Big Batch SGD: Automated Inference using Adaptive Batch Sizes

Soham De, Abhay Yadav, David Jacobs and Tom Goldstein

Department of Computer Science, University of Maryland, College Park
{sohamde, jaiabhay, djacobs, tomg}@cs.umd.edu

April 10, 2017

Abstract

Classical stochastic gradient methods for optimization rely on noisy gradient approximations that become progressively less accurate as iterates approach a solution. The large noise and small signal in the resulting gradients makes it difficult to use them for adaptive stepsize selection and automatic stopping. We propose alternative "big batch" SGD schemes that adaptively grow the batch size over time to maintain a nearly constant signal-to-noise ratio in the gradient approximation. The resulting methods have similar convergence rates to classical SGD, and do not require convexity of the objective. The high fidelity gradients enable automated learning rate selection and do not require stepsize decay. Big batch methods are thus easily automated and can run with little or no oversight.

1 Introduction

We are interested in problems of the form

$$\min_{x \in \mathcal{X}} \ell(x) := \begin{cases} \mathbb{E}_{z\sim p}[f(x;z)], \\ \frac{1}{N}\sum_{i=1}^{N} f(x;z_i), \end{cases} \tag{1}$$

where $\{z_i\}$ is a collection of data drawn from a probability distribution p. We assume that ℓ and f are differentiable, but possibly non-convex, and domain $\mathcal{X}$ is convex. In typical applications, each term $f(x;z)$ measures how well a model with parameters x fits one particular data observation z. The expectation over z measures how well the model fits the entire corpus of data on average.

When N is large (or even infinite), it becomes intractable to exactly evaluate $\ell(x)$ or its gradient $\nabla\ell(x)$, which makes classical gradient methods impossible. In such situations, the method of choice for minimizing (1) is the stochastic gradient descent (SGD) algorithm (Robbins and Monro, 1951). On iteration t, SGD selects a batch $\mathcal{B} \subset \{z_i\}$ of data uniformly at random, and then computes

$$x_{t+1} = x_t - \alpha_t \nabla_x \ell_{\mathcal{B}}(x_t), \tag{2}$$
$$\text{where} \quad \ell_{\mathcal{B}}(x) = \frac{1}{|\mathcal{B}|}\sum_{z\in\mathcal{B}} f(x;z),$$

and α_t denotes the stepsize used on the t-th iteration. Note that $\mathbb{E}_{\mathcal{B}}[\nabla_x \ell_{\mathcal{B}}(x_t)] = \nabla_x \ell(x_t)$, and so the calculated gradient $\nabla_x \ell_{\mathcal{B}}(x_t)$ can be interpreted as a "noisy" approximation to the true gradient.

Because the gradient approximations are noisy, the stepsize α_t must vanish as $t \to \infty$ to guarantee convergence of the method. Typical stepsize rules require the user to find the optimal decay rate schedule, which usually requires an expensive grid search over different possible parameter values.

A preliminary version of this paper appears in AISTATS 2017 (International Conference on Artificial Intelligence and Statistics), Fort Lauderdale, USA De et al. (2017). This is the extended version.

In this paper, we consider a "big batch" strategy for SGD. Rather than letting the stepsize vanish over time as the iterates approach a minimizer, we let the minibatch $\mathcal{B}$ *adaptively grow* in size to maintain a constant signal-to-noise ratio of the gradient approximation. This prevents the algorithm from getting overwhelmed with noise, and guarantees convergence with an appropriate constant stepsize. Recent results (Keskar et al., 2016) have shown that large *fixed* batch sizes fail to find good minimizers for non-convex problems like deep neural networks. Adaptively increasing the batch size over time overcomes this limitation: intuitively, in the initial iterations, the increased stochasticity (corresponding to smaller batches) can help land the iterates near a good minimizer, and larger batches later on can increase the speed of convergence towards this minimizer.

Using this batching strategy, we show that we can keep the stepsize constant, or let it adapt using a simple Armijo backtracking line search, making the method completely adaptive with no user-defined parameters. We also derive an adaptive stepsize method based on the Barzilai and Borwein (1988) curvature estimate that fully automates the big batch method, while empirically enjoying a faster convergence rate than the Armijo backtracking line search.

