AdaBatch: Efficient Gradient Aggregation Rules for Sequential and Parallel Stochastic Gradient Methods

Alexandre Défossez, Francis Bach

AdaBatch: Efficient Gradient Aggregation Rules for Sequential and Parallel Stochastic Gradient Methods

Alexandre Défossez
Facebook AI Research
Paris, France
defossez@fb.com

Francis Bach
Département d'Informatique
École Normale Supérieure
Paris, France
francis.bach@inria.fr

November 3, 2017

We study a new aggregation operator for gradients coming from a mini-batch for *stochastic gradient* (SG) methods that allows a significant speed-up in the case of sparse optimization problems. We call this method AdaBatch and it only requires a few lines of code change compared to regular mini-batch SGD algorithms. We provide a theoretical insight to understand how this new class of algorithms is performing and show that it is equivalent to an implicit per-coordinate rescaling of the gradients, similarly to what Adagrad methods can do. In theory and in practice, this new aggregation allows to keep the same sample efficiency of SG methods while increasing the batch size. Experimentally, we also show that in the case of smooth convex optimization, our procedure can even obtain a better loss when increasing the batch size for a fixed number of samples. We then apply this new algorithm to obtain a parallelizable stochastic gradient method that is synchronous but allows speed-up on par with Hogwild! methods as convergence does not deteriorate with the increase of the batch size. The same approach can be used to make mini-batch provably efficient for variance-reduced SG methods such as SVRG.

1. Introduction

We consider large-scale supervised learning with sparse features, such as logistic regression, least-mean-square or support vector machines, with a very large dimension as well as a very large number of training samples with many zero elements, or even an infinite stream. A typical example of such use of machine learning is given by Ads click prediction where many sparse features can be used to improve prediction on a problem with a massive online usage. For such problems, *stochastic gradient* (SG) methods have been used successfully [1, 2, 3].

Sparse optimization requires the usage of CPUs and unlike other domains in machine learning, it did not benefit much from the ever increasing parallelism accessible in GPUs or dedicated hardware. The frequency of CPUs has been stagnating and we can no longer rely on the increase of CPU sequential computational power for SG methods to scale with the increase of data [4]. New CPUs now rely on multi-core and sometimes multi-socket design to offer more power to its users. As a consequence, many attemps have been made at parallelizing and distributing SG methods [5, 6, 7, 8]. Those approaches can be classified in two types: (a) synchronous methods, that seek a speed-up while staying logically equivalent to a sequential algorithm, (b) asynchronous methods, which allow some differences and approximations from the sequential algorithms, such as allowing delays in gradient updates, dropping overlapping updates from different workers or allowing inconsistent reads from the model parameters. The latter such as Hogwild! [7] have been more successful as the synchronization overhead between workers from synchronous methods made them impractical.

Such methods however do not lead to a complete provability of convergence for step-sizes used in practice, as most proof methods require some approximation [7, 9]. Proving convergence for such methods is not as straightforward as there is not anymore a clear sequence of iterates that actually exist in memory and conflicting writes to memory or inconsistent reads can occur. When increasing the number of workers it is also likely to increase how stale a gradient update is when being processed.

Synchronous approaches rely mostly on the usage of mini-batches in order to parallelize the workload [10]. Increasing the size of the mini-batches will lead to a reduction of the variance of the gradients and the overall estimator. However for the same number of samples we will be doing B times less iterations where B is the size of the mini-batch. In practice one has to increase the learning rate (i.e., the step size) in order to compensate and achieve the same final accuracy as without mini-batches; however increasing the step size can lead to divergence and is sometimes impossible [11]. The decrease in sample efficiency (i.e., a worse performance for a given number of processed training samples) is especially visible early during optimization and will lower over time as the algorithm reaches an asymptotic regime where using mini-batches of size B will have the same sample efficiency as without mini-batches.

In this paper, we make the following contributions:

- We propose in Section 2 a new merging operator for gradients computed over a mini-batch, to replace taking the average. Instead, for each mini-batch we count for each coordinate how many samples had a non zero gradient in that direction. Rather than taking the sum of all gradients and dividing by B we instead divide each coordinate independently by the number of times it was non zero in the batch. This happens to be equivalent to reconditioning the initial problem in order to exploit its sparsity. Because each coordinate is still an average (albeit a stochastic one), the norm of the gradient will stay under control. In order to notice this effect, one has to look at the problem in a different geometry that accounts for the sparsity of the data. We also draw a parallel with Adagrad-type methods [12, 13] as our operator is equivalent to an implicit rescaling of the gradients per coordinate.
- We show in Section 3 that this new merging rule outperforms regular mini-batch on sparse data and that it can have the same if not an improved sample efficiency compared to regular SGD without mini-batch.
- We explain in Section 4 how this can be used to make synchronous parallel or distributed methods able to compete with asynchronous ones while being easier to study as they are logically equivalent to the sequential version.
- We extend our results to variance-reduced SG methods such as SVRG in Section 5 and show similar gains are obtained when using AdaBatch.
- We present in Section 6 experimental results to support our theoretical claims as well as a proof of concept that our new merging operator can make synchronous parallel SG methods competitive with asynchronous ones. We also extend our experiments to variance reduction methods like SVRG and show that AdaBatch yields similar improvement as in the case of SG methods.

Notations. Throughout this paper, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$ and for any symmetric positive definite matrix D, $\|\cdot\|_D$ is the norm defined by D so that $\forall x \in \mathbb{R}^d$, $\|x\|_D^2 = x^T D x$; for a set A, $|A|$ denotes the cardinality of A. If $x \in \mathbb{R}^d$ then $x^{(k)}$ denotes the k-th coordinate and $\mathrm{Diag}\,(x)$ is the $d \times d$ diagonal matrix with the coefficient of x on its diagonal. We define for any integer n, $[n] := \{1, 2, \ldots, n\}$. Finally, for any function $h : \mathbb{R}^d \to \mathbb{R}$, we will define the support of h as

$$\mathcal{S}\,(h) = \{k \in [d] : \exists (x, y) \in \mathbb{R}^d \times \mathbb{R}^d, x^{(k)} \neq y^{(k)} \text{ and } h(x) \neq h(y)\}. \tag{1.1}$$

1.1. Problem setup

We consider f a random variable with values in the space of convex functions $\mathcal{F}$ from $\mathbb{R}^d$ to $\mathbb{R}$. We define $F(w) := \mathbb{E}\,[f(w)]$ and we wish to solve the optimization problem

$$F_* = \min_{w \in \mathbb{R}^d} F(w). \tag{1.2}$$

It should be noted that the gradient f' will only have non zero coordinate along the directions of the support $\mathcal{S}\,(f)$ so that if the support of f is sparse, so will the update for SG methods. We define $p \in \mathbb{R}^d$ by $\forall k \in [d], p^{(k)} := \mathbb{P}\,\{k \in \mathcal{S}\,(f)\}$. We take $p_{\min} := \min(p)$ and $p_{\max} = \max(p)$.

This setup covers many practical cases, such as finite sum optimization where f would have the uniform distribution over the sum elements or stochastic online learning where f would be an infinite stream of training samples.

One example of possible values for f is given by linear predictions with sparse features. Let us assume X is a random variable with values in $\mathbb{R}^d$ and $\phi : \mathbb{R} \to \mathbb{R}$ a random convex function. Then one can take $f(w) := \phi(X^T w)$; $\mathcal{S}\,(f)$ would be the same as $\mathcal{S}\,(X)$ defined as the non zero coordinates of the vector X. The problem given by (1.2) becomes

$$F_* = \min_{w \in \mathbb{R}^d} \mathbb{E}\left[\phi(X^T w)\right]. \tag{1.3}$$

In the case of logistic regression, one would have for instance $\phi(X^T w) = \log(1 + \exp -Y X^T w)$ for $Y \in \{-1, 1\}$ the random label associated with the feature vector X.

The convergence properties of SG methods depend on the properties of the Hessian F'' of our objective function F, as we will show in Section 3. The closer it is to identity, the faster SG methods will converge and this convergence will be as fast for all the coordinates of w. For example, in the case of sparse linear prediction such as given by (1.3), with binary features $X^{(k)} \in \{0, 1\}$ that are uncorrelated, the Hessian is such that $F''(w) = \mathbb{E}\left[\phi''(X^T w) X X^T\right]$. If there exist M and m so that we have $\forall z \in \mathbb{R}, m \leq \phi''(z) \leq M$, then we immediately have

$$\begin{aligned}\forall w \in \mathbb{R}^d, m(1 - p_{\max})\mathrm{Diag}\,(p) &\preceq F''(w), \\ F''(w) &\preceq M\big(1 + \textstyle\sum_{k \in [d]} p^{(k)}\big)\mathrm{Diag}\,(p)\,.\end{aligned} \tag{1.4}$$

We notice here that we have a specific structure to the geometry of F which depends on p and which need to be taken into account. The proof of (1.4) is given in the supplementary material (Section C.4).

Finally, we want not only to solve problems (1.2) or (1.3), but to be able to do so while using W workers. Those workers can either be running on the same machine with shared memory or in a distributed fashion.

1.2. Related work

There have been several approaches for parallelizing or distributing SG methods.

Parallelized stochastic gradient descent. This approach described by [5] consists in splitting a dataset in W different parts and averaging the model obtained by W independent workers. Model averaging always reduces the variance of the final estimator but the impact on the bias is not as clear. For sparse optimization this approach does not in practice outperform a purely sequential algorithm [7].

Delayed stochastic gradient descent. The effect of delay for constant step-size stochastic gradient descent has been studied by [6]. Allowing for delay will remove the need for synchronization and thus limit the overhead when parallelizing. The main result of [6] concludes that there is two different regimes. During the first phase, delay will not help convergence, although once the asymptotic terms are dominating, a theoretical linear speedup with the number of worker is recovered.

Using mini-batches is a popular alternative for parallelizing or distributing SGD. In [10], the reduction of the variance of the gradient estimate is used to prove improvement in convergence. Our theoretical and practical results show that in the case of sparse learning, mini-batch do not offer an improvement during the first stage of optimization. We believe our merging rule is a simple modification of mini-batch SGD that can considerably improve convergence speed compared to regular mini-batch.

The case of averaged stochastic gradient descent with constant step size for least-squares regression has been studied in much detail and in that case it is possible to get an explicit expression for the convergence of the algorithm [11]. During the first phase of optimization, in order to achieve the same accuracy after a given number of samples, the step size must be increased proportionally to the batch size which is possible up to a point after which the algorithm will diverge. We draw the same conclusions in a more generic case in Section 3.

In [14], a specific subproblem is solved instead of just averaging the gradients in a mini-batch. However, solving a subproblem is much more complex to put in place and requires the tuning of extra parameters. Our method is very simple as it only requires a per-coordinate rescaling of the gradients and does not require any parameter tuning.

Hogwild! is a very simple parallel SG method. Each worker processes training examples completely in parallel, with no synchronization and accessing the same model in memory [7]. The overhead is minimal, however the theoretical analysis is complex. New proof techniques have been introduced to tackle those issues [9]. Our contribution here is to make synchronous methods almost as fast as Hogwild!. This has an interest for cases where Hogwild! cannot perform optimally, for instance with a mixture of dense and sparse features, or in the distributed setting where memory cannot be shared. Hogwild! has inspired parallel versions for SDCA, SVRG and SAGA [8, 9, 15]. AdaBatch can similarly be extended to those algorithms and we provide proof for SVRG. **Cyclades** [16] builds on Hogwild!, assigning training samples to specific workers using graph theory results to remove conflicts.

Adagrad. Adagrad [12] performs a per coordinate rescaling dependent on the size of past gradients that has proven to be highly efficient for sparse problem, besides it can be combined with Hogwild! for parallel optimization [17]. Adagrad rescaling is similar in nature to the one performed by AdaBatch. Adagrad has a step size going to 0 with the number of iterations which gives good convergence properties for various problems. On the other hand, AdaBatch works with a wider range of methods such as SVRG. Constant step size has proven useful for least-mean-square problems [18] or in the field of deep learning [19].

2. AdaBatch for SGD

In this section, we will focus on constant step size stochastic gradient descent to give an intuition on how AdaBatch works. AdaBatch can be extended in the same way to SVRG (see Section 5). We assume we are given a starting point $w_0 \in \mathbb{R}^d$ and we define recursively

$$\forall n > 0, w_n = w_{n-1} - g_n, \tag{2.1}$$

for a sequence g_n of stochastic gradient estimates based on independent gradients $f'_{n,1}, \dots, f'_{n,B}$. We define $g_{\mathrm{mb},n}$ as

$$\forall k \in [d],\ g_{\mathrm{mb},n}^{(k)} = \frac{1}{B} \sum_{b:k\in\mathcal{S}(f)} f'_{n,b}(w_{n-1})^{(k)}. \tag{2.2}$$

Plugging (2.2) into (2.1) yields the regular SGD mini-batch algorithm with constant step size γ and batch size B.

