for all $n \ge r$. Let us choose the smallest r that satisfies this condition, in which case $\|g_{r-1}\|^2 \ge \sum_{k=0}^{r-2} \|g_k\|^2$, otherwise we could have chosen $r-1$. Moreover, we have by definition $\gamma_k \le \frac{1}{\|g_0\|}$ for all $k \le r-1$. Combining this with the first bound from Proposition 8, we derive

$$\begin{aligned}
\sum_{k=0}^{n} \gamma_k \|g_k\|^2 &= \sum_{k=r}^{n} \gamma_k \|g_k\|^2 + \sum_{k=0}^{r-1} \gamma_k \|g_k\|^2 \\
&\le 2\sqrt{\sum_{k=r}^{n} \|g_k\|^2} + \frac{1}{\|g_0\|} \sum_{k=0}^{r-1} \|g_k\|^2 \\
&\le 2\sqrt{\sum_{k=0}^{n} \|g_k\|^2} + \frac{2}{\|g_0\|} \|g_{r-1}\|^2 \\
&\le 2\sqrt{\sum_{k=0}^{n} \|g_k\|^2} + 2\frac{G^2}{\|g_0\|}.
\end{aligned}$$

We continue with the bound from Lemma 9:

$$\sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \le 2Dd_{n+1} \sqrt{\sum_{k=0}^{n} \|g_k\|^2} + Dd_{n+1} \sum_{k=0}^{n} \gamma_k \|g_k\|^2.$$

From Criterion 1, we have that:

$$\sum_{k=0}^{n} d_k \ge \sum_{k=\hat{n}}^{n} d_k \ge \sum_{k=\hat{n}}^{n} \frac{1}{2} d_{n+1} = \frac{1}{2}(n - \hat{n} + 1) d_{n+1} \ge \frac{1}{4}(n+1) d_{n+1},$$

hence

$$\frac{1}{\sum_{k=0}^{n} d_k} \le \frac{4}{(n+1)d_{n+1}}.$$

Plugging this back yields

$$\frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \le \frac{8D}{(n+1)} \sqrt{\sum_{k=0}^{n} \|g_k\|^2} + \frac{4D}{n+1} \sum_{k=0}^{n} \gamma_k \|g_k\|^2.$$

Using the bound obtained from Criterion 2, we further get

$$\frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \le \frac{8D}{(n+1)} \sqrt{\sum_{k=0}^{n} \|g_k\|^2} + \frac{4D}{n+1} \left(2\sqrt{\sum_{k=0}^{n} \|g_k\|^2} + 2\frac{G^2}{\|g_0\|} \right).$$

Using $\|g_k\|^2 \le G^2$, we simplify this to

$$\frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k \left(f(x_k) - f_*\right) \le \frac{16DG}{\sqrt{n+1}} + \frac{8DG^2}{(n+1)\|g_0\|}.$$

Using Jensen's inequality, we can convert this to a bound on the average iterate defined as

$$\hat{x}_n = \frac{1}{\sum_{k=0}^{n} d_k} \sum_{k=0}^{n} d_k x_k,$$

implying

$$f(\hat{x}_n) - f_* \leq \frac{12DG}{\sqrt{n+1}} + \frac{8DG^2}{(n+1)\|g_0\|}.$$

Note that the second term on the right decreases faster than the first term with respect to n, so

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

■

Appendix B. Non-asymptotic analysis

Lemma 11 *Consider a sequence $d_0, \ldots d_{N+1}$, where for each k, $d_{k+1} \geq d_k$. Assume that $N \geq \log_2(d_N/d_0)$, then*

$$\min_{n \leq N} \frac{d_{n+1}}{\sum_{k=0}^{n} d_k} \leq 2\frac{\log_2(d_{N+1}/d_0)}{N+1}. \tag{5}$$

Let $r = \lceil \log_2(d_N/d_0) \rceil$. We proceed by an inductive argument on r. In the base case, if $r = 0$ then the result follows immediately:

$$\begin{aligned}\min_{n \leq N} \frac{d_{n+1}}{\sum_{k=0}^{n} d_k} &= \frac{d_{N+1}}{\sum_{k=0}^{N} d_k} = \frac{d_{N+1}}{(N+1)d_{N+1}} \\ &= \frac{1}{N+1} \leq 2\frac{\log_2(d_{N+1}/d_0)}{N+1}.\end{aligned}$$