Big batch methods that adaptively grow the batch size over time have several potential advantages over conventional small-batch SGD:

- Big batch methods don't require the user to choose stepsize decay parameters. Larger batch sizes with less noise enable easy estimation of the accuracy of the approximate gradient, making it straightforward to adaptively scale up the batch size and maintain fast convergence.
- Backtracking line search tends to work very well when combined with big batches, making the methods completely adaptive with no parameters. A nearly constant signal-to-noise ratio also enables us to define an adaptive stepsize method based on the Barzilai-Borwein curvature estimate, that performs better empirically on a range of convex problems than the backtracking line search.
- Higher order methods like stochastic L-BFGS typically require more work per iteration than simple SGD. When using big batches, the overhead of more complex methods like L-BFGS can be amortized over more costly gradient approximations. Furthermore, better Hessian approximations can be computed using less noisy gradient terms.
- For a restricted class of non-convex problems (functions satisfying the Polyak-Łojasiewicz Inequality), the per-iteration complexity of big batch SGD is linear and the approximate gradients vanish as the method approaches a solution, which makes it easy to define automated stopping conditions. In contrast, small batch SGD exhibits sub-linear convergence, and the noisy gradients are not usable as a stopping criterion.
- Big batch methods are much more efficient than conventional SGD in massively parallel/distributed settings. Bigger batches perform more computation between parameter updates, and thus allow a much higher ratio of computation to communication.

For the reasons above, big batch SGD is potentially much easier to automate and requires much less user oversight than classical small batch SGD.

1.1 Related work

In this paper, we focus on automating stochastic optimization methods by reducing the noise in SGD. We do this by adaptively growing the batch size to control the variance in the gradient estimates, maintaining an approximately constant signal-to-noise ratio, leading to automated methods that do not require vanishing stepsize parameters. While there has been some work on adaptive stepsize methods for stochastic optimization (Mahsereci and Hennig, 2015; Schaul et al., 2013; Tan et al., 2016; Kingma and Ba, 2014; Zeiler, 2012), the methods are largely heuristic without any kind of theoretical guarantees or convergence rates. The work in Tan et al. (2016) was a first step towards provable automated stochastic methods, and we explore in this direction to show provable convergence rates for the automated big batch method.

While there has been relatively little work in provable automated stochastic methods, there has been recent interest in methods that control gradient noise. These methods mitigate the effects of vanishing stepsizes, though choosing the (constant) stepsize still requires tuning and oversight. There have been a few papers in this direction that use dynamically increasing batch sizes. In Friedlander and Schmidt (2012), the authors propose to increase the size of the batch by a constant factor on *every* iteration, and prove linear convergence in terms of the iterates of the algorithm. In Byrd et al. (2012), the authors propose an adaptive strategy for growing the batch size; however, the authors do not present a theoretical guarantee for this method, and instead prove linear convergence for a continuously growing batch, similar to Friedlander and Schmidt (2012).

Variance reduction (VR) SGD methods use an error correction term to reduce the noise in stochastic gradient estimates. The methods enjoy a provably faster convergence rate than SGD and have been shown to outperform SGD on convex problems (Defazio et al., 2014a; Johnson and Zhang, 2013; Schmidt et al., 2013; Defazio et al., 2014b), as well as in parallel (Reddi et al., 2015) and distributed settings (De and Goldstein, 2016). A caveat, however, is that these methods require either extra storage or full gradient computations, both limiting factors when the dataset is very large. In a recent paper (Harikandeh et al., 2015), the authors propose a growing batch strategy for a VR method that enjoys the same convergence guarantees. However, as mentioned above, choosing the constant stepsize still requires tuning. Another conceptually related approach is importance sampling, i.e., choosing training points such that the variance in the gradient estimates is reduced (Bouchard et al., 2015; Csiba and Richtárik, 2016; Needell et al., 2014).

2 Big Batch SGD

2.1 Preliminaries and motivation

Classical stochastic gradient methods thrive when the current iterate is far from optimal. In this case, a small amount of data is necessary to find a descent direction, and optimization progresses efficiently. As x_t starts approaching the true solution $x^\star$, however, noisy gradient estimates frequently fail to produce descent directions and do not reliably decrease the objective. By choosing larger batches with less noise, we may be able to maintain descent directions on each iteration and uphold fast convergence. This observation motivates the proposed "big batch" method. We now explore this idea more rigorously.