For each iteration $n > 0$ and dimension $k \in [d]$, we denote $D_{n,k} := \{b \in [B] : k \in \mathcal{S}(f_{n,b})\}$. We introduce $g_{\mathrm{ab},n}$, the gradient estimate of AdaBatch, as, $\forall k \in [d]$,

$$g_{\mathrm{ab},n}^{(k)} = \begin{cases} \frac{\sum_{b\in D_{n,k}} f'_{n,b}(w_{n-1})^{(k)}}{|D_{n,k}|} & \text{if } D_{n,k} \neq \varnothing, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

Instead of taking the average of all gradients, for each coordinate we make an average but taking only the non zero gradients into account. Let us take a coordinate $k \in [d]$; if $p^{(k)}$ is close to 1, then the AdaBatch update for this coordinate will be the same as with regular mini-batch with high probability. On the other hand, if $p^{(k)}$ is close to zero, the update of AdaBatch will be close to summing the gradients instead of averaging them. Adding updates instead of averaging has been shown to be beneficial in previous work such as in CoCoa+ [20], a distributed SDCA-inspired optimization algorithm. We observed experimentally that in order to achieve the same performance with mini-batch compared to AdaBatch, one has to take a step size that is proportional to B. This will boost convergence for less frequent features but can lead to divergence when B increases because the gradient for frequent features will get too large. Our method allows to automatically and smoothly move from summing to averaging depending on how frequent a feature is.

Let us consider the expectation for those two updates rules, we have to use the expectation of $g_{\mathrm{mb},n}$ and $g_{\mathrm{ab},n}$. We have immediately $\mathbb{E}[g_{\mathrm{mb},n}] = F'(w_{n-1})$. Thus when using the regular mini-batch update rule, one obtains an unbiased estimate of the gradient. The main advantage of mini-batch is a reduction by a factor B of the stochastic noise near the optimal value, as explained in Section 3. With our new rule, using Lemma 1 from the supplementary material (with divisions of probabilites taken element-wise), we have:

$$\mathbb{E}[g_{\mathrm{ab},n}] = \mathrm{Diag}\left(\frac{1-(1-p)^B}{p}\right) F'(w_{n-1}). \tag{2.4}$$

Interestingly, we now have a gradient that is equivalent to a reconditioning of F'. We can draw here a parallel with Adagrad [12] which similarly uses per-coordinate step sizes. When using Adagrad, the update rule becomes

$$w_n = w_{n-1} - \gamma (C_n^{\mathrm{adag}})^{-1} f'_n(w_{n-1}) \text{ with}$$
$$C_n^{\mathrm{adag}} = \mathrm{Diag}\left(\alpha^{-1}\sqrt{\epsilon + \textstyle\sum_{i\in[n-1]} (f'_i(w_{i-1})^{(k)})^2}\right)_{k\in[d]}.$$

The goal is to have an adaptative step size that will have a larger step size for coordinate for which the gradients have a smaller magnitude. One should note a few differences though:

- Adagrad relies on past informations and updates the reconditioning at every iteration. It works without any particular requirement on the problem. Adagrad also forces a decaying step size. Although this give Adagrad good convergence properties, it is not always suitable, for instance when using variance reduction methods such as SVRG where the step size is constant.

- On the other side, the AdaBatch scaling stays the same (in expectation) through time and only tries to exploit the structure coming from the sparsity of the problem. It does not require storing extra information and can be adapted to other algorithms such as SVRG or SAGA.

When used together with mini-batch, Adagrad will act similarly to AdaBatch and maintain the same sample efficiency automatically even when increasing the batch size.

Deterministic preconditioning. One can see that when $B \to \infty$, the reconditioning in (2.4) goes to $\mathrm{Diag}\,(p)^{-1}$. One could think of directly scaling the gradient by $\mathrm{Diag}\,(p)^{-1}$ to achieve a tighter bound in (1.4). However this would make the variance of the gradient explodes and thus is not usable in practice as shown in Section 3. A key feature of our update rule is stability; because every coordinate of $g_{\mathrm{ab},n}$ is an average, there is no chance it can diverge.

Using directly $\mathrm{Diag}\,(p)^{-1}$ is not possible, however reconditioning by $C_{B,p} := \mathrm{Diag}\left(\frac{1-(1-p)^B}{p}\right)$ when using a mini-batch of size B might just work as it will lead to the same expectation as (2.3). If we define $g_{\mathrm{C},n} := C_{B,p} f'(w_{n-1})$, we immediately have $\mathbb{E}\,[g_{\mathrm{C},n}] = C_{B,p} F'(w_{n-1})$. Intuitively, $C_{B,p}$ is getting us as close as possible to the ideal reconditioning $\mathrm{Diag}\,(p)^{-1}$ leveraging the mini-batch size in order to keep the size of the gradients under control. Both $g_{\mathrm{C},n}$ and $g_{\mathrm{ab},n}$ allow to obtain a very similar performance both in theory and in practice, so that which version to choose will depend on the specific task to solve. If it is possible to precompute the probabilities p then one can use $g_{\mathrm{C},n}$ which has the advantages of not requiring the extra step of counting the features present in a batch. On the other hand, using $g_{\mathrm{ab},n}$ allows to automatically perform the same reconditioning with no prior knowledge of p.

3. Convergence results

We make the following assumptions which generalize our observation from Section 1.1 for sparse linear prediction.

Assumption 1. *We assume there exists a convex compact set $\mathcal{D} \subset \mathbb{R}^d$, μ, L and R strictly positive so that the following assumptions are satisfied.*

1. *The Hessians F'' (resp. f'') of F (resp. f) are such that:*

$$\begin{aligned} &\forall w \in \mathcal{D},\ \ \mu \mathrm{Diag}\,(p) \preceq F''(w) \preceq L \mathrm{Diag}\,(p)\,, \quad \textit{and} \\ &\forall w \in \mathcal{D}, f''(w) \preceq R^2\,\mathrm{Id}\,. \end{aligned} \tag{3.1}$$

2. *Let $w_* := \arg\min_{w\in\mathcal{D}} F(w)$,*

$$F'(w_*) = 0. \tag{3.2}$$

In particular, w_* is a global minimizer of F over $\mathbb{R}^d$.

Those assumptions are easily met in the case of sparse linear predictions. If $f(w) := \phi(X^T w)$, with $\forall k \in [d],\ X^{(k)} \in \{0,1\}$ uncorrelated, $\|X\|^2 \le G^2$ almost surely and $m \le \phi'' \le M$, then the assumptions above are verified for $L = M(1 + \sum_{k\in[d]} p^{(k)})$, $\mu = m(1 - p_{\max})$, and $R^2 = G^2 M$. In the case of the logistic loss, $M := 1/4$ and μ typically exist on any compact but is not explicitly available. Note though that it is not required to know μ in order to train any of the algorithms studied here. More details are given in the supplementary material (Section C.4).

Detailed proofs of the following results are given in the supplementary material (Section C). Our proof technique is based on a variation from the one introduced by [21]. It requires an extra projection step on $\mathcal{D}$. In practice however, we did not require it for any reasonable step size that does not make the algorithm diverge and our bounds do not depend on $\mathcal{D}$ because of (3.2). The results are summarized in Table 1. We use a constant step size as a convenience for comparing the different algorithms. It is different but equivalent to using a decreasing step size (see [22] just before Corollary 1). For instance, if we know the total number of iterations is $n \gg 1$, taking $\gamma = \frac{6\log(n)}{n}$, the bias term is approximately $\frac{\mu}{4\log(n)n^2}$. The variance term which is proportional to γ is a $O(\log(n)/n)$ which is the usual rate for strongly convex SGD.

Bias/variance decomposition. We notice that the final error is made of two terms, one that decreases exponentially fast and measures how quickly we move away from the starting point and one that is constant, proportional to γ and that depends on the stochastic noise around the optimal value. We will call the former the *bias* term and the latter the *variance* term, following the terminology introduced by [18]. The bias term decreases exponentially fast and will be especially important during the early stage of optimization and when μ is very small. The variance term is the asymptotically dominant term and will prevail when close to the optimum. In practical applications, the bias term can be the most important one to optimize for [23]. This has also been observed for deep learning, where most of the optimization is spent far from the optimum [19].

We immediately notice that rescaling gradients by $\mathrm{Diag}\,(p)^{-1}$ is infeasible in practice unless B is taken of the order of $p_{\min}^{-1}$, because of the exploding variance term and the tiny step size.

Method	$F_{N/B} - F_*$		$\gamma_{\max}$
Mini-batch	$(1 - \gamma p_{\min}\mu/2)^{N/B}\frac{\delta_0}{\gamma}$	$+\gamma\frac{2\sigma^2}{B}$	$\gamma\left[Lp_{\max} + \frac{2R^2}{B}\right] \leq 1$
AdaBatch	$(1 - \gamma p_{\min}^{+B}\mu/2)^{N/B}\frac{\delta_0}{\gamma}$	$+2\gamma\sigma^2$	$\gamma\left[L + 2R^2\right] \leq 1$
$\mathrm{Diag}\,(p)^{-1}$	$(1 - \gamma\mu/2)^{N/B}\frac{\delta_0}{\gamma}$	$+\gamma\frac{2\sigma^2}{p_{\min}B}$	$\gamma\left[L + \frac{2R^2}{p_{\min}B}\right] \leq 1$

Table 1: Convergence rates for the different methods introduced in Section 2. N represents the total number of samples, so that the number of iterations is N/B; $\gamma_{\max}$ is the maximum step size that guarantees this convergence, $\sigma^2 := \mathbb{E}\left[\|f'(w_*)\|^2\right]$ the gradient variance at the optimum, and $\forall p \in [0,1], p^{+B} := 1 - (1-p)^B$. The $\mathrm{Diag}\,(p)^{-1}$ method consists in reconditioning by $\mathrm{Diag}\,(p)^{-1}$. Moreover, $\delta_0 := \|w_0 - w_*\|_A^2$ where $A = \mathrm{Id}$ for mini-batch SGD, $A = \mathrm{Diag}\left(p^{+B}/p\right)$ for AdaBatch and $A = \mathrm{Diag}\,(p)^{-1}$ for the last method.

Mini-batch. Let us now study the results for mini-batch SGD. The variance term is always improved by a factor of B if we keep the same step size. Most previous works on mini-batch only studied this asymptotic term and concluded that because of this linear scaling, mini-batch was efficient for parallel optimization [10]. However, when increasing the batch size, the number of iterations is divided by B but the exponential rate is still the same. The bias term will thus not converge as fast unless we increase the step size. We can see two regimes depending on B. If $B \ll \frac{2R^2}{Lp_{\max}}$, then the constraint is $\gamma \leq \frac{B}{2R^2}$. Thus, we can scale γ linearly and achieve the same convergence for both the variance and bias term as when $B = 1$. However, if $B \gg \frac{2R^2}{Lp_{\max}}$, then the constraint is $\gamma \leq \frac{1}{Lp_{\max}}$. In this regime, it is not possible to scale up infinitely γ and thus it is not possible to achieve the same convergence for the bias term as when $B = 1$. We have observed this in practice on some datasets.

AdaBatch. With AdaBatch though, if $p_{\min} \ll 1$, then $1 - (1 - p_{\min})^B \approx Bp_{\min}$. In such case, the bias term is $(1 - \gamma Bp_{\min}\mu/2)^{N/B}\frac{\delta_0}{\gamma} \approx (1 - \gamma p_{\min}\mu/2)^N\frac{\delta_0}{\gamma}$, thus showing that we can achieve the same convergence speed for a given number of samples as when $B = 1$ as far as the bias term is concerned. The variance term and the maximum step size are exactly the same as when $B = 1$. Thus, as long as $1 - (1 - p_{\min})^B \approx Bp_{\min}$, AdaBatch is able to achieve at least the same sample efficiency as when $B = 1$. This is a worst case scenario, in practice we observe on some datasets an improved efficiency when increasing B, see Section 6. Indeed for rare features the variance of the gradient estimate will not be decreased with larger batch-size, however for features that are likely to appear more than once in a batch, AdaBatch will still obtain partial variance reduction through averaging more than one gradient. Although we do not provide the full proof of this fact, this is a consequence of Lemma 2 from the supplementary material.

4. Wild AdaBatch

We now have a SG method trick that allows us to increase the size of mini-batches while retaining the same sample efficiency. Intuitively one can think of sample efficiency as how much information we extract from each training example we process i.e. how much the loss will decrease after a given number of samples have been processed. Although it is easy to increase the number of samples processed per seconds when doing parallel optimization, this will only lead to a true speedup if we can retain the same sample efficiency as sequential SGD. If the sample efficiency get worse, for instance when using regular mini-batch, then we will have to perform more iterations to reach the same accuracy, potentially canceling out the gain obtained from parallelization.

When using synchronous parallel SG methods such as [10], using large mini-batches allows to reduce the overhead and thus increase the number of samples processed per second. SGD with mini-batches typically suffers from a lower sample efficiency when B increases. It has been shown to be asymptotically optimal [14, 10], however our results summarized in Section 3 show that in cases where the step size cannot be taken too large, it will not be able to achieve the same sample efficiency as SGD without mini-batch.

We have shown in Section 3 that for sparse linear prediction, AdaBatch can achieve the same sample efficiency as for $B = 1$. Therefore, we believe it is a better candidate than regular mini-batch for parallel SGD. In Section 6, we will present our experimental results for Wild AdaBatch, a Hogwild! inspired, synchronous SGD algorithm. Given a batch size B and W workers, they will first compute in parallel B gradients, wait for everyone to be done and then apply the updates in parallel to update w_n. The key advantage here is that thanks to the synchronization, analysis of this algorithm is easier. We have a clear sequence of iterates w_n in memory and there is no delay or inconsistent read. It is still possible that during the update phase, some updates will be dropped because of overlapping writes to memory, but that can be seen simply as a slight decrease of the step size for those coordinate. In practice, we did not notice any difference with the sequential version of AdaBatch.

5. AdaBatch for SVRG

SVRG [24] is a variance-reduced SG method that has a linear rate of convergence on the training error when F is given by a finite mean of functions $F := \frac{1}{N}\sum_{i\in[N]} f_i$. This is equivalent to f following the uniform law over the set $\{f_1, \ldots, f_N\}$. SVRG is able to converge with a constant step size. To do so, it replaces the gradient $f'(w)$ by $f'(w) - f'(y) + F'(y)$ where y is updated every epoch (an epoch being m iterations where m is a parameter to the algorithm, typically of the order of the number of training samples). Next, we show the difference between regular mini-batch SVRG and AdaBatch SVRG and give theoretical results showing improved convergence for the latter.