So assume that $r > 0$. First we show that no induction is needed, and we may take $n = N$, if

$$d_k \geq \frac{1}{2} d_{N+1}, \quad \text{for all } k \geq \left\lfloor N + 1 - \frac{N+1}{\log_2(d_{N+1}/d_0)} \right\rfloor.$$

Since, in that case we have:

$$\begin{aligned}\sum_{k=0}^{N} d_k &\geq \sum_{k=\lfloor N+1-(N+1)/\log_2(d_{N+1}/d_0)\rfloor}^{N} d_k \geq \frac{1}{2}\left(N + 1 - \left\lfloor N + 1 - \frac{N+1}{\log_2(d_N/d_0)} \right\rfloor\right) d_{N+1} \\ &\geq \frac{1}{2} \frac{(N+1)\, d_{N+1}}{\log_2(d_{N+1}/d_0)}.\end{aligned}$$

Rearranging this bound gives:

$$\frac{d_{N+1}}{\sum_{k=0}^{N} d_k} \leq 2\frac{\log_2(d_{N+1}/d_0)}{N+1},$$

and therefore

$$\min_{n \le N} \frac{d_{n+1}}{\sum_{k=0}^{n} d_k} \le 2 \frac{\log_2(d_{N+1}/d_0)}{N+1}.$$

So, instead suppose that $d_{n'+1} \le \frac{1}{2} d_{N+1}$, for $n' = \left\lfloor N + 1 - \frac{(N+1)}{\log_2(d_{N+1}/d_0)} \right\rfloor$. Note that $+1$ is due to the fact that the above case includes the edge case where an increase occurs exactly at the beginning of the interval. Assume the inductive hypothesis that:

$$\min_{n \le n'} \frac{d_{n+1}}{\sum_{k=0}^{n} d_k} \le 2 \frac{\log_2(d_{n'+1}/d_0)}{n'+1}, \quad \text{for } n' = \left\lfloor N + 1 - \frac{(N+1)}{\log_2(d_{N+1}/d_0)} \right\rfloor.$$

Under this inductive hypothesis assumption, we note that:

$$\begin{aligned} \frac{\log_2(d_{n'+1}/d_0)}{n'+1} &\le \frac{1}{\left\lfloor N + 1 - \frac{(N+1)}{\log_2(d_{N+1}/d_0)} \right\rfloor + 1} \log_2(d_{n'+1}/d_0) \\ &\le \frac{1}{N - \frac{(N+1)}{\log_2(d_{N+1}/d_0)} + 1} \log_2(d_{n'+1}/d_0) \\ &= \frac{\log_2(d_{N+1}/d_0)}{(N+1)\left(\log_2(d_{N+1}/d_0) - 1\right)} \log_2(d_{n'+1}/d_0) \\ &= \frac{\log_2(d_{N+1}/d_0)}{N+1} \cdot \frac{\log_2(d_{n'+1}/d_0)}{\log(d_{N+1}/d_0) - 1} \\ &\le \frac{\log_2(d_{N+1}/d_0)}{N+1}, \end{aligned}$$

where the last inequality follows from $d_{n'} \le \frac{1}{2} d_{N+1}$, as it implies that:

$$\log_2(d_{n'+1}/d_0) \le \log_2\left(\frac{1}{2} d_{N'+1}/d_0\right) = \log_2(d_{N+1}/d_0) - 1.$$

Putting it all together, we have that:

$$\min_{n \le N} \frac{d_{n+1}}{\sum_{k=0}^{n} d_k} \le \left[\frac{d_{n+1}}{\sum_{k=0}^{n} d_k}\right]_{n = N - \frac{(N+1)}{\log_2(d_N/d_0)}} \le 2 \frac{\log_2(d_{N+1}/d_0)}{N+1}.$$

Theorem 12 *Consider Algorithm 1 run for n steps, where $n \ge \log_2(D/d_0)$, if we return the point $\hat{x}_t = \frac{1}{\sum_{k=0}^{t} d_k} \sum_{k=0}^{t} d_k x_k$ where t is chosen to be:*