To simplify notation, we hereon use $\nabla \ell$ to denote $\nabla_x \ell$. We wish to show that a noisy gradient approximation produces a descent direction when the noise is comparable in magnitude to the true gradient.

Lemma 1. *A sufficient condition for $-\nabla \ell_{\mathcal{B}}(x)$ to be a descent direction is*

$$\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2 < \|\nabla \ell_{\mathcal{B}}(x)\|^2.$$

This is a standard result in stochastic optimization (see the supplemental). In words, if the error $\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2$ is small relative to the gradient $\|\nabla \ell_{\mathcal{B}}(x)\|^2$, the stochastic approximation is a descent direction. But how big is this error and how large does a batch need to be to guarantee this condition? By the weak law of large numbers[1]

$$\begin{aligned} \mathbb{E}[\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2] &= \frac{1}{|\mathcal{B}|}\mathbb{E}[\|\nabla f(x;z) - \nabla \ell(x)\|^2] \\ &= \frac{1}{|\mathcal{B}|}\operatorname{Tr}\operatorname{Var}_z \nabla f(x;z), \end{aligned}$$

and so we can estimate the error of a stochastic gradient if we have some knowledge of the variance of $\nabla f(x;z)$. In practice, this variance could be estimated using the sample variance of a batch $\{\nabla f(x;z)\}_{z\in\mathcal{B}}$.

[1] We assume the random variable $\nabla f(x;z)$ is measurable and has bounded second moment. These conditions will be guaranteed by the hypothesis of Theorem 1.

However, we would like some bounds on the magnitude of this gradient to show that it is well-behaved, and also to analyze worst-case convergence behavior. To this end, we make the following assumption.

Assumption 1. *We assume f has L_z-Lipschitz dependence on data z, i.e., given two data points $z_1, z_2 \sim p(z)$, we have:*

$$\|\nabla f(x; z_1) - \nabla f(x; z_2)\| \leq L_z \|z_1 - z_2\|.$$

Under this assumption, we can bound the error of the stochastic gradient. The bound is uniform with respect to x, which makes it rather useful in analyzing the convergence rate for big batch methods.

Theorem 1. *Given the current iterate x, suppose Assumption 1 holds and that the data distribution p has bounded second moment. Then the estimated gradient $\nabla \ell_{\mathcal{B}}(x)$ has variance bounded by*

$$\begin{aligned} \mathbb{E}_{\mathcal{B}} \|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2 &:= \operatorname{Tr} \operatorname{Var}_{\mathcal{B}}(\nabla \ell_{\mathcal{B}}(x)) \\ &\leq \frac{4L_z^2 \operatorname{Tr} \operatorname{Var}_z(z)}{|\mathcal{B}|}, \end{aligned}$$

where $z \sim p(z)$. Note the bound is uniform in x.

The proof is in the supplemental. Note that, using a finite number of samples, one can easily approximate the quantity $\operatorname{Var}_z(z)$ that appears in our bound.

2.2 A template for big batch SGD

Theorem 1 and Lemma 1 together suggest that we should expect $d = -\nabla \ell_{\mathcal{B}}$ to be a descent direction reasonably often provided

$$\begin{aligned} & \theta^2 \|\nabla \ell_{\mathcal{B}}(x)\|^2 \geq \frac{1}{|\mathcal{B}|} [\operatorname{Tr} \operatorname{Var}_z(\nabla f(x; z_i))], \qquad (3) \\ \text{or} \quad & \theta^2 \|\nabla \ell_{\mathcal{B}}(x)\|^2 \geq \frac{4L_z^2 \operatorname{Tr} \operatorname{Var}_z(z)}{|\mathcal{B}|}, \end{aligned}$$

for some $\theta < 1$. Big batch methods capitalize on this observation.