We now only assume that F verifies the following inequalities for $\mu > 0$, almost surely,

$$\forall w \in \mathbb{R}^d,\ \ \mu \mathrm{Diag}\,(p) \preceq F''(w) \text{ and } f(w) \preceq L\mathrm{Diag}\,(p)\,. \tag{5.1}$$

Let us take a starting point $y_0 \in \mathbb{R}^d$ and $m \in \mathbb{N}^*$. For all $s = 0, 1, \ldots$, we have $w_{s,0} := y_s$ and for all $n \in [m]$ let us define

$$w_{s,n} := w_{s,n-1} - \gamma g_{s,n}, \quad y_{s+1} := \frac{1}{m}\sum_{n\in[m]} w_{s,n},$$

with $g_{s,n}$ the SVRG update based on $(f_{s,n,b})_{b\in[B]}$ i.i.d. samples of f. Let us introduce

$$\forall k \in [d], D_{s,n}^{(k)} := \left\{b \in [B] : k \in \mathcal{S}\left(f'_{s,n,b}\right)\right\}.$$

For any dimension k such that $D_{s,n}^{(k)} \neq \varnothing$ we have

$$g_{s,n}^{(k)} := \frac{1}{C_{s,n}^{(k)}}\Big(\sum_{b\in D_{s,n}^{(k)}} f'_{s,n,b}(w_{s,n-1})^{(k)} - f'_{s,n,b}(y_s)^{(k)} + F'(y_s)^{(k)}/p^{(k)}\Big),$$

and for $D_{s,n}^{(k)} = \varnothing$ we take $g_{s,n}^{(k)} := 0$. Regular mini-batch SVRG is recovered for $C_{s,n}^{(k)} := B$. On the other hand, AdaBatch SVRG is obtained for $C_{s,n}^{(k)} := \left|D_{s,n}^{(k)}\right|$. One can note that we used a similar trick to the one in [9] in order to preserve the sparsity of the updates.

For both updates, there exists a choice of γ and m such that

$$\mathbb{E}\left[F(y_s) - F_*\right] \leq 0.9^s (F(y_0) - F_*).$$

For regular mini-batch, it is provably sufficient to take $\gamma_{\text{mb}} = \frac{1}{L}$ and $m_{\text{mb}} \approx \frac{2.2L}{p_{\min}\mu}$. For AdaBatch, we have $\gamma_{\text{ab}} = \frac{1}{10L}$ and $m_{\text{ab}} \approx \frac{20L}{Bp_{\min}\mu}$. We notice that as we increase the batch size, we require the same number of inner iterations when using regular mini-batch update. However, each update requires B times more samples as when $B = 1$. On the other hand when using AdaBatch, m_{ab} is inversely proportional to B so that the total number of samples required to reach the same accuracy is the same as when $B = 1$. We provide detailed results and proofs in the supplementary material (Section D). Note that similar results should hold for epoch-free variance-reduced SG methods such as SAGA [25].

6. Experimental results

We have implemented both Hogwild! and Wild AdaBatch and compared them on three datasets, *spam*[1], *news20*[2], and *url* [26]. *Spam* has 92,189 samples of dimension 823,470 and an average of 155 active features per sample. *News20* has 19,996 samples of dimension 1,355,191 and an average of 455 active features per example. Finally, *url* has 2,396,130 samples of dimension 3,231,961 and an average of 115 active features per example.

We implemented both in C++ and tried our best to optimize both methods. We ran them on an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz with 24 CPUs divided in two sockets. Each socket contains 12

[1] http://plg.uwaterloo.ca/~gvcormac/trecspamtrack05/trecspam05paper.pdf
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

physical CPUs for 24 virtual ones. In our experiments though, it is better to keep the number of threads under the number of actual physical CPUs on a single socket. We restricted each experiment to run on a single socket in order to prevent NUMA (non uniform memory access) issues. For all the experiments we ran (and all methods), we perform a grid search to optimize the step size (or the main parameter of Adagrad). We then report the test error on a separate test set.

Wild AdaBatch. We trained each algorithm for logistic regression with 5 passes over the dataset, 1 pass only in the case of *url*. We normalized the features so that each sample has norm 1. For each dataset, we evaluate 3 methods: Wild AdaBatch (AB) and Wild mini-batch SGD (MB, the same as AB but with the regular average of the gradients) as well as Hogwild! (HW) for various numbers of workers W. For AdaBatch and mini-batch SGD, the batch size is set to $B = 10W$ for *news20* and $B = 50$ for *url* and *spam* which was giving a better speedup for those datasets. We take W going from 1 to 12, which is the maximum number of physical CPUs on a single socket on our machine. We also evaluate purely sequential SGD without mini-batch (SEQ).

We only present here the result for *news20*. The figures for the other datasets can be found in the supplementary material Section E. In Figure 1 we show the convergence as a function of the wall-clock time. In Figure 3, we give the wall-clock time to reach a given test error (where our method is achieving close to a linear speed-up) and the number of processed samples per seconds. On *news20*, the gain in sample efficiency actually allows Wild AdaBatch to reach the goal the fastest, even though Wild AdaBatch can process less samples per seconds than Hogwild!, thanks to its improved sampled efficiency.

Comparison with Adagrad. We compare AdaBatch with Adagrad for various batch-size on Figure 4 when trained with a fixed number of samples, so that when B increases, we perform less iterations. On the *url* dataset, Adagrad performs significantly better than AdaBatch, however we notice that as the batch-size increases, the gap between AdaBatch and Adagrad reduces. On the *spam* dataset with the least-mean-square loss, constant step size SGD performs better than Adagrad. We believe this is because Adagrad is especially well suited for non strictly convex problems. For strictly convex problem though, constant step size SGD is known to be very efficient [18]. We also plotted the performance of constant step size regular mini-batch SGD. In all cases, regular mini-batch scales very badly as the batch size increases. We fine tune the step size for each batch size and observed the regular mini-batch will take a larger step size for small batch sizes, that allows to keep roughly the same final test error. However, when the batch size increases too much, this is no longer possible as it makes optimization particularly unstable, thus leading to a clear decrease in sample efficiency. Finally, AdaBatch can even improve the sample complexity when increasing B. We believe this comes from the variance reduction of the gradient for features that occurs more than once in a mini-batch, which in turn allows for a larger step size.

SVRG. We also compared the effect of AdaBatch on SVRG. On Figure 2 we show the training gap $F_N - F^*$ on *url* for the log loss with a small L2 penalty. This penalty is given by $\frac{10^{-4}}{2}\left\|w\right\|^2_{\mathrm{diag}(p)}$, chosen to respect our hypothesis and to prevent overfitting without degrading the testing error. All the models are trained with 10 iterations over all the samples in the dataset, so that if B is larger, the model will perform less iterations. We observe that as we increase the batch size, the sample efficiency of regular mini-batch deteriorates. The variance reduction coming from using a larger mini-batch is not sufficient to compensate the fact that we perform less iterations, even when we increase the step size. On the other hand, AdaBatch actually allows to improve the sample efficiency as it allows to take a larger step size thanks to the gradient variance reduction for the coordinates with $p^{(k)}B$ large enough. To the best of ourknowledge, AdaBatch is the first mini-batch aggregation rule that is both extremely simple and allows for a better sample efficiency than sequential SGD.

7. Conclusion

We have introduced a new way of merging gradients when using SG methods with mini-batches. We have shown both theoretically and experimentally that this approach allows to keep the same sample efficiency as when not using any mini-batch and sometimes even improve it. Thanks to this feature, AdaBatch allowed us to make synchronous parallel SG methods competitive with Hogwild!. Our approach can extend to any SG methods including variance-reduced methods. Although not explored yet, we also believe that AdaBatch is promising for distributed optimization. In such a case, memory is no longer shared so that Hogwild! cannot be used. Distributed mini-batch or SGD with delay have been used in such case [10, 6]; AdaBatch is a few line change for distributed mini-batch which could vastly improve the convergence of those methods.

Acknowledgements. We thank Nicolas Flammarion, Timothée Lacroix, Nicolas Usunier and Léon Bottou for interesting discussions related to this work. We acknowledge support from the European Research Council (SEQUOIA project 724063).

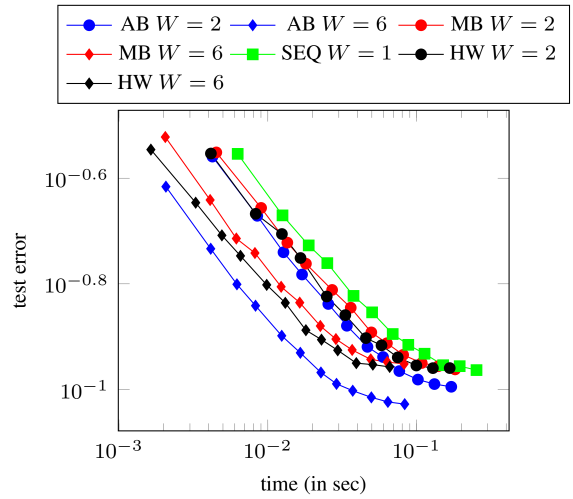


Figure 1: Convergence result for *news20*. The error is given as a function of the wall-clock time.

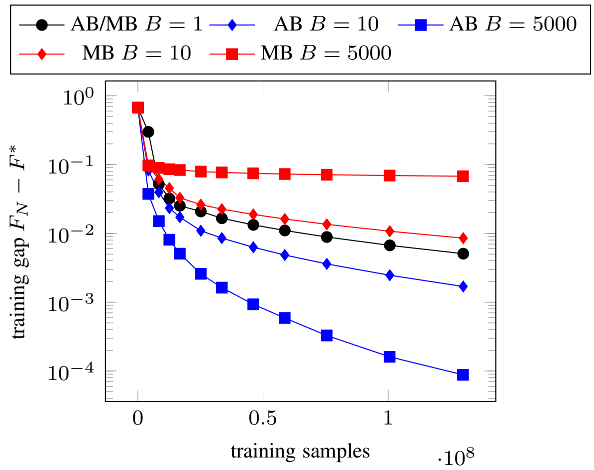


Figure 2: Comparison of the training gap $F_n - F_*$ for regular mini-batch vs. AdaBatch with SVRG on *url* for the log loss with an L2 penalty of $\frac{10^{-4}}{2} \|w\|^2_{\mathrm{diag}(p)}$.

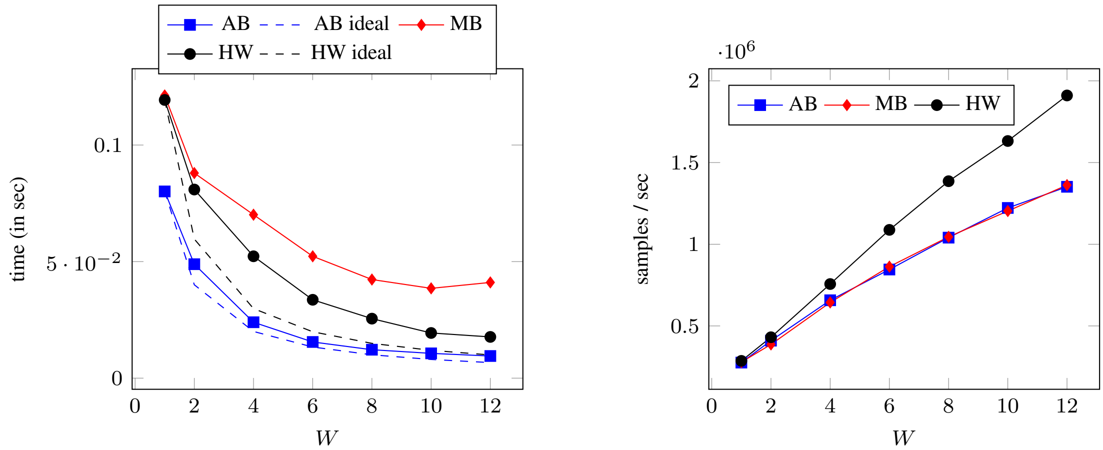


Figure 3: **Left**: time to achieve a given test error when varying the number of workers on *news20*. The dashed line is the ideal speedup. **Right**: number of process sampled per second as a function of W.

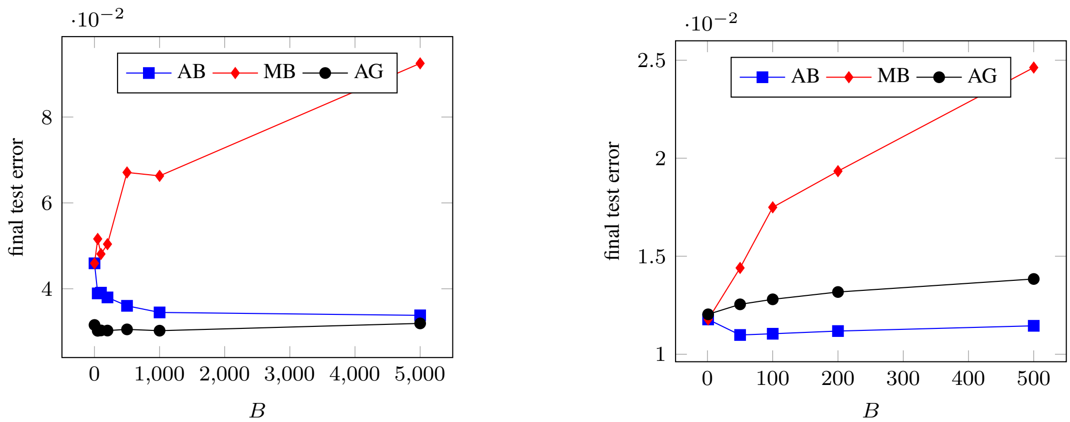


Figure 4: Comparison of Adagrad (AG), AdaBatch (AB) and regular mini-batch (MB). **Left**: on *url* for the log loss; **right**: on *spam* for the least-mean-square loss. We plot the final test error after the same number of samples (larger batches means less iterations).