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

Then:

$$f(\hat{x}_t) - f_* \le 8 \frac{\log_2(D/d_0)}{n+1} D \sqrt{\sum_{k=0}^{t} \|g_k\|^2}.$$

Proof Consider the bound from Lemma 9:

$$\begin{aligned}
\frac{1}{\sum_{k=0}^{n} d_k}\sum_{k=0}^{n} d_k\left(f(x_k)-f_*\right) &\le \frac{2Dd_{n+1}}{\sum_{k=0}^{n} d_k}\sqrt{\sum_{k=0}^{n}\|g_k\|^2}+\frac{Dd_{n+1}}{\sum_{k=0}^{n} d_k}\sum_{k=0}^{n}\gamma_k\|g_k\|^2 \\
&\overset{(3)}{\le} \frac{2Dd_{n+1}}{\sum_{k=0}^{n} d_k}\sqrt{\sum_{k=0}^{n}\|g_k\|^2}+\frac{Dd_{n+1}}{\sum_{k=0}^{n} d_k}2\sqrt{\sum_{k=0}^{n}\|g_k\|^2} \\
&= \frac{4Dd_{n+1}}{\sum_{k=0}^{n} d_k}\sqrt{\sum_{k=0}^{n}\|g_k\|^2}.
\end{aligned}$$

Now using Lemma 11, we can return the point $\hat{x}_t$ and at time $t=\arg\min_{k\le n}\frac{d_{k+1}}{\sum_{i=0}^{k} d_i}$, ensuring that

$$\frac{d_{t+1}}{\sum_{k=0}^{t} d_k}=\min_{k\le n}\frac{d_{k+1}}{\sum_{i=0}^{k} d_i}\overset{(5)}{\le} 2\frac{\log_2(d_{n+1}/d_0)}{n+1},$$

giving us an upper bound:

$$f(\hat{x}_t)-f_*\le 8\frac{\log_2(D/d_0)}{n+1}D\sqrt{\sum_{k=0}^{t}\|g_k\|^2}.$$

∎

We note that a similar proof can be used to remove the G^2 term from the numerator of γ_k. To this end, we could reuse the bound obtained in the proof of Theorem 10:

$$\sum_{k=0}^{n}\gamma_k\left\|g_k\right\|^2\le 2\sqrt{\sum_{k=0}^{n}\left\|g_k\right\|^2}+2\frac{G^2}{\|g_0\|},$$

which holds for $\gamma_k=\frac{1}{\sqrt{\sum_{i=0}^{k-1}\|g_i\|^2}}$. In the proof of Theorem 10, this bound was stated for $n\ge r$, where r is the smallest number such that $\left\|g_k\right\|^2\le\sum_{i=0}^{k-1}\left\|g_i\right\|^2$ for all $k\ge r$. However, the bound itself does not require $n\ge r$, since for $n<r$ it holds even without the first term in the right-hand side. The second term in that bound does not increase with n, and it would result in the following bound for the same iterate $\hat{x}_t$ as in Theorem 12:

$$f(\hat{x}_t)-f_*\le\frac{8DG\log_2(D/d_0)}{\sqrt{n+1}}+\frac{4DG^2\log_2(D/d_0)}{(n+1)\|g_0\|}.$$

Since the leading term in the bound above is of order $\mathcal{O}\left(\frac{1}{\sqrt{n+1}}\right)$, the extra term for not using G is negligible.

Appendix C. Coordinate-wise setting

In the coordinate-wise setting we define the matrices A_{n+1} as diagonal matrices with diagonal elements a_i at step n defined as

$$a_{(n+1)i} = \sqrt{G_\infty^2 + \sum_{k=0}^{n} g_{ki}^2}.$$

Let p be the number of dimensions. Define:

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

and:

$$\hat{d}_{n+1} = \frac{\|s_{n+1}\|^2_{A_{n+1}^{-1}} - \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}}{2\|s_{n+1}\|_1}.$$

The following lemma applies to Algorithm 2 with general weights λ_k.