On each iteration t, starting from a point x_t, the big batch method performs the following steps:

1. Estimate the variance $\operatorname{Tr} \operatorname{Var}_z[\nabla f(x_t; z)]$, and a batch size K large enough that

$$\begin{aligned} \theta^2 \mathbb{E} \|\nabla \ell_{\mathcal{B}_t}(x_t)\|^2 &\geq \mathbb{E} \|\nabla \ell_{\mathcal{B}_t}(x_t) - \nabla \ell(x_t)\|^2 \\ &= \frac{1}{K} \operatorname{Tr} \operatorname{Var}_z f(x_t; z), \qquad (4) \end{aligned}$$

   where $\theta \in (0, 1)$ and $\mathcal{B}_t$ denotes the selected batch on the t-th iteration with $|\mathcal{B}| = K$.

2. Choose a stepsize α_t.

3. Perform the update $x_{t+1} = x_t - \alpha_t \nabla \ell_{\mathcal{B}_t}(x_t)$.

Clearly, we have a lot of latitude in how to implement these steps using different variance estimators and different stepsize strategies. In the following section, we show that, if condition (4) holds, then linear convergence can be achieved using an appropriate constant stepsize. In subsequent sections, we address the issue of how to build practical big batch implementations using automated variance and stepsize estimators that require no user oversight.

3 Convergence Analysis

We now present convergence bounds for big batch SGD methods (5). We study stochastic gradient updates of the form

$$x_{t+1} = x_t - \alpha \nabla \ell_{\mathcal{B}_t}(x_t) = x_t - \alpha(\nabla \ell(x_t) + e_t), \tag{5}$$

where $e_t = \nabla \ell_{\mathcal{B}_t}(x_t) - \nabla \ell(x_t)$, and $\mathbb{E}_{\mathcal{B}}[e_t] = 0$. Let us also define $g_t = \nabla \ell(x_t) + e_t$.

Before we present our results, we first state two assumptions about the loss function $\ell(x)$.

Assumption 2. *We assume that the objective function ℓ has L-Lipschitz gradients:*

$$\ell(x) \leq \ell(y) + \nabla \ell(y)^T (x - y) + \frac{L}{2}\|x - y\|^2.$$

This is a standard smoothness assumption used widely in the optimization literature. Note that a consequence of Assumption 2 is the property:

$$\|\nabla \ell(x) - \nabla \ell(y)\| \leq L\|x - y\|.$$

Assumption 3. *We also assume that the objective function ℓ satisfies the Polyak-Lojasiewicz Inequality:*

$$\|\nabla \ell(x)\|^2 \geq 2\mu(\ell(x) - \ell(x^\star)).$$

Note that this inequality does *not* require ℓ to be convex, and is, in fact, a weaker assumption than what is usually used. It does, however, imply that every stationary point is a global minimizer (Karimi et al., 2016; Polyak, 1963).

We now present a result that establishes an upper bound on the objective value in terms of the error in the gradient of the sampled batch. We present all the proofs in the Supplementary Material.

Lemma 2. *Suppose we apply an update of the form (5) where the batch $\mathcal{B}_t$ is uniformly sampled from the distribution p on each iteration t. If the objective ℓ satisfies Assumption 2, then we have*

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \mathbb{E}\big[\ell(x_t) - \ell(x^\star) - \big(\alpha - \frac{L\alpha^2}{2}\big)\|\nabla \ell(x_t)\|^2 + \frac{L\alpha^2}{2}\|e_t\|^2\big].$$

Further, if the objective ℓ satisfies the PL Inequality (Assumption 3), we have:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2}{2}\big)\Big)\mathbb{E}[\ell(x_t) - \ell(x^\star)] + \frac{L\alpha^2}{2}\mathbb{E}\|e_t\|^2.$$

Using Lemma 2, we now provide convergence rates for big batch SGD.

Theorem 2. *Suppose ℓ satisfies Assumptions 2 and 3. Suppose further that on each iteration the batch size is large enough to satisfy (4) for $\theta \in (0, 1)$. If $0 \leq \alpha < \frac{2}{L\beta}$, where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$, then we get the following linear convergence bound for big batch SGD using updates of the form 5:*

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \gamma \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)],$$

where $\gamma = \big(1 - 2\mu(\alpha - \frac{L\alpha^2\beta}{2})\big)$. Choosing the optimal stepsize of $\alpha = \frac{1}{\beta L}$, we get

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \big(1 - \frac{\mu}{\beta L}\big) \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)].$$

Note that the above linear convergence rate bound holds without requiring convexity. Comparing it with the convergence rate of deterministic gradient descent under similar assumptions, we see that big batch SGD suffers a slowdown by a factor β, due to the noise in the estimation of the gradients. We now present a result proving a $\mathcal{O}(1/t)$ convergence rate for general smooth convex functions.