Supplementary material

Introduction

We present in Section A the pseudocode for AdaBatch and Wild AdaBatch. In Section B, we give two lemma from which we can derive the expectation and variance of the AdaBatch gradient update. In Section C we study the convergence of regular mini-batch and AdaBatch for SGD as well as the convergence of reconditionned SGD. In Section D we compare the convergence of regular mini-batch and AdaBatch for SVRG. Finally in Section E we give convergence plots for Wild AdaBatch and SVRG on the remaining datasets that were not included in the main paper.

A. Algorithms

We present two possible uses of AdaBatch. Algorithm 1 counts for each mini-batch the number of time each feature is non zero and use that to recondition the gradient. This is the algorithm that we study in Section 2 of the main paper.

Algorithm 2 is an Hogwild! inspired synchronous SGD method that we introduce in Section 4 of the main paper. Instead of counting the features, we directly use the reconditioning $\frac{1-(1-p^{(k)})^B}{p}$ where B is the batch size and $\forall k \in [d], p^{(k)} = \mathbb{P}\left\{k \in \mathcal{S}\left(f\right)\right\}$ i.e., the probability that feature k is active in a random training sample. We prove in section C.2 that this reconditioning benefit from the same convergence speed as regular AdaBatch and does not require to keep count of the features which is easier to implement in the parallel setting, although it requires to precompute the probabilities $p^{(k)}$.

Algorithm 1 AdaBatch

function ADABATCH(w_0, N, B, γ, f)
  for $n \in [N]$ **do**
    for $b \in [B]$ **do**
      Sample $f_{n,b}$ from the distribution of f
      Compute $f'_{n,b}(w)$
    end for
    for $b \in [B]$ **do**
      for $k \in \mathcal{S}\left(f_{n,b}\right)$ **do**

$$w_n^{(k)} \leftarrow w_{n-1}^{(k)} - \gamma \frac{f'_{n,b}(w_{n-1})^{(k)}}{\left|\left\{b : k \in \mathcal{S}\left(f_{n,b}\right)\right\}\right|} \tag{7.1}$$

      end for
    end for
  end for
end function

Algorithm 2 Wild AdaBatch

function WILD ADABATCH(w_0, N, B, γ, p, f)
  for $n \in [N]$ **do**
    parallel for $b \in [B]$ **do**
      Sample $f_{n,b}$ from the distribution of f
      Compute $f'_{n,b}(w)$
    end parallel for
    parallel for $b \in [B]$ **do**
      for $k \in \mathcal{S}\left(f_{n,b}\right)$ **do**

$$w_n^{(k)} \leftarrow w_{n-1}^{(k)} - \frac{\gamma}{B} \frac{1-(1-p^{(k)})^B}{p} f'_{n,b}(w_{n-1})^{(k)} \tag{7.2}$$

      end for
    end parallel for
  end for
end function

B. Expectation and variance of the AdaBatch update

The AdaBatch update $g_{\mathrm{ab},n}$ is defined as

$$\forall k \in [d], g_{\mathrm{ab},n}^{(k)} = \begin{cases} \frac{\sum_{b:k\in\mathcal{S}(f)} f'_{n,b}(w_{n-1})^{(k)}}{\sum_{b:k\in\mathcal{S}(f)} 1} & \text{if } \sum_{b:k\in\mathcal{S}(f)} 1 \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

so that we have the recurrence rule for SGD

$$w_n = w_{n-1} - \gamma g_{\mathrm{ab},n}.$$

$g_{\mathrm{ab},n}$ is a per coordinate stochastic average of only the subset of the gradients which have a non zero coordinate in that direction. We will need the following Lemma in order to get the expected value and the variance of $g_{\mathrm{ab},n}$.

Lemma 1. *Let Z, $(Z_i)_{i\in[N]}$ N i.i.d. random variables with value in $\mathbb{R}$ for which the set $\{0\}$ is measurable with $p = \mathbb{P}\{Z \neq 0\} > 0$, and $A \in \mathbb{R}$ a random variable defined as*

$$\begin{cases} A := 0 \quad \textit{if} \quad \forall i \in [N], Z_i = 0 \\ A := \frac{\sum_{i\in[N]} Z_i}{\sum_{i\in[N]:Z_i!=0} 1} \quad \textit{otherwise.} \end{cases} .$$

We have

$$\mathbb{E}[A] = \frac{1-(1-p)^N}{p}\mathbb{E}[Z], \tag{7.4}$$

$$\mathbb{E}\left[A^2\right] = \frac{(1-(1-p)^N)^2}{p^2}\mathbb{E}[Z]^2 + \left(\sum_{i\in[N]}\binom{N}{i}p^i(1-p)^{N-i}\frac{1}{i}\right)\left(\frac{\mathbb{E}\left[Z^2\right]}{p} - \frac{\mathbb{E}[Z]^2}{p^2}\right) \tag{7.5}$$

$$\leq \frac{(1-(1-p)^N)^2}{p^2}\mathbb{E}[Z]^2 + \frac{(1-(1-p)^N)}{p}\mathbb{E}\left[Z^2\right]. \tag{7.6}$$

Proof. We introduce μ the measure of Z and μ^+ the measure of Z^+ defined for any measurable $A \subset \mathbb{R}$

$$\mu^+(A) = \frac{\mu(A\setminus\{0\})}{\mu(\mathbb{R}\setminus\{0\})}.$$

Intuitively Z^+ is the random variable we obtain if we drop all realizations where $Z = 0$. One can verify that $\mathbb{E}[Z^+] = \frac{\mathbb{E}[Z]}{p}$ and $\mathbb{E}\left[(Z^+)^2\right] = \frac{\mathbb{E}[Z^2]}{p}$.

Taking $Q \sim \mathcal{B}(p)$ a Bernoulli of parameter p independent from Z^+, one can readily notice that $Z \sim QZ^+$. We thus take N i.i.d. such copies $(Q_i, Z_i^+)_{i\in[N]}$. Let us take any $q \in \{0,1\}^N$ so that $|q| := \sum_{i\in[N]} q_i > 0$,

$$\begin{aligned}\mathbb{E}[A|Q=q] &= \mathbb{E}\left[\frac{\sum_{i\in[N]} q_i Z_i^+}{|q|}\right] \\ &= \frac{\sum_{i:q_i=1}\mathbb{E}\left[Z_i^+\right]}{|q|} \\ &= \mathbb{E}\left[Z^+\right].\end{aligned}$$

Given that $\mathbb{E}\left[A|\sum_{i\in[N]} Q_i = 0\right] = 0$ and that $\mathbb{P}\left\{\sum_{i\in[N]} Q_i \neq 0\right\} = 1-(1-p)^N$, we get (7.4).

We will denote $\mathbb{V}[A|Q=q] = \mathbb{E}\left[A^2|Q=q\right] - \mathbb{E}[A|Q=q]^2$. We study

$$\begin{aligned}\mathbb{E}\left[A^2|Q=q\right] &= \mathbb{V}[A|Q=q] + \mathbb{E}[A|Q=q]^2 \\ &= \mathbb{V}\left[\frac{\sum_{i:q_i=1} Z_i^+}{|q|}\right] + \frac{\mathbb{E}[Z]^2}{p^2} \\ &= \frac{\mathbb{V}[Z^+]}{|q|} + \frac{\mathbb{E}[Z]^2}{p^2} \\ &= \frac{\mathbb{E}\left[Z^2\right]}{p\,|q|} - \frac{\mathbb{E}[Z]^2}{p^2\,|q|} + \frac{\mathbb{E}[Z]^2}{p^2}.\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}\left[A^2\right] &= \sum_{k\in[N]} \mathbb{E}\left[A^2 | \, |Q| = k\right] \mathbb{P}\left\{|Q| = k\right\} \\
&= \sum_{k\in[N]} \binom{N}{k} p^k (1-p)^{N-k} \frac{1}{k} \left(\frac{\mathbb{E}\left[Z^2\right]}{p} - \frac{\mathbb{E}\left[Z\right]^2}{p^2} \right) + \frac{\mathbb{E}\left[Z\right]^2}{p^2} (1-(1-p)^N) \\
&\leq (1-(1-p)^N) \frac{\mathbb{E}\left[Z^2\right]}{p} + (1-(1-p)^N) \frac{\mathbb{E}\left[Z\right]^2}{p^2},
\end{aligned}
$$

as $\sum_{k\in[N]} \binom{N}{k} p^k (1-p)^{N-k} \frac{1}{k} \leq 1-(1-p)^N$ which gives us (7.5) and conclude this proof. □

Thanks to Lemma 1 we get

$$
\forall k \in [d], \mathbb{E}\left[g_{\mathrm{ab},n}^{(k)}\right] = \frac{1-(1-p^{(k)})^B}{p^{(k)}} \mathbb{E}\left[F'(w_{n-1})\right] \tag{7.7}
$$

$$
\forall k \in [d], \mathbb{E}\left[\left(g_{\mathrm{ab},n}^{(k)}\right)^2\right] \leq \frac{(1-(1-p^{(k)})^N)}{p} \mathbb{E}\left[\left(f'(w_{n-1})^{(k)}\right)^2\right] + \frac{(1-(1-p^{(k)})^N)}{p^2} \left\|F'(w_{n-1})^{(k)}\right\|^2. \tag{7.8}
$$

We now present an improved bound for the second order moment of Z that can be better if than the previous one in the case where Np is large enough. Although we will not provide a full proof of convergence using this result for simplicity, we will comment on how this impact convergence in the proof of theorem 2.

Lemma 2. *With the same notation as in lemma 1, if $Np \geq 5$ we have*

$$
\mathbb{E}\left[A^2\right] \leq \frac{5(1-(1-p)^N)\mathbb{E}\left[Z^2\right]}{Np^2} + \frac{(1-(1-p)^N)\mathbb{E}\left[Z\right]^2}{p^2}. \tag{7.9}
$$

Proof. We reuse the notation from the proof of Lemma 1. Let us define $M := |Q|$ which follows a binomial law of parameter N and p. Using Chernoff's inequality, we have for any $k \leq Np$,

$$
\mathbb{P}\left\{M \leq k\right\} \leq \exp\left(-\frac{(Np-k)^2}{2Np}\right),
$$

taking $k = \frac{Np}{2}$ we obtain

$$
\mathbb{P}\left\{M \leq \frac{Np}{2}\right\} \leq \exp\left(-\frac{Np}{8}\right).
$$

We have

$$
\begin{aligned}
\mathbb{E}\left[\frac{1}{M} | M > 0\right] &\leq \mathbb{P}\left\{M \leq \frac{Np}{2} | M > 0\right\} + \frac{2}{Np} \\
&= \frac{\mathbb{P}\left\{M \leq \frac{Np}{2}\right\}}{\mathbb{P}\left\{M > 0\right\}} + \frac{2}{Np} \\
&\leq \frac{\exp\left(-\frac{Np}{8}\right)}{1-(1-p)^N} + \frac{2}{Np}.
\end{aligned}
$$

We have as $p \geq 5/N$ and using standard analysis techniques,

$$
\frac{\exp\left(-\frac{Np}{8}\right)}{1-(1-p)^N} \leq \frac{3}{Np}.
$$

We obtain

$$
\mathbb{E}\left[\frac{1}{M} | M > 0\right] \leq \frac{5}{Np}.
$$

Plugging this result into (7.5), we immediately have

$$
\mathbb{E}\left[A^2\right] \leq \frac{5(1-(1-p)^N)\mathbb{E}\left[Z^2\right]}{Np^2} + \frac{(1-(1-p)^N)\mathbb{E}\left[Z\right]^2}{p^2}.
$$

□

C. Proof of convergence of AdaBatch and mini-batch SGD

C.1. Constant step size SGD with mini-batch

We will first give a convergence result for the regular mini-batch SGD, which is adapted from [21].

Assumption 2. *We assume there exists a convex compact set $\mathcal{D} \subset \mathbb{R}^d$ so that f and F verifies the following assumptions for μ, L and R strictly positive,*

1. *The hessian F'' of F is bounded from above and below as:*
$$\forall w \in \mathcal{D}, \mu\,\mathrm{Id} \preceq F''(w) \preceq L\,\mathrm{Id}, \tag{7.10}$$
with $\mu > 0$ so that f is μ strongly convex and L smooth over $\mathcal{D}$.

2. *We assume f'' is almost surely bounded,*
$$\forall w \in \mathcal{D}, f''(w) \preceq R^2\,\mathrm{Id}\,. \tag{7.11}$$

3. *Let $w_* := \arg\min_{w\in\mathcal{D}} F(w)$,*
$$F'(w_*) = 0, \tag{7.12}$$
which means in particular that w_* is a global minimizer of F over $\mathbb{R}^d$.

This does not limit us to the case of *globally* strongly convex functions F as we only require it to be strongly convex on a compact subset that contains the global optimum w_*. In practice, this is often going to be the case, even when using a non strictly convex loss such as the logistic loss as soon as the problem is not perfectly separable, i.e., there is no hyperplane that perfectly separates the classes we are trying to predict.