Lemma 13 *The inner product $\lambda_k \left\langle g_k, A_k^{-1} s_k \right\rangle$ is a key quantity that occurs in our theory. Suppose that $A_{n+1} \succeq A_n$ for all n, then we can bound the sum of these inner products as follows:*

$$-\sum_{k=0}^{n} \lambda_k \left\langle g_k, A_k^{-1} s_k \right\rangle \le -\frac{1}{2}\|s_{n+1}\|^2_{A_{n+1}^{-1}} + \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

Proof We start by expanding $\frac{1}{2}\|s_{n+1}\|^2_{A_{n+1}^{-1}}$

$$\begin{aligned} \frac{1}{2}\|s_{n+1}\|^2_{A_{n+1}^{-1}} &\le \frac{1}{2}\|s_{n+1}\|^2_{A_n^{-1}} \\ &= \frac{1}{2}\|s_n\|^2_{A_n^{-1}} + \lambda_n \left\langle g_n, A_n^{-1} s_n \right\rangle + \frac{1}{2}\lambda_n^2 \|g_n\|^2_{A_n^{-1}}. \end{aligned}$$

Therefore

$$-\lambda_n \left\langle g_n, A_n^{-1} s_n \right\rangle \le \frac{1}{2}\|s_n\|^2_{A_n^{-1}} - \frac{1}{2}\|s_{n+1}\|^2_{A_{n+1}^{-1}} + \frac{1}{2}\lambda_n^2 \|g_n\|^2_{A_n^{-1}}.$$

Telescoping over time gives:

$$-\sum_{k=0}^{n} \lambda_k \left\langle g_k, A_k^{-1} s_k \right\rangle \le -\frac{1}{2}\|s_{n+1}\|^2_{A_{n+1}^{-1}} + \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

■

Below, we provide the analogue of Proposition 8 for the coordinate-wise setting.

Proposition 14 *(From Duchi et al. [10]) The gradient error term can be bounded as:*

$$\sum_{j=1}^{p}\sum_{k=0}^{n} \frac{g_{kj}^2}{\sqrt{G^2 + \sum_{i=0}^{k-1} g_{ij}^2}} \le 2\sum_{j=1}^{p}\sqrt{G^2 + \sum_{k=0}^{n-1} g_{kj}^2},$$

as long as $G \ge g_{ij}$ for all i, j.

Lemma 15 *It holds for the iterates of Algorithm 2*

$$\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) \leq \|s_{n+1}\|_1 D_\infty - \frac{1}{2}\|s_{n+1}\|^2_{A^{-1}_{n+1}} + \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

Proof We start by applying convexity:

$$\begin{aligned}
\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) &\leq \sum_{k=1}^{n} \lambda_k \langle g_k, x_k - x_*\rangle \\
&= \sum_{k=1}^{n} \lambda_k \langle g_k, x_k - x_0 + x_0 - x_*\rangle \\
&= \langle s_{n+1}, x_0 - x_*\rangle + \sum_{k=1}^{n} \lambda_k \langle g_k, x_k - x_0\rangle \\
&= \langle s_{n+1}, x_0 - x_*\rangle - \sum_{k=1}^{n} \lambda_k \left\langle g_k, A_k^{-1} s_k\right\rangle \\
&\leq \|s_{n+1}\|_1 \|x_0 - x_*\|_\infty - \sum_{k=1}^{n} \lambda_k \left\langle g_k, A_k^{-1} s_k\right\rangle.
\end{aligned}$$

Applying Lemma 13 we have:

$$\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_*\right) \leq \|s_{n+1}\|_1 \|x_0 - x_*\|_\infty - \frac{1}{2}\|s_{n+1}\|^2_{A^{-1}_{n+1}} + \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

∎

Theorem 16 *Consider the iterates of Algorithm 2. The ℓ_∞ initial distance $D_\infty = \|x_0 - x_*\|_\infty$ satisfies*

$$D_\infty \geq \hat{d}_{n+1} = \frac{\|s_{n+1}\|^2_{A^{-1}_{n+1}} - \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}}{2\|s_{n+1}\|_1}.$$

Proof Applying $f(x_k) - f_* \geq 0$ to the bound from Lemma 15 gives:

$$0 \leq \|s_{n+1}\|_1 D_\infty - \frac{1}{2}\|s_{n+1}\|^2_{A^{-1}_{n+1}} + \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

Rearranging this inequality, we obtain

$$\|s_{n+1}\|_1 D_\infty \geq \frac{1}{2}\|s_{n+1}\|^2_{A^{-1}_{n+1}} - \frac{1}{2}\sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}.$$

and, therefore,

$$D_\infty \geq \frac{\|s_{n+1}\|^2_{A^{-1}_{n+1}} - \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}}}{2\|s_{n+1}\|_1}.$$