Theorem 3. *Suppose ℓ satisfies Assumptions 2, is convex, and condition (4) is satisfied on each iteration. Then we get sub-linear convergence of the form:*

$$\mathbb{E}[\ell(x_t) - \ell(x^\star)] \leq \frac{\|x_0 - x^\star\|^2}{(2\alpha - 2L\alpha^2\beta)(t+1)} = \mathcal{O}(1/t),$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2}$ and $\alpha < \frac{1}{L\beta}$. Choosing the optimal step size of $\alpha = \frac{1}{2L\beta}$, we get

$$\mathbb{E}[\ell(x_t) - \ell(x^\star)] \leq \frac{2L\beta\|x_0 - x^\star\|^2}{t+1} = \mathcal{O}(1/t).$$

3.1 Comparison to classical SGD

Conventional small batch SGD methods can attain only $O(1/t)$ convergence for strongly convex problems, thus requiring $O(1/\epsilon)$ gradient evaluations to achieve an optimality gap less than ϵ, and this has been shown to be *optimal* in the online setting (i.e., the infinite data setting) (Rakhlin et al., 2011). In the previous section, however, we have shown that big batch SGD methods converge linearly in the number of iterations, under a weaker assumption than strong convexity, in the online setting. Unfortunately, per-iteration convergence rates are not a fair comparison between these methods because the cost of a big batch iteration grows with the iteration count, unlike classical SGD. For this reason, it is interesting to study the convergence rate of big batch SGD as a function of *gradient evaluations*.

From Lemma 2, we see that we should not expect to achieve an optimality gap less than ϵ until we have: $\frac{L\alpha^2}{2}\mathbb{E}_{\mathcal{B}_t}\|e_t\|^2 < \epsilon$. In the worst case, by Theorem 1, this requires $\frac{L\alpha^2}{2}\frac{4L_z^2 \operatorname{Tr}\operatorname{Var}_z(z)}{|\mathcal{B}|} < \epsilon$, or $|\mathcal{B}| \geq O(1/\epsilon)$ gradient evaluations. Note that in the online or infinite data case, this is an optimal bound, and matches that of other SGD methods.

We choose to study the infinite sample case since the finite sample case is fairly trivial with a growing batch size: asymptotically, the batch size becomes the whole dataset, at which point we get the same asymptotic behavior as deterministic gradient descent, achieving linear convergence rates.

4 Practical Implementation with Backtracking Line Search

While one could implement a big batch method using analytical bounds on the gradient and its variance (such as that provided by Theorem 1), the purpose of big batch methods is to enable automated adaptive estimation of algorithm parameters. Furthermore, the stepsize bounds provided by our convergence analysis, like the stepsize bounds for classical SGD, are fairly conservative and more aggressive stepsize choices are likely to be more effective.

The framework outlined in Section 2.2 requires two ingredients: estimating the batch size and estimating the stepsize. Estimating the batch size needed to achieve (4) is fairly straightforward. We start with an initial batch size K, and draw a random batch $\mathcal{B}$ with $|\mathcal{B}| = K$. We then compute the stochastic gradient estimate $\nabla\ell_{\mathcal{B}}(x_t)$ and the sample variance

$$\begin{aligned} V_{\mathcal{B}} &:= \frac{1}{|\mathcal{B}|-1}\sum_{z\in\mathcal{B}}\|\nabla f(x_t; z) - \nabla\ell_{\mathcal{B}}(x_t)\|^2 \\ &\approx \operatorname{Tr}\operatorname{Var}_{z\in\mathcal{B}}(\nabla f(x_t; z)). \end{aligned} \tag{6}$$

We then test whether $\|\nabla\ell_{\mathcal{B}}(x_t)\|^2 > V_{\mathcal{B}}/|\mathcal{B}|$ as a proxy for (4). If this condition holds, we proceed with a gradient step, else we increase the batch size $K \leftarrow K + \delta_K$, and check our condition again. We fix $\delta_K = 0.1K$ for all our experiments. Our aggressive implementation also simply chooses $\theta = 1$. The fixed stepsize big batch method is listed in Algorithm 1.