We will now study the recursion for a given $w_0 \in \mathbb{R}^d$ given by

$$\forall n > 0, w_n = \Pi_{\mathcal{D}}\left[w_{n-1} - \frac{\gamma}{B}\sum_{b\in[B]} f'_{n,b}(w_{n-1})\right], \tag{7.13}$$

where $\Pi_{\mathcal{D}}[w] := \arg\min_{x\in\mathcal{D}} \|w-x\|^2$ is the orthogonal projection on the set $\mathcal{D}$. This extra step of projection is required for this proof technique but experience shows that it is not needed.

Theorem 1 (Convergence of $F_n - F_*$ for SGD with mini-batch)**.** *If Assumptions 2 are verified and*

$$\gamma\left[L\left(1-\frac{1}{B}\right) + \frac{2R^2}{B}\right] \leq 1, \tag{7.14}$$

then for any $N > 0$,

$$\|w_N - w_*\|^2 \leq (1-\gamma\mu/2)^N \|w_0 - w_*\|^2 + \frac{4\gamma}{B\mu}\mathbb{E}\left[\|f'(w_*)\|^2\right], \tag{7.15}$$

and introducing

$$\bar{w}_N = \frac{\sum_{n\in[N]}(1-\gamma\mu/2)^{N-n} w_n}{\sum_{n\in[N]}(1-\gamma\mu/2)^{N-n}},$$

we have

$$\mathbb{E}\left[F(\bar{w}_N)\right] - F_* \leq \gamma^{-1}(1-\gamma\mu/2)^N \|w_0 - w_*\|^2 + \frac{2\gamma}{B}\mathbb{E}\left[\|f'(w_*)\|^2\right]. \tag{7.16}$$

Introducing $\bar{w}_N$ allows for an easier comparison directly on the objective function. This is made for qualitative analysis and we do not in practice perform this averaging.

We can see that the error given by (7.16) can be composed in two terms, one that measure how quickly we move away from the starting point and the second that depends on the stochastic noise around the optimum. We will call the former the *bias* term and the latter the *variance* term, following the terminology introduced by [18].

Proof. We introduce $\forall n \in [N], \mathcal{F}_{n-1}$ the σ-field generated by $(f_{i,b})_{i\in[n-1],b\in[B]}$. Let us take $n \in [N]$ and introduce $\eta_n := w_n - w_*$ and $g_n := \frac{1}{B}\sum_{b\in[B]} f'_{n,b}(w_{n-1})$. We then proceed to bound $\|\eta_n\|^2$,

$$\begin{aligned}\|\eta_n\|^2 &\leq \|\eta_{n-1} - \gamma g_n\|^2 \quad \text{as } \Pi_{\mathcal{D}} \text{ is contractant for } \|\cdot\| \\ &= \|\eta_{n-1}\|^2 - 2\gamma g_n^T \eta_{n-1} + \gamma^2 \|g_n\|^2\,.\end{aligned}$$

Taking the expectation while conditioning on $\mathcal{F}_{n-1}$ we obtain

$$\mathbb{E}\left[\|\eta_n\|^2 \,|\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|^2 - 2\gamma F'(w_{n-1})^T\eta_{n-1} + \gamma^2\mathbb{E}\left[\|g_n\|^2 \,|\mathcal{F}_{n-1}\right], \tag{7.17}$$

$$\mathbb{E}\left[\|g_n\|^2 \,|\mathcal{F}_{n-1}\right] = \frac{\mathbb{E}\left[\|f'(w_{n-1})\|^2\right]}{B} + \|F'(w_{n-1})\|^2\left(1 - \frac{1}{B}\right).$$

Injecting this in (7.17) gives us

$$\mathbb{E}\left[\|\eta_n\|^2 \,|\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|^2 - 2\gamma F'(w_{n-1})^T\eta_{n-1} + \frac{\gamma^2}{B}\mathbb{E}\left[\|f'(w_{n-1})\|^2\right] + \gamma^2\|F'(w_{n-1})\|^2\left(1 - \frac{1}{B}\right). \tag{7.18}$$

As $F'' \preceq L\,\mathrm{Id}$ and using the co-coercivity of F' we have

$$\begin{aligned}
\|F'(w_{n-1})\|^2 &= \|F'(w_{n-1}) - F'(w_*)\|^2 \\
&\leq L(F'(w_{n-1}) - F'(w_*))^T(w_{n-1} - w_*) \\
&= LF'(w_{n-1})^T(w_{n-1} - w_*).
\end{aligned}$$

We have

$$\|f'(w_{n-1})\|^2 \leq 2\|f'(w_{n-1}) - f'(w_*)\|^2 + 2\|f'(w_*)\|^2 \leq 2R^2(f'(w_{n-1}) - f'(w_*))^T(w_{n-1} - w_*) + 2\|f'(w_*)\|^2$$

and

$$\mathbb{E}\left[\|f'(w_{n-1})\|^2 \,|\mathcal{F}_{n-1}\right] \leq 2R^2F'(w_{n-1})^T(w_{n-1} - w_*) + 2\|f'(w_*)\|^2.$$

Injecting in (7.18) we get

$$\mathbb{E}\left[\|\eta_n\|^2 \,|\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|^2 - \gamma F'(w_{n-1})^T\eta_{n-1}\underbrace{\left(2 - \gamma L\left(1 - \frac{1}{B}\right) - \frac{2\gamma R^2}{B}\right)}_{A} + \frac{2\gamma^2\mathbb{E}\left[\|f'(w_*)\|^2\right]}{B}.$$

We want A to be large enough, we will take

$$\gamma\left[L\left(1 - \frac{1}{B}\right) + \frac{2R^2}{B}\right] \leq 1, \tag{7.19}$$

which gives us $A \geq 1$ and

$$\mathbb{E}\left[\|\eta_n\|^2 \,|\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|^2 - \gamma F'(w_{n-1})^T\eta_{n-1} + \frac{2\gamma^2\mathbb{E}\left[\|f'(w_*)\|^2\right]}{B}.$$

As F is μ strictly convex, we have

$$F_* - F(w_{n-1}) \geq F'(w_{n-1})^T(w_* - w_{n-1}) + \frac{\mu}{2}\|\eta_{n-1}\|^2,$$

which allows to obtain

$$\mathbb{E}\left[\|\eta_n\|^2 \,|\mathcal{F}_{n-1}\right] \leq (1 - \gamma\mu/2)\|\eta_{n-1}\|^2 - \gamma(F(w_{n-1}) - F_*) + \frac{2\gamma^2}{B}\mathbb{E}\left[\|f'(w_*)\|^2\right]. \tag{7.20}$$

Taking the full expectation gives us

$$\begin{aligned}
\mathbb{E}\left[\|\eta_n\|^2\right] &\leq (1 - \gamma\mu/2)\mathbb{E}\left[\|\eta_{n-1}\|^2\right] - \gamma(\mathbb{E}\left[F(w_{n-1})\right] - F_*) + \frac{2\gamma^2\mathbb{E}\left[\|f'(w_*)\|^2\right]}{B}, \\
&\leq (1 - \gamma\mu/2)^n\|\eta_0\|^2 + \frac{2\gamma^2\mathbb{E}\left[\|f'(w_*)\|^2\right]}{B}\sum_{0\leq i<n}(1 - \gamma\mu/2)^i \\
&\leq (1 - \gamma\mu/2)^n\|\eta_0\|^2 + \frac{4\gamma}{B\mu}\mathbb{E}\left[\|f'(w_*)\|^2\right],
\end{aligned}$$

which gives us (7.15).

Let us now take $\alpha := (1-\gamma\mu/2)$, let us call $u_n := \mathbb{E}\left[\|\eta_n\|^2\right]$, we have using (7.20),

$$\begin{aligned}\gamma\delta_{n-1} &\leq \alpha u_{n-1} - u_n + 2\gamma^2 R^2\\ \gamma\delta_{n-1}\alpha^{-n} &\leq \alpha^{-n+1}u_{n-1} - u_n\alpha^{-n} + 2\gamma^2 R^2\alpha^{-n},\end{aligned}$$

summing for n from 1 to N we obtain

$$\gamma \sum_{n\in[N]} \delta_{n-1}\alpha^{-n} \leq u_0 - u_N\alpha^{-N} + \frac{2\gamma^2}{B}\mathbb{E}\left[\|f'(w_*)\|^2\right]\sum_{n\in[N]}\alpha^{-n},$$

dividing by $\sum_{n\in[N]}\alpha^{-n}$ on each side and using the convexity of F we get

$$\mathbb{E}\left[F(\bar{w}_{N-1})\right] - F_* \leq \alpha^N u_0 + \frac{2\gamma^2}{B}\mathbb{E}\left[\|f'(w_*)\|^2\right],$$

which gives us (7.16) and concludes this proof. □

C.2. Convergence of reconditioned SGD

Let us now assume that we have for some matrices T and C definite positive so that

$$\begin{aligned}&\forall w \in \mathcal{D}, \mu T \preceq F''(w) \preceq LT,\\ &\forall w \in \mathcal{D}, f''(w) \preceq L\,\mathrm{Id} \quad a.s.\end{aligned}$$

We now study w_n defined by the following recurence

$$w_n = w_{n-1} - \frac{\gamma}{B}C\sum_{b\in[B]} f'_{n,b}(w_{n-1}). \tag{7.21}$$

First let us introduce $v_0 := \sqrt{C}^{-1}w_0$, $v_* := \sqrt{C}^{-1}w_*$ and $h(w) := f(\sqrt{C}w)$ as well as $\forall n \in [N], b \in [B], h_{n,b}(w) := f_{n,b}(\sqrt{C}w)$, then we define

$$v_n := v_{n-1} - \frac{\gamma}{B}\sum_{b\in[B]} h'_{m,b}(v_{n-1}).$$

Multiplying by $\sqrt{C}$ we recover the same recurrence rule as (7.21) for $w_n = \sqrt{C}v_n$. Therefore, the convergence of v_n will give us the convergence of w_n.

Let us take $H(w) := \mathbb{E}\left[h(w)\right] = F(\sqrt{C}w)$. By definition we have

$$\begin{aligned}&\forall w \in \mathcal{D}_C, \mu\sqrt{C}T\sqrt{C} \preceq F''(w) \preceq L\sqrt{C}T\sqrt{C},\\ &\forall w \in \mathcal{D}_C, h''(w) \preceq R^2 L_C\,\mathrm{Id} \quad a.s.,\end{aligned}$$

where $\mathcal{D}_C = \sqrt{C}^{-1}\mathcal{D}$, L_C is the largest eigen value of C. If we take $\mu_{C,T}$ (resp $L_{C,T}$) the smallest (resp largest) eigenvalue of $\sqrt{C}T\sqrt{C}$, then using Theorem 1, we have for

$$\gamma\left[LL_{C,T}\left(1-\frac{1}{B}\right) + \frac{2L_C R^2}{B}\right] \leq 1, \tag{7.22}$$

$$\|v_N - v_*\|^2 \leq (1-\gamma\mu_{C,T})^N\|v_0 - v_*\|^2 + \frac{2\gamma}{B\mu}\mathbb{E}\left[\|h'(v_*)\|^2\right],$$

and introducing

$$\bar{v}_N = \frac{\sum_{n\in[N]}(1-\gamma\mu_{C,T}/2)^{N-n}v_n}{\sum_{n\in[N]}(1-\gamma\mu_{C,T}/2)^{N-n}},$$

we have

$$\mathbb{E}\left[H(\bar{v}_N)\right] - H_* \leq \gamma^{-1}(1-\gamma\mu_{C,T})^N\|v_0 - v_*\|^2 + \frac{2\gamma L_C R^2}{B}\mathbb{E}\left[\|h'(v_*)\|^2\right].$$

Using $w_n = \sqrt{C}v_n$, we obtain

$$\|w_N - w_*\|^2_{C^{-1}} \leq (1-\gamma\mu_{C,T})^N\|w_0 - w_*\|^2_{C^{-1}} + \frac{4\gamma}{B\mu}\mathbb{E}\left[\|f'(w_*)\|^2_C\right],$$

$$\mathbb{E}\left[F(\bar{w}_N)\right] - F_* \leq \gamma^{-1}(1-\gamma\mu_{C,T})^N\|w_0 - w_*\|^2_{C^{-1}} + \frac{2\gamma L_C}{B}\mathbb{E}\left[\|f'(w_*)\|^2_C\right]. \tag{7.23}$$

Application to sparse optimization. In the sparse setting, we have made the assumption that $T := \mathrm{Diag}\,(p)$. We suggested two reconditioning strategies in such case. The first one is to take $C = \mathrm{Diag}\,(p)^{-1}$. In such case $\mu_{C,T} = L_{C,T} = 1$ so that we have a perfect conditioning. However $L_C = p_{\min}^{-1}$ so that if $B \ll p_{\min}^{-1}$ we would have to take a much smaller step size and the term $\frac{1}{B}\mathbb{E}\left[\|f'(w_*)\|^2\right]_{\mathrm{Diag}(p)^{-1}}$ would explode.

The second one, $C := \mathrm{Diag}\left(\frac{1-(1-p)^B}{p}\right)$ so that $\mu_{C,T} = 1-(1-p_{\min})^B$ and $L_{C,T} = 1-(1-p_{\max})^B$. Because the fonction $p \to 1-(1-p)^B$ increases faster for small probabilities, the conditioning of the problem is improved. If $p_{\max}$ is close to 1 and $p_{\min}$ close to 0, then $\mu_{C,T} \approx Bp_{\min}$ and $L_C \approx p_{\max}$. We have $L_C \leq B$ as $\forall p \in [0,1], (1-(1-p)^B) \leq Bp$, so that the increase due to L_C is perfectly balanced out by the batch size B in (7.22). Besides, as $\frac{C}{B} \preceq \mathrm{Id}$, we have $\frac{1}{B}\mathbb{E}\left[\|f'(w_*)\|_C^2\right] \leq \mathbb{E}\left[\|f'(w_*)\|^2\right]$.