∎

Lemma 17 *The ℓ_1-norm of s_{n+1} is bounded by:*

$$\|s_{n+1}\|_1 \leq 3d_{n+1} \|a_{n+1}\|_1 .$$

Proof By the definition of $\hat{d}_{n+1}$ we have:

$$\frac{1}{2} \|s_{n+1}\|^2_{A^{-1}_{n+1}} = \hat{d}_{n+1} \|s_{n+1}\|_1 + \frac{1}{2} \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}} .$$

and since $\hat{d}_{n+1} \leq d_{n+1}$,

$$\frac{1}{2} \|s_{n+1}\|^2_{A^{-1}_{n+1}} \leq d_{n+1} \|s_{n+1}\|_1 + \frac{1}{2} \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}} .$$

Furthermore, using Proposition 14, we obtain

$$\begin{aligned}
\frac{1}{2} \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}} &\leq \frac{1}{2} d^2_{n+1} \sum_{k=0}^{n} \|g_k\|^2_{A_k^{-1}} \\
&\leq d^2_{n+1} \sum_{i=1}^{p} \sqrt{G^2_\infty + \sum_{k=0}^{n-1} g^2_{ki}} \\
&= d^2_{n+1} \|a_{n+1}\|_1 .
\end{aligned}$$

Therefore, using inequality $2\alpha\beta \leq \alpha^2 + \beta^2$ with $\alpha^2 = 2d^2_{n+1} a_{(n+1)i}$ and $\beta^2 = \frac{s^2_{(n+1)i}}{2a_{(n+1)i}}$, we get

$$\begin{aligned}
2d_{n+1} \|s_{n+1}\|_1 = \sum_{i=1}^{p} 2d_{n+1} |s_{(n+1)i}| &\leq \sum_{i=1}^{p} \left(2d^2_{n+1} a_{(n+1)i} + \frac{s^2_{(n+1)i}}{2a_{(n+1)i}} \right) \\
&= 2d^2_{n+1} \|a_{n+1}\|_1 + \frac{1}{2} \|s_{n+1}\|^2_{A^{-1}_{n+1}} \\
&\leq 2d^2_{n+1} \|a_{n+1}\|_1 + d_{n+1} \|s_{n+1}\|_1 + \frac{1}{2} \sum_{k=0}^{n} \lambda_k^2 \|g_k\|^2_{A_k^{-1}} \\
&\leq 2d^2_{n+1} \|a_{n+1}\|_1 + d_{n+1} \|s_{n+1}\|_1 + d^2_{n+1} \|a_{n+1}\|_1 .
\end{aligned}$$

Rearranging, we get

$$d_{n+1} \|s_{n+1}\|_1 \leq 3d^2_{n+1} \|a_{n+1}\|_1 .$$

■

Theorem 18 *For a convex function with $G_\infty = \max_x \|\nabla f(x)\|_\infty$, D-Adapted AdaGrad returns a point $\hat{x}_n$ such that*

$$f(\hat{x}_n) - f_* = \mathcal{O}\left(\frac{\|a_{n+1}\|_1 D_\infty}{n+1} \right) = \mathcal{O}\left(\frac{pG_\infty D_\infty}{\sqrt{n+1}} \right)$$

as $n \to \infty$, where $D = \|x_0 - x_*\|_\infty$ for any x_* in the set of minimizers of f, as long as $d_0 \leq D_\infty$

Proof We will show the result holds for some n, where we choose n sufficiently large so that the following condition is satisfied. Since d_k is a non-decreasing sequence upper bounded by D, there must exist some $\hat{n}$ such that after $\hat{n}$ steps, $d_k \geq \frac{1}{2}d_{n+1}$ for all $k, n \geq \hat{n}$. We take $n \geq 2\hat{n}$.