C.3. Convergence of AdaBatch

We now make the following assumptions:

Assumption 3. *We assume there exists a convex compact set $\mathcal{D} \subset \mathbb{R}^d$, μ, L and R strictly positive so that we have the following assumptions verified.*

1. *The Hessian F'' (resp f'') of F (resp f) are bounded from above and below as:*

$$\forall w \in \mathcal{D},\ \ \mu\mathrm{Diag}\,(p) \preceq F''(w) \preceq L\mathrm{Diag}\,(p) \quad \textit{and} \quad \forall w \in \mathcal{D}, f''(w) \preceq R^2\,\mathrm{Id}\,. \tag{7.24}$$

2. *Let $w_* := \arg\min_{w\in\mathcal{D}} F(w)$,*

$$F'(w_*) = 0. \tag{7.25}$$

In particular, w_* is a global minimizer of F over $\mathbb{R}^d$.

We will now study convergence when we use the AdaBatch update. We are given $w_0 \in \mathbb{R}^d$ and we define $C := \mathrm{Diag}\left(\frac{1-(1-p)^B}{p}\right)$ and we define recursively,

$$\begin{aligned}
&\forall k \in [d], g_{\mathrm{ab},n}^k = \begin{cases} \frac{\sum_{b\in B_n^k} f'_{n,b}(w_{n-1})^k}{\sum_{b:k\in\mathcal{S}(f)} 1} & \text{if } \sum_{b:k\in\mathcal{S}(f)} 1 \neq 0 \\ 0 & \text{otherwise} \end{cases}\\
&w_n = \Pi_{\mathcal{D},C}\left[w_{n-1} - \gamma g_{\mathrm{ab},n}\right],
\end{aligned}$$

where $\Pi_{\mathcal{D},C}[w] := \arg\min_{x\in\mathcal{D}} \|w-x\|_{C^{-1}}^2$ is the projection on the set $\mathcal{D}$ with respect to $\|\cdot\|_{C^{-1}}$. This extra step of projection is required for this proof technique but experience shows that it is not needed. We introduce $\forall p \in [0,1], p^{+B} := 1-(1-p)^B$.

Theorem 2 (Convergence of $F_n - F_*$ for AdaBatch)**.** *If Assumptions 3 are verified and*

$$\gamma\left(L + 2R^2\right) \leq 1, \tag{7.26}$$

then for any $N > 0$,

$$\|w_N - w_*\|^2 \leq (1-\gamma p_{\min}^{+B}\mu/2)^N \|w_0 - w_*\|^2 + \frac{4\gamma}{\mu}\mathbb{E}\left[\|f'(w_*)\|^2\right], \tag{7.27}$$

and introducing

$$\bar{w}_N = \frac{\sum_{n\in[N]}(1-\gamma p_{\min}^{+B}\mu/2)^{N-n} w_n}{\sum_{n\in[N]}(1-\gamma p_{\min}^{+B}\mu/2)^{N-n}},$$

we have

$$\mathbb{E}\left[F(\bar{w}_N)\right] - F_* \leq \gamma^{-1}(1-\gamma p_{\min}^{+B}\mu/2)^N \|w_0 - w_*\|^2 + 2\gamma\mathbb{E}\left[\|f'(w_*)\|^2\right]. \tag{7.28}$$

Proof. We introduce $\forall n \in [N], \mathcal{F}_{n-1}$ the σ-field generated by $(f_{i,b})_{i\in[n-1],b\in[B]}$. Let us take $n \in [N]$ and introduce $\eta_n := w_n - w_*$. We then proceed to bound $\|\eta_n\|_{C^{-1}}^2$,

$$\begin{aligned}
\|\eta_n\|_{C^{-1}}^2 &\leq \|\eta_{n-1} - \gamma g_n\|_{C^{-1}}^2 \quad \text{ as } \Pi_{\mathcal{D}} \text{ is contractant for } \|\cdot\|_{C^{-1}}\\
&= \|\eta_{n-1}\|_{C^{-1}}^2 - 2\gamma g_n^T\eta_{n-1} + \gamma^2\|g_n\|_{C^{-1}}^2\,.
\end{aligned}$$

Taking the expectation while conditioning on $\mathcal{F}_{n-1}$ we obtain

$$\mathbb{E}\left[\|\eta_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|_{C^{-1}}^2 - 2\gamma F'(w_{n-1})^T \eta_{n-1} + \gamma^2 \mathbb{E}\left[\|g_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right]. \tag{7.29}$$

Using Lemma 1,

$$\mathbb{E}\left[\|g_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right] \leq \mathbb{E}\left[\|f'(w_{n-1})\|^2\right] + \|F'(w_{n-1})\|_{\mathrm{diag}(p)^{-1}}^2. \tag{7.30}$$

Although we will not do it in the following, it is also possible to use Lemma 2. Indeed, for any $k \in [d]$ such that $Bp^{(k)} \geq 5$, we have

$$\mathbb{E}\left[\left(g_n^{(k)}\right)^2 C_{k,k}^{-1} |\mathcal{F}_{n-1}\right] \leq \frac{5\mathbb{E}\left[(f'(w_{n-1})^{(k)})^2\right]}{Np} + \frac{(F'(w_{n-1})^{(k)})^2}{p^{(k)}},$$

which would be equivalent for the dimension k to having a regular batch size of $\frac{p^{(k)}B}{5}$. This shows that AdaBatch will benefit from a reduced variance for features that are frequent enough. For simplicity we will however stick with the simpler bound given by (7.30).

As $F'' \preceq L\mathrm{diag}\,(p)$ and using the co-coercivity of F' we have

$$\begin{aligned}
\|F'(w_{n-1})\|_{\mathrm{diag}(p)^{-1}}^2 &= \|F'(w_{n-1}) - F'(w_*)\|_{\mathrm{diag}(p)^{-1}}^2 \\
&\leq L(F'(w_{n-1}) - F'(w_*))^T (w_{n-1} - w_*) \\
&= LF'(w_{n-1})^T (w_{n-1} - w_*).
\end{aligned}$$

Injecting this in (7.29) gives us

$$\mathbb{E}\left[\|\eta_n\|^2 |\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|_{C^{-1}}^2 - 2\gamma F'(w_{n-1})^T \eta_{n-1} + \gamma^2 \mathbb{E}\left[\|f'(w_{n-1})\|^2\right]. \tag{7.31}$$

We have

$$\|f'(w_{n-1})\|^2 \leq 2\|f'(w_{n-1}) - f'(w_*)\|^2 + 2\|f'(w_*)\|^2 \leq 2R^2(f'(w_{n-1}) - f'(w_*))^T (w_{n-1} - w_*) + 2\|f'(w_*)\|^2$$

and

$$\mathbb{E}\left[\|f'(w_{n-1})\|^2 |\mathcal{F}_{n-1}\right] \leq 2R^2 F'(w_{n-1})^T (w_{n-1} - w_*) + 2\|f'(w_*)\|^2.$$

Injecting in (7.31) we get

$$\mathbb{E}\left[\|\eta_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|_{C^{-1}}^2 - \gamma F'(w_{n-1})^T \eta_{n-1} \underbrace{(2 - \gamma L - 2\gamma R^2)}_{A} + 2\gamma^2 \mathbb{E}\left[\|f'(w_*)\|^2\right].$$

We want A to be large enough, we will take

$$\gamma\left(L + 2R^2\right) \leq 1, \tag{7.32}$$

which gives us $A \geq 1$ and

$$\mathbb{E}\left[\|\eta_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right] \leq \|\eta_{n-1}\|_{C^{-1}}^2 - \gamma F'(w_{n-1})^T \eta_{n-1} + 2\gamma^2 \mathbb{E}\left[\|f'(w_*)\|^2\right].$$

As $F'' \succeq \mu\mathrm{Diag}\,(p)$ we have

$$\begin{aligned}
F_* - F(w_{n-1}) &\geq F'(w_{n-1})^T (w_* - w_{n-1}) + \frac{\mu}{2}\|\eta_{n-1}\|_{\mathrm{Diag}(p)}^2 \\
&\geq F'(w_{n-1})^T (w_* - w_{n-1}) + \frac{p_{\min}^{+B}\mu}{2}\|\eta_{n-1}\|_{C^{-1}}^2,
\end{aligned}$$

which allows to obtain

$$\mathbb{E}\left[\|\eta_n\|_{C^{-1}}^2 |\mathcal{F}_{n-1}\right] \leq (1 - \gamma\mu p_{\min}^{+B}/2)\|\eta_{n-1}\|_{C^{-1}}^2 - \gamma(F(w_{n-1}) - F_*) + 2\gamma^2 \mathbb{E}\left[\|f'(w_*)\|^2\right]. \tag{7.33}$$

Taking the full expectation gives us

$$\begin{aligned}
\mathbb{E}\left[\|\eta_n\|_{C^{-1}}^2\right] &\leq (1 - \gamma\mu p_{\min}^{+B}/2)\mathbb{E}\left[\|\eta_{n-1}\|_{C^{-1}}^2\right] - \gamma(\mathbb{E}\left[F(w_{n-1})\right] - F_*) + 2\gamma^2 \mathbb{E}\left[\|f'(w_*)\|^2\right], \\
&\leq (1 - \gamma\mu p_{\min}^{+B}/2)^n \|\eta_0\|_{C^{-1}}^2 + 2\gamma^2 \mathbb{E}\left[\|f'(w_*)\|^2\right] \sum_{0 \leq i < n} (1 - \gamma\mu p_{\min}^{+B}/2)^i \\
&\leq (1 - \gamma\mu p_{\min}^{+B}/2)^n \|\eta_0\|_{C^{-1}}^2 + \frac{4\gamma}{\mu}\mathbb{E}\left[\|f'(w_*)\|^2\right],
\end{aligned}$$

which gives us (7.27). We obtain (7.28) in the exact same way as in the proof of Theorem 1. □

C.4. Sparse linear prediction

We will now show that Assumption 3 is easy to meet in the case of linear predictions. For simplicity, let us assume X is a random variable with values in $\{0,1\}^d$ with uncorrelated features, i.e.,

$$\forall k, k' \in [d] : k \neq k', \mathbb{E}\left[X^{(k)}X^{(k')}\right] = \mathbb{E}\left[X^{(k)}\right] X^{(k')},$$

and $\phi : \mathbb{R} \to \mathbb{R}$ a random convex function. Then one can take $f(w) := \phi(X^T w)$. For m, M and G strictly positive and $\mathcal{D} \subset \mathbb{R}^d$ a convex compact, We assume almost surely we have

$$\begin{aligned}
&\forall w \in \mathcal{D}, m \leq \phi''(w) \leq M, \\
&\|X\|^2 \leq G^2 \quad a.s.
\end{aligned}$$

Then, for any $w \in \mathcal{D}$ we have

$$\begin{aligned}
F''(w) &= \mathbb{E}\left[f''(w)\right] \\
&= \mathbb{E}\left[\phi''(w)XX^T\right] \\
&\preceq M\mathbb{E}\left[XX^T\right] \\
&= M\left(\mathrm{Diag}\,(p) - \mathrm{Diag}\,(p)^2 + pp^T\right).
\end{aligned}$$

Moreover, we have

$$pp^T = \sqrt{\mathrm{Diag}\,(p)}\left(\sqrt{p}\sqrt{p}^T\right)\sqrt{\mathrm{Diag}\,(p)}. \tag{7.34}$$

As $\sqrt{p}\sqrt{p}^T \preceq \left\|\sqrt{p}\right\|^2 \mathrm{Id} = \sum_{k\in[d]} p^{(k)}\,\mathrm{Id}$, we obtain

$$F''(w) \preceq M\left(1 + \sum_{k\in[d]} p^{(k)}\right)\mathrm{Diag}\,(p).$$

Besides, we have

$$\begin{aligned}
F''(w) &\succeq m\left(\mathrm{Diag}\,(p) - \mathrm{Diag}\,(p)^2\right) \\
&\succeq m(1 - p_{\max})\mathrm{Diag}\,(p).
\end{aligned}$$

Finally,

$$\begin{aligned}
f''(w) &= \phi''(w)XX^T \\
&\preceq MG^2.
\end{aligned}$$

As a conclusion, Assumptions 3 are verified for $\mu := m(1 - p_{\max})$, $L := M\left(1 + \sum_{k\in[d]} p^{(k)}\right)$ and $R^2 := G^2 M$.

D. AdaBatch for SVRG

D.1. Mini-batch SVRG

We now only assume that F verifies the following inequalities for $\mu > 0$,

$$\forall w \in \mathbb{R}^d,\ \ \mu \preceq F''(w) \quad \text{and} \quad f(w) \preceq L \text{ a.s.} \tag{7.35}$$

Let us take a starting point $y_0 \in \mathbb{R}^d$ and $m \in \mathbb{N}^*$. For all $s = 0, 1, \dots$ we have $w_{s,0} := y_s$ and for all $n \in [m]$ let us define

$$\begin{aligned}
w_{s,n} &:= w_{s,n-1} - \gamma g_{s,n}, \\
y_{s+1} &:= \frac{1}{m}\sum_{n\in[m]} w_{s,n}.
\end{aligned}$$

with $g_{s,n}$ the SVRG update based on $(f_{s,n,b})_{b\in[B]}$ i.i.d samples of f. Let us introduce

$$\forall k \in [d], D_{s,n}^{(k)} := \left\{b \in [B] : k \in \mathcal{S}\left(f'_{s,n,b}\right)\right\}.$$

For any dimension $k \in [d]$ we have

$$g_{s,n}^{(k)} := \frac{1}{B}\left(\sum_{b\in D_{s,n}^{(k)}} f'_{s,n,b}(w_{s,n-1})^{(k)} - f'_{s,n,b}(y_s)^{(k)} + F'(y_s)^{(k)}/p^{(k)}\right).$$

Theorem 3 (Convergence of SVRG with mini-batch)**.** *If Assumptions 7.35 are verified and γ verifies*

$$\gamma L\left(1-\frac{1}{B}\right)<1,$$

then for all $s>0$ we have

$$\mathbb{E}\left[F(y_s)-F_*\right]\leq\alpha^s(F(y_0)-F_*) \tag{7.36}$$

where

$$\alpha:=\frac{1}{\mu\gamma\left(1-\frac{\gamma L(3+B)}{2B}\right)m}+\frac{2L\gamma}{B\left(1-\gamma L\frac{(3+B)}{2B}\right)}. \tag{7.37}$$

Proof. We will reuse the proof technique from [27, section 6.3]. We introduce $\forall n\in[N],\mathcal{F}_{s,n-1}$ the σ-field generated by $(f_{u,i,b})_{u\in[s],i\in[n-1],b\in[B]}$. For simplicity, we will drop all the s indices. We define $\Gamma_{n,b}:=\operatorname{diag}\left(\left(1_{k\in\mathcal{S}(f_{n,b})}/p^{(k)}\right)_{k\in[d]}\right)$ and $\Gamma:=\operatorname{diag}\left(\left(1_{k\in\mathcal{S}(f)}/p^{(k)}\right)_{k\in[d]}\right)$ so that

$$g_n=\frac{1}{B}\left(\sum_{b\in[B]}f'_{n,b}(w_{n-1})-f'_{n,b}(y)+\Gamma_{n,b}F'(y_s)\right).$$

One can immediately notice that

$$\mathbb{E}\left[g_n|\mathcal{F}_{n-1}\right]=F'(w_{n-1}).$$

Besides, we have

$$\mathbb{E}\left[\|f'(w_{n-1})-f'(y)+\Gamma F'(y)\|^2|\mathcal{F}_{n-1}\right]\leq 2\mathbb{E}\left[\|f'(w_{n-1})-f'(w_*)\|^2+\|f'(y)-f'(w_*)+\Gamma F'(y)\|^2|\mathcal{F}_{n-1}\right].$$

We reuse the same proof as in [9, Lemma 10]. Using the fact that $\mathbb{E}\left[(f'(y)-f'(w_*))^T\Gamma F'(y)|\mathcal{F}_{n-1}\right]=\|F'(y)\|^2_{\operatorname{diag}(p)^{-1}}$ and $\mathbb{E}\left[\|\Gamma F'(y)\|^2|\mathcal{F}_{n-1}\right]=\|F'(y)\|^2_{\operatorname{diag}(p)^{-1}}$ we have

$$\begin{aligned}\mathbb{E}\left[A^{(k)}|\mathcal{F}_{n-1}\right]&=\mathbb{E}\left[\|f'(y)-f'(w_*)\|^2|\mathcal{F}_{n-1}\right]-2\|F'(y)\|^2_{\operatorname{diag}(p)^{-1}}+\|F'(y)\|^2_{\operatorname{diag}(p)^{-1}}\\&\leq\mathbb{E}\left[\|f'(y)-f'(w_*)\|^2|\mathcal{F}_{n-1}\right].\end{aligned}$$

It follows that

$$\begin{aligned}\mathbb{E}\left[\|g_n\|^2|\mathcal{F}_{n-1}\right]&=\frac{1}{B}\mathbb{E}\left[\|f'(w_{n-1})-f'(y)+\Gamma F'(y)\|^2|\mathcal{F}_{n-1}\right]+\|F'(w_{t-1})\|^2\left(1-\frac{1}{B}\right)\\&\leq\frac{2}{B}\left(\mathbb{E}\left[\|f'(w_{n-1})-f'(w_*)\|^2+\|f'(y)-f'(w_*)\|^2|\mathcal{F}_{n-1}\right]\right)+\|F'(w_{t-1})\|^2\left(1-\frac{1}{B}\right)\\&\leq\frac{4L}{B}(F(w_{n-1})-F_*+F(y)-F_*)+\|F'(w_{t-1})\|^2\left(1-\frac{1}{B}\right),\end{aligned}$$

using Lemma 6.4 from [27]. We also have

$$\|F'(w_{t-1})\|^2\leq LF'(w_{n-1})^T(w_{n-1}-w_*),$$

so that

$$\begin{aligned}\mathbb{E}\left[\|w_n-w_*\|^2|\mathcal{F}_{n-1}\right]\leq\|w_{n-1}-w_*\|^2&-2\gamma\left(1-\frac{\gamma L}{2}\left(1-\frac{1}{B}\right)\right)F'(w_{n-1})^T(w_{n-1}-w_*)\\&+\frac{4\gamma^2L}{B}(F(w_{n-1})-F_*+F(y)-F_*).\end{aligned}$$

We choose γ so that

$$\gamma L\left(1-\frac{1}{B}\right)<1,$$

and using that $F'(w_{n-1})^T(w_{n-1}-w_*)\geq F(w_{n-1})-F_*$ we have

$$\mathbb{E}\left[\|w_n-w_*\|^2|\mathcal{F}_{n-1}\right]\leq\|w_{n-1}-w_*\|^2-\gamma\left(2-\frac{\gamma L(3+B)}{B}\right)(F(w_{n-1})-F_*)+\frac{4\gamma^2L}{B}(F(y)-F_*).$$

Summing the above inequality for $n \in [m]$ and taking the expectation with respect to $\mathcal{F}_0$,

$$\mathbb{E}\left[\|w_m - w_*\|^2\right] \le \|y - w_*\| - \gamma\left(2 - \frac{\gamma L(3+B)}{B}\right)\mathbb{E}\left[\sum_{n\in[m]} F(w_n) - F_*|\mathcal{F}_0\right] + \frac{4L\gamma^2 m}{B}(F(y) - F_*).$$

Using the strong convexity of F we have $\|w_0 - w_*\|^2 \le \frac{2}{\mu}(F(y) - F_*)$ and finally

$$\mathbb{E}\left[F\left(\frac{1}{m}\sum_{n\in[m]} w_n\right) - F_*|\mathcal{F}_0\right] \le \left(\frac{1}{\mu\gamma\left(1 - \frac{\gamma L(3+B)}{2B}\right)m} + \frac{2L\gamma}{B\left(1 - \gamma L\frac{(3+B)}{2B}\right)}\right)(F(y) - F(w_*)).$$

□

One can derive a simplified convergence result when we assume B large enough.

Corollary 1. *If we assume $B \gg 1$, then with $\gamma = \frac{1}{L}$ and $m = \frac{2BL}{\mu(0.9B-4)} \approx \frac{2.2L}{\mu}$, we have*

$$\mathbb{E}\left[F(y_s) - F_*\right] \le 0.9^s(F(y_0) - F_*).$$

D.2. AdaBatch SVRG

Let us now assume that F verify the following inequalities for $\mu > 0$,

$$\forall w \in \mathbb{R}^d,\ \ \mu\mathrm{diag}\,(p) \preceq F''(w) \quad \text{and} \quad f(w) \preceq L\mathrm{diag}\,(p) \ \text{ a.s.} \tag{7.38}$$

We now define for any dimension k such that $D_{s,n}^{(k)} \neq \emptyset$,

$$g_{s,n}^{(k)} := \frac{1}{\left|D_{s,n}^{(k)}\right|}\left(\sum_{b\in D_{s,n}^{(k)}} f'_{s,n,b}(w_{s,n-1})^{(k)} - f'_{s,n,b}(y_s)^{(k)} + F'(y_s)^{(k)}/p^{(k)}\right),$$

and $g_{s,n}^{(k)} := 0$ otherwise.

Theorem 4 (Convergence of SVRG with AdaBatch)**.** *If Assumptions 7.38 are verified and γ verifies*

$$\gamma < \frac{L}{2},$$

then for all $s > 0$ we have

$$\mathbb{E}\left[F(y_s) - F_*\right] \le \alpha^s(F(y_0) - F_*), \tag{7.39}$$

where

$$\alpha := \frac{1}{\mu(1 - (1 - p_{\min})^B)\gamma\,(1 - 2\gamma L)\,m} + \frac{2L\gamma}{1 - 2\gamma L}. \tag{7.40}$$

Proof. We reuse the same proof technique as previously and introduce the same operators Γ and $\Gamma_{n,b}$, again dropping all s indices for simplicity.

We introduce $C := \mathrm{Diag}\left(\frac{1-(1-p)^B}{p}\right)$ and using Lemma 1 we have

$$\begin{aligned}\mathbb{E}\left[\|g_n\|_{C^{-1}}^2|\mathcal{F}_{n-1}\right] &\le \mathbb{E}\left[\|f'(w_{n-1}) - f'(y) + \Gamma F'(y)\|^2|\mathcal{F}_{n-1}\right]\\ &\le 4L(F(w_{n-1}) - F_* + F(y) - F_*),\end{aligned}$$

using similar arguments as for regular mini-batch. Therefore, we have

$$\begin{aligned}\mathbb{E}\left[\|w_n - w_*\|_{C^{-1}}^2|\mathcal{F}_{n-1}\right] &\le \|w_{n-1} - w_*\|_{C^{-1}}^2 - 2\gamma F'(w_{n-1})^T(w_{n-1} - w_*) + 4\gamma^2 L(F(w_{n-1}) - F_* + F(y) - F_*)\\ &\le \|w_{n-1} - w_*\|_{C^{-1}}^2 - 2\gamma\,(1 - 2\gamma L)\,(F(w_{n-1}) - F_*) + 4\gamma^2 L\,(F(y) - F_*)\,.\end{aligned}$$

Summing the above inequality for $n \in [m]$ and taking the expectation with respect to $\mathcal{F}_0$,

$$\mathbb{E}\left[\|w_m - w_*\|_{C^{-1}}^2\right] \le \|y - w_*\|_{C^{-1}} - 2\gamma\,(1 - 2\gamma L)\,\mathbb{E}\left[\sum_{n\in[m]} F(w_n) - F_*|\mathcal{F}_0\right] + 4L\gamma^2 m(F(y) - F_*).$$

Using the strong convexity of F we have

$$\begin{aligned}\|w_0 - w_*\|_{C^{-1}} &\leq \frac{1}{1-(1-p_{\min})^B} \|w_0 - w_*\|^2_{\mathrm{diag}(p)} \\ &\leq \frac{2}{\mu(1-(1-p_{\min})^B)} \left(F(y) - F_*\right),\end{aligned}$$

and finally

$$\mathbb{E}\left[F\left(\frac{1}{m}\sum_{n\in[m]} w_n\right) - F_* | \mathcal{F}_0\right] \leq \left(\frac{1}{\mu(1-(1-p_{\min})^B)\gamma\,(1-2\gamma L)\,m} + \frac{2L\gamma}{1-2\gamma L}\right)(F(y) - F(w_*)).$$

□

One can derive a simplified convergence result when we assume $p_{\min}$ small enough.

Corollary 2. *If we assume $p_{\min} \ll 1$ so that $(1-(1-p_{\min})^B) \approx Bp_{\min}$ then with $\gamma = \frac{1}{10L}$ and $m = \frac{20L}{Bp_{\min}\mu}$, we have*

$$\mathbb{E}\left[F(y_s) - F_*\right] \leq 0.9^s(F(y_0) - F_*).$$

D.3. Comparing the effect of regular mini-batch and AdaBatch for SVRG

If F verifies our sparse convexity condition

$$\forall w \in \mathbb{R}^d,\ \ \mu\mathrm{diag}\,(p) \preceq F''(w) \preceq L\mathrm{diag}\,(p)\,, \tag{7.41}$$

we can apply Theorem 3 for

$$\forall w \in \mathbb{R}^d,\ \ \mu p_{\min} \preceq F''(w) \preceq L, \tag{7.42}$$

We will assume that the batch size B is large enough (for instance $B = 50$), $p_{\min}$ is small enough so that $1-(1-p_{\min})^B \approx Bp_{\min}$. Then using corollary 1, in order to achieve a linear convergence rate of 0.9 for regular mini-batch we would need to have a number of inner iterations given by

$$m_{\text{mb}} \approx \frac{2.2L}{p_{\min}\mu}.$$

This number is roughly constant with the batch size. However the cost of each single iteration is now B times larger, thus meaning that we would need to process B times more samples before reaching the same accuracy as when $B = 1$.

On the other hand, using Corollary 2, in order to achieve the same rate of convergence, we would require the number of inner iterations to be

$$m_{\text{ab}} \approx \frac{20L}{Bp_{\min}\mu},$$

thus the number of inner iterations is inversely proportionnal to the batch size, which balances perfectly the increased cost of each iteration. We will reach the same accuracy as for $B = 1$ without requiring to process more samples.

It should be noted that using Lemma 2, it is possible to show that AdaBatch also benefits from variance reduction for the coordinates where $p^{(k)}B$ is large enough. This will depend on the datasets but we have observed such an effect in practice, which allows us to take a larger step-size and further improve convergence.