Then:

$$\sum_{k=0}^{n} d_k \geq \frac{1}{4}(n+1)d_{n+1},$$
$$\therefore \frac{1}{\sum_{k=0}^{n} d_k} \leq \frac{4}{(n+1)d_{n+1}}.$$

So we have from Lemma 15 that:

$$\frac{1}{\sum_{k=0}^{n} d_k}\sum_{k=0}^{n} d_k\left(f(x_k)-f_*\right) \leq \frac{4}{(n+1)d_{n+1}}\left(\left\|s_{n+1}\right\|_1 D_\infty + \frac{1}{2}\sum_{k=0}^{n} d_k^2 \left\|g_k\right\|_{A_k^{-1}}^2\right).$$

From Proposition 14 we have:

$$\begin{aligned}\frac{1}{2}\sum_{k=0}^{n} d_k^2 \left\|g_k\right\|_{A_k^{-1}}^2 &\leq \frac{1}{2}d_{n+1}^2 \sum_{k=0}^{n} \left\|g_k\right\|_{A_k^{-1}}^2 \\ &\leq d_{n+1}^2 \left\|a_{n+1}\right\|_1.\end{aligned}$$

Plugging this in together with Lemma 17 gives:

$$\begin{aligned}\frac{1}{\sum_{k=0}^{n} d_k}\sum_{k=0}^{n} d_k\left(f(x_k)-f_*\right) &\leq \frac{4}{(n+1)d_{n+1}}\left(3d_{n+1}\left\|a_{n+1}\right\|_1 D_\infty + d_{n+1}^2\left\|a_{n+1}\right\|_1\right) \\ &= \frac{4}{n+1}\left(3\left\|a_{n+1}\right\|_1 D_\infty + d_{n+1}\left\|a_{n+1}\right\|_1\right).\end{aligned}$$

So using $d_{n+1} \leq D_\infty$ we have:

$$\frac{1}{\sum_{k=0}^{n} d_k}\sum_{k=0}^{n} d_k\left(f(x_k)-f_*\right) \leq \frac{16}{n+1}\left\|a_{n+1}\right\|_1 D_\infty.$$

Using Jensen's inequality on the left:

$$f(\hat{x}_n) - f_* \leq \frac{16}{n+1}\left\|a_{n+1}\right\|_1 D_\infty.$$

We can further simplify using $\left\|a_{n+1}\right\|_1 = \sum_{j=1}^{p}\sqrt{G_\infty^2 + \sum_{k=0}^{n} g_{kj}^2} \leq p\sqrt{n+1}G_\infty$:

$$f(\hat{x}_n) - f_* \leq \frac{16pG_\infty D_\infty}{\sqrt{n+1}},$$

which yields the result. ∎

Appendix D. Parameter settings

In this section, we list the parameters, architectures and hardware that we used for the experiments. The information is collected in Tables 1–11.

Table 1: Logistic regression experiment. The problems are part of the LIBSVM repository. Since there are no standard train/test splits, and due to the small sizes of the datasets, we present training accuracy curves only.

Hyper-parameter	Value
Epochs	100
GPUs	$1\times$V100
Batch size	16
Epochs	100
LR schedule	60,80,95 tenthing
Seeds	10
Decay	0.0
Momentum	0.0
Baseline LR	grid search

Table 2: CIFAR10 experiment. Our data augmentation pipeline followed standard practice: random horizontal flipping, then random cropping to 32×32 (padding 4), then normalization by centering around (0.5, 0.5, 0.5).

Hyper-parameter	Value
Architecture	Wide Resnet 16-8
Epochs	300
GPUs	$1\times$V100
Batch size per GPU	128
LR schedule	150-225 tenthing
Seeds	10
decay	0.0001
momentum	0.9
SGD LR	0.1

Table 3: CIFAR100 experiment. Following standard practice, we normalized the channels by subtracting ((0.5074,0.4867,0.4411) and dividing by (0.2011,0.1987,0.2025)). Augmentations used at training time were: random horizontal flips, random crop (32, padding=4, reflect).

Hyper-parameter	Value
Architecture	DenseNet [6,12,24,16], growth rate 12
Epochs	300
GPUs	$1\times$V100
Batch size per GPU	64
LR schedule	150-225 tenthing
Seeds	10
Decay	0.0002
Momentum	0.9
SGD LR	0.05

Table 4: ImageNet experiment. Normalization of the color channels involved subtracting (0.485, 0.456, 0.406), and dividing by (0.229, 0.224, 0.225). For data augmentation at training we used PyTorch's RandomResizedCrop to 224, then random horizontal flips. At test time images were resized to 256 then center cropped to 224.

Hyper-parameter	Value
Architecture	ResNet50
Epochs	100
GPUs	$8\times$V100
Batch size per GPU	32
LR schedule	30-60-90 tenthing
Seeds	5
Decay	0.0001
Momentum	0.9
SGD LR	0.1

Table 5: fastMRI experiment. We used the implementation from https://github.com/facebookresearch/fastMRI.

Hyper-parameter	Value
Architecture	12 layer VarNet 2.0
Epochs	50
GPUs	8×V100
Batch size per GPU	1
Acceleration factor	4
Low frequency lines	16
Mask type	Offset-1
LR schedule	flat
Seeds	5
Decay	0.0
Adam LR	0.0003
β_1, β_2	0.9, 0.999

Table 6: IWSLT14 experiment. Our implementation used FairSeq https://github.com/facebookresearch/fairseq defaults except for the parameters listed below. Note that the default Adam optimizer uses decoupled weight decay.

Hyper-parameter	Value
Architecture	lstm_wiseman_iwslt_de_en
Max Epoch	55
GPUs	1×V100
Max tokens per batch	4096
Warmup steps	4000
Dropout	0.3
Label smoothing	0.1
Share decoder, input, output embed	True
Float16	True
Update Frequency	1
LR schedule	Inverse square-root
Seeds	10
Decay	0.05
Adam LR	0.01
β_1, β_2	0.9, 0.98

Table 7: RoBERTa BookWiki experiment. Our implementation used FairSeq defaults except for the parameters listed below.

Hyper-parameter	Value
Architecture	roberta_base
Task	masked_lm
Max updates	23,000
GPUs	8×V100
Max tokens per sample	512
Dropout	0.1
Attention Dropout	0.1
Max sentences	16
Warmup	10,000
Sample Break Mode	Complete
Float16	True
Update Frequency	16
LR schedule	Polynomial decay
Seeds	5
Decay	0.0
Adam LR	0.001
β_1, β_2	0.9, 0.98

Table 8: GPT BookWiki experiment. Our implementation used FairSeq defaults except for the parameters listed below.

Hyper-parameter	Value
Architecture	transformer_lm_gpt
Task	language_modeling
Max updates	65,000
GPUs	8×V100
Max tokens per sample	512
Dropout	0.1
Attention Dropout	0.1
Max sentences	1
Warmup	10,000
Sample Break Mode	Complete
Share decoder, input, output embed	True
Float16	True
Update Frequency	16
LR schedule	Polynomial decay
Seeds	5
Decay	0.005
Adam LR	0.001
β_1, β_2	0.9, 0.98

Table 9: COCO Object Detection experiment. We used the Detectron2 codebase https://github.com/facebookresearch/detectron2, with the `faster_rcnn_X_101_32x8d_FPN_3x` configuration. We list its key parameters below.

Hyper-parameter	Value
Architecture	X-101-32x8d
Solver Steps (Schedule)	210000, 250000
Max Iter	270000
IMS Per Batch	16
Momentum	0.9
Decay	0.0001
SGD LR	0.02

Table 10: Vision Transformer experiment. We used the Pytorch Image Models codebase https://github.com/rwightman/pytorch-image-models.

Hyper-parameter	Value
Model	vit_tiny_patch16_224
Epochs	300
Batch Size	512
Sched	Cosine
Warmup Epochs	5
Hflip	0.5
aa	rand-m6-mstd0.5
mixup	0.1
cutmix	1.0
Crop Pct	0.9
BCE Loss	True
Seeds	5
Decay	0.1
Adam LR	0.001
β_1, β_2	0.9, 0.999

Table 11: Criteo Kaggle experiment. We used our own implementation of DLRM, based on the codebase provided at https://github.com/facebookresearch/dlrm.