As for regular SGD, we have noticed experimentally that mini-batch SVRG will become more efficient than AdaBatch when we are close to the optimum. For instance, on datasets that are much smaller than *url* such as *news20* or *spam*, we observed that mini-batch SVRG will perform better than AdaBatch. Therefore, we would advice using AdaBatch for early optimization and regular mini-batch for fine tuning when close to the optimum.

E. Experimental results

E.1. Experimental results for AdaBatch Wild

We present here the same graphs as in the main paper but for the *spam* and *url* dataset. We also provide the convergence with respect to the number of samples for *news20*. On both datasets, AdaBatch performs competitively with Hogwild! and significantly better than mini-batch SGD, especially when increasing the number of workers and batch-size.

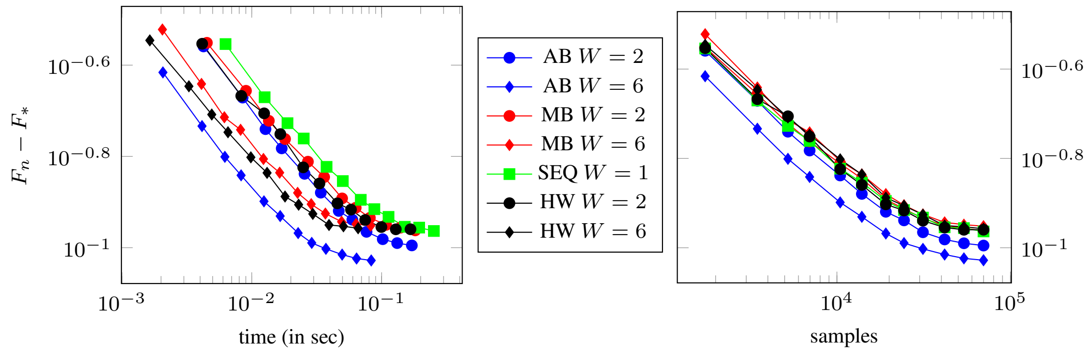


Figure 5: Convergence result for *news20*. The error is given either as a function of the wall-clock time (left) or of the number of samples processed (right).

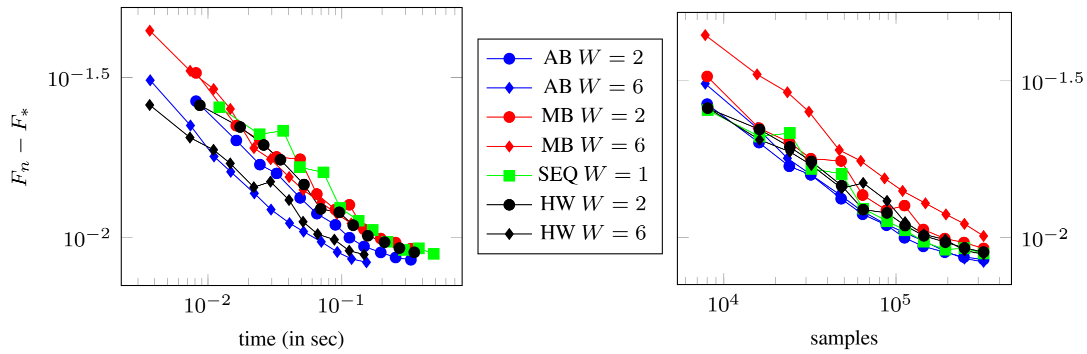


Figure 6: Convergence result for *spam*. The error is given either as a function of the wall-clock time (left) or of the number of samples processed (right).

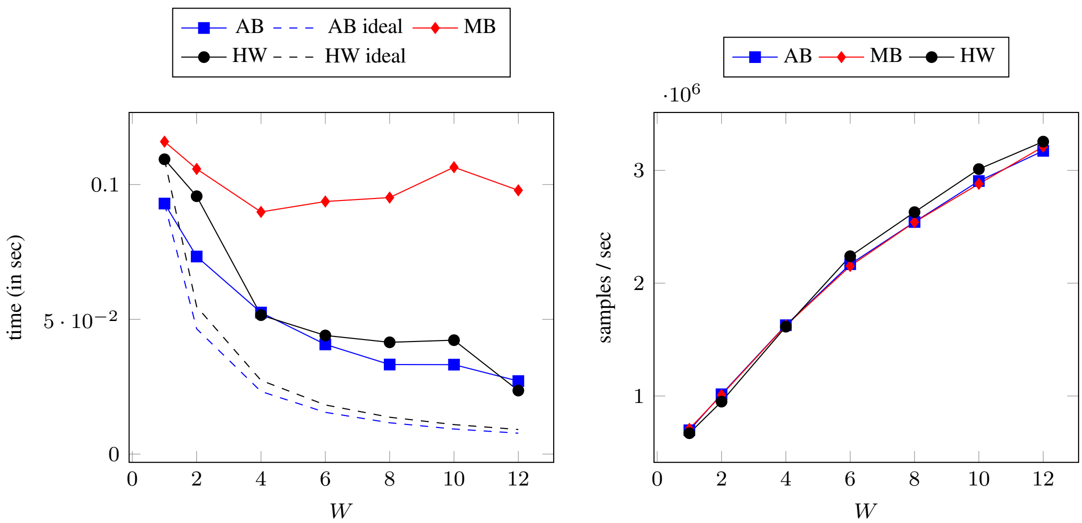


Figure 7: On the left, time to achieve a given test error when varying the number of workers for the *spam* dataset. The dashed line represents an ideal speedup dividing the time for 1 worker by W. On the right, number of process sampled per second as a function of W for the *spam* dataset.

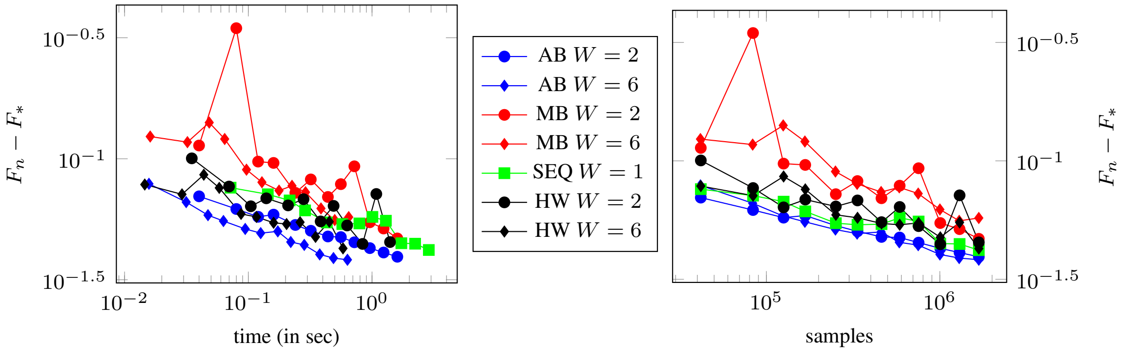


Figure 8: Convergence result for *url*. The error is given either as a function of the wall-clock time (left) or of the number of samples processed (right).

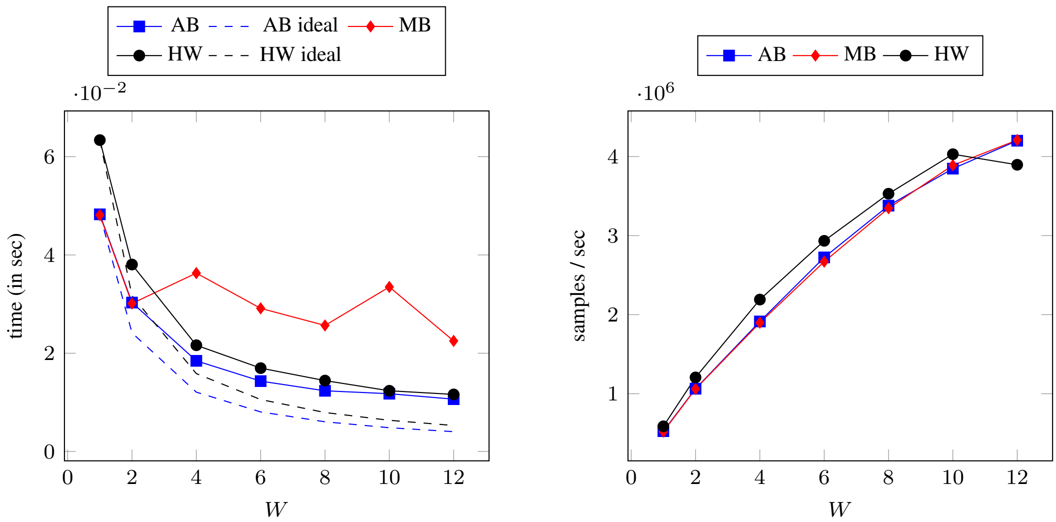
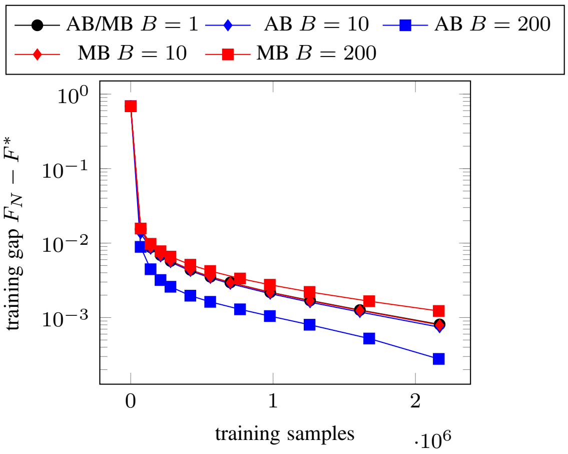


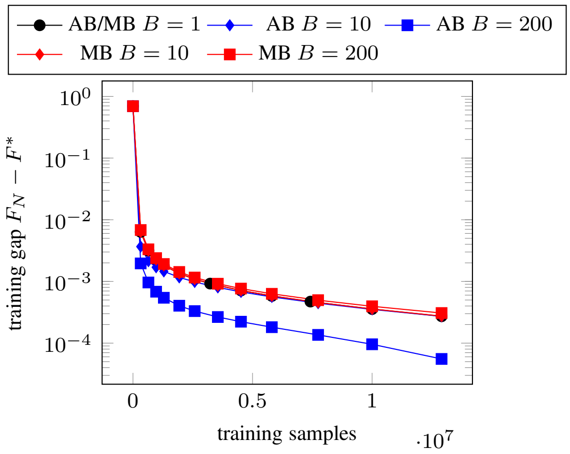
Figure 9: On the left, time to achieve a given test error when varying the number of workers on *url*. The dashed line represents an ideal speedup dividing the time for 1 worker by W. On the right, number of process sampled per second as a function of W on *url*.

E.2. Experimental results for SVRG

We give a comparison of regular mini-batch and AdaBatch on news20 and spam on figure 10. The difference is less marked than on the url dataset which we believe is due to the relative simplicity of the optimization problem on such datasets. The L2 regularization was chosen in order to achieve relatively good validation error while retaining the good convergence of SVRG in the strongly convex case.



(a) Comparison of the training gap $F_n - F_*$ for regular mini-batch vs AdaBatch with SVRG on *news20* for the log loss with an L2 penalty of $\frac{10^{-5}}{2} \|w\|^2_{\mathrm{diag}(p)}$.



(b) Comparison of the training gap $F_n - F_*$ for regular mini-batch vs AdaBatch with SVRG on *spam* for the log loss with an L2 penalty of $\frac{10^{-6}}{2} \|w\|^2_{\mathrm{diag}(p)}$.

Figure 10: Comparison of regular mini-batch vs AdaBatch with SVRG on *news20* and *spam* dataset.

References

[1] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 21*, 2008.

[2] Francis Bach and Eric Moulines. Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning. In *Advances in Neural Information Processing Systems 24*, 2011.

[3] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: a view from the trenches. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[4] Balaji Venu. Multi-core processors - an overview. Technical Report 1110.3535, arXiv, 2011.

[5] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, 2010.

[6] Martin Zinkevich, John Langford, and Alex J. Smola. Slow learners are fast. In *Advances in Neural Information Processing Systems 22*, 2009.

[7] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 24*, 2011.

[8] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. Passcode: Parallel asynchronous stochastic dual co-ordinate descent. In *International Conference on Machine Learning*, 2015.

[9] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed Iterate Analysis for Asynchronous Stochastic Optimization. Technical Report 1507.06970, arXiv, 2015.

[10] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 2012.

[11] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Parallelizing Stochastic Approximation Through Mini-Batching and Tail-Averaging. Technical Report 1610.03774, arXiv, 2016.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

[13] Nicolas Le Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In *Advances in Neural Information Processing Systems 20*, 2008.

[14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[15] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. ASAGA: Asynchronous Parallel SAGA. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[16] X. Pan, M. Lam, S. Tu, D. Papailiopoulos, C. Zhang, M. I. Jordan, K. Ramchandran, C. Re, and B. Recht. CYCLADES: Conflict-free Asynchronous Machine Learning. In *Advances in Neural Information Processing Systems 29*, 2016.

[17] John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems 26*, 2013.

[18] Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n). In *Advances in Neural Information Processing Systems 26*, 2013.

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

[20] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik, and Martin Takác. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, 2015.

[21] D. Needell and R. Ward. Batched Stochastic Gradient Descent with Weighted Sampling. Technical Report 1608.07641, arXiv, 2016.

[22] Yu Nesterov and J-Ph Vial. Confidence level solutions for stochastic programming. *Automatica*, 2008.

[23] Alexandre Défossez and Francis Bach. Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *Artificial Intelligence and Statistics*, 2015.

[24] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, 2013.

[25] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, 2014.

[26] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *International Conference on Machine Learning*, 2009.

[27] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 2015.