Hyper-parameter	Value
Iterations	300 000
Batch Size	128
Schedule	Flat
Emb Dimension	16
Seeds	5
Decay	0.0
Adam LR	0.0001
β_1, β_2	0.9, 0.999

Appendix E. Additional notes

Theorem 19 *If $\|x_n - x_*\| \to 0$, and the learning rate (1) is used, then:*

$$\lim_{n\to\infty} d_n \geq \frac{D}{3}.$$

Proof By triangle inequality, we can bound the distance to x_* as

$$D = \|x_0 - x_*\| \leq \|x_n - x_*\| + \|x_n - x_0\| = \|x_n - x_*\| + \gamma_n \|s_n\|.$$

Let us plug in $\lambda_k = d_k \leq d_{n+1}$ in Lemma 7:

$$\|s_n\| \overset{(2)}{\leq} \frac{2d_n}{\gamma_n} + \frac{\sum_{k=0}^{n-1} \gamma_k \lambda_k^2 \|g_k\|^2}{2d_n} \leq \frac{2d_n}{\gamma_n} + \frac{d_n}{2} \sum_{k=0}^{n-1} \gamma_k \|g_k\|^2.$$

Using Proposition 8, we can further obtain

$$\gamma_n \|s_n\| \leq 2d_n + \frac{\gamma_n d_n}{2} \sum_{k=0}^{n-1} \frac{\gamma_k}{2} \|g_k\|^2 \overset{(4)}{\leq} 2d_n + \gamma_n^2 d_n \left(G^2 + \sum_{k=0}^{n-1} \|g_k\|^2 \right).$$

The last term can be simplified using the definition of γ_n to finally produce:

$$\gamma_n \|s_n\| \leq 2d_n + \gamma_n^2 d_n \left(G^2 + \sum_{k=0}^{n-1} \|g_k\|^2 \right) = 2d_n + d_n = 3d_n.$$

Now, assume that $x_n \to x_*$ in norm, so $\|x_n - x_*\| \to 0$. In that case, the bounds combined yield

$$D \leq \lim_n (\|x_n - x_*\| + \gamma_n \|s_n\|) = \lim_{n\to\infty} \gamma_n \|s_n\| \leq 3 \lim_{n\to\infty} d_n.$$

Thus, the value of d_n is asymptotically lower bounded by $\frac{D}{3}$. ■

E.1. A tighter lower bound on D

Using Lemma 4, we can obtain a slightly tighter bound than in Theorem 6. In particular, we have previously used the following bound:

$$\begin{aligned}
\sum_{k=0}^{n} \lambda_k \left(f(x_k) - f_* \right) &\leq \sum_{k=0}^{n} \lambda_k \langle g_k, x_k - x_* \rangle \\
&= \sum_{k=0}^{n} \lambda_k \langle g_k, x_k - x_0 + x_0 - x_* \rangle \\
&= \langle s_{n+1}, x_0 - x_* \rangle + \sum_{k=0}^{n} \lambda_k \langle g_k, x_k - x_0 \rangle \\
&= \langle s_{n+1}, x_0 - x_* \rangle - \sum_{k=0}^{n} \lambda_k \gamma_k \langle g_k, s_k \rangle \\
&\leq \|s_{n+1}\| \, \|x_0 - x_*\| - \sum_{k=0}^{n} \lambda_k \gamma_k \langle g_k, s_k \rangle .
\end{aligned}$$

From here, we can immediately conclude that

$$D = \|x_0 - x_*\| \geq \widetilde{d}_n = \frac{\sum_{k=0}^{n} \lambda_k \gamma_k \langle g_k, s_k \rangle}{\|s_{n+1}\|}.$$

Notice that it always holds $\widetilde{d}_n \geq \hat{d}_n$. The only complication that we can face is with Lemma 7, where we used the definition of $\hat{d}_n$ to obtain the upper bound. Nevertheless, one can prove the same bound with $\hat{d}_n$ replaced by $\widetilde{d}_n$ by repeating the same argument:

$$\frac{\gamma_{n+1}}{2} \|s_{n+1}\|^2 - \sum_{k=0}^{n} \frac{\gamma_k}{2} \lambda_k^2 \|g_k\|^2 = \hat{d}_{n+1} \|s_{n+1}\| \leq \widetilde{d}_{n+1} \|s_{n+1}\| \leq d_{n+1} \|s_{n+1}\|.$$

From that place, the rest of the proof of Lemma 7 follows in exactly the same way. The other proofs only use the monotonicity of the sequence and its boundedness by D, $d_k \leq d_{n+1} \leq D$, which would remain valid if replace $\hat{d}_n$ with $\widetilde{d}_n$.