Algorithm 1 Big batch SGD: fixed stepsize

```
**initialize** starting pt. $x_0$, stepsize $\alpha$, initial batch size $K > 1$, batch size increment $\delta_k$
**while** not converged **do**
  Draw random batch with size $|\mathcal{B}| = K$
  Calculate $V_\mathcal{B}$ and $\nabla \ell_\mathcal{B}(x_t)$ using (6)
  **while** $\|\nabla \ell_\mathcal{B}(x_t)\|^2 \leq V_\mathcal{B}/K$ **do**
    Increase batch size $K \leftarrow K + \delta_K$
    Sample more gradients
    Update $V_\mathcal{B}$ and $\nabla \ell_\mathcal{B}(x_t)$
  **end while**
  $x_{t+1} = x_t - \alpha \nabla \ell_\mathcal{B}(x_t)$
**end while**
```

We also consider a backtracking variant of SGD that adaptively tunes the stepsize. This method selects batch sizes using the same criterion (6) as in the constant stepsize case. However, after a batch has been selected, a backtracking Armijo line search is used to select a stepsize. In the Armijo line search, we keep decreasing the stepsize by a constant factor (in our case, by a factor of 2) until the following condition is satisfied on each iteration:

$$\ell_\mathcal{B}(x_{t+1}) \leq \ell_\mathcal{B}(x_t) - c\alpha_t \|\nabla \ell_\mathcal{B}(x_t)\|^2, \tag{7}$$

where c is a parameter of the line search usually set to $0 < c \leq 0.5$. We now present a convergence result of big batch SGD using the Armijo line search.

Theorem 4. *Suppose that ℓ satisfies Assumptions 2 and 3 and on each iteration, and the batch size is large enough to satisfy* (4) *for $\theta \in (0, 1)$. If an Armijo line search, given by* (7)*, is used, and the stepsize is decreased by a factor of 2 failing* (7)*, then we get the following linear convergence bound for big batch SGD using updates of the form 5:*

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \gamma \cdot \mathbb{E}[\ell(x_t) - \ell(x^\star)],$$

where $\gamma = \Big(1 - 2c\mu \min\big(\alpha_0, \frac{1}{2\beta L}\big)\Big)$ and $0 < c \leq 0.5$. If the initial stepsize α_0 is set large enough such that $\alpha_0 \geq \frac{1}{2\beta L}$, then we get:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \Big(1 - \frac{c\mu}{\beta L}\Big) \mathbb{E}[\ell(x_t) - \ell(x^\star)].$$

In practice, on iterations where the batch size increases, we double the stepsize before running line search to prevent the stepsizes from decreasing monotonically. The complete details are listed in Algorithm 2.

5 Adaptive Step Sizes using the Barzilai-Borwein Estimate

While the Armijo backtracking line search leads to an automated big batch method, the stepsize sequence is monotonic (neglecting the heuristic mentioned in the previous section). In this section, we derive a non-monotonic stepsize scheme that uses curvature estimates to propose new stepsize choices.

Our derivation follows the classical adaptive Barzilai and Borwein (1988) (BB) method. The BB methods fits a quadratic model to the objective on each iteration, and a stepsize is proposed that is optimal for the local quadratic model (Goldstein et al., 2014). To derive the analog of the BB method for stochastic problems, we consider quadratic approximations of the form $\ell(x) = \mathbb{E}_\phi f(x; \phi)$, where $f(x; \phi) = \frac{\nu}{2}\|x - \phi\|^2$ and $\phi \sim \mathcal{N}(x^\star, \sigma^2 I)$. We now derive the optimal stepsize on each iteration for this quadratic approximation (for complete details of this derivation see the supplemental). We can write:

$$\ell(x) = \mathbb{E}_\phi f(x; \phi) = \frac{\nu}{2}\big(\|x - x^\star\|^2 + d\sigma^2\big).$$

Algorithm 2 Big batch SGD: backtracking line search

```
initialize starting pt. $x_0$, initial stepsize $\alpha$, initial batch size $K > 1$, batch size increment $\delta_k$, backtracking line search parameter $c$, flag $F = 0$
while not converged do
  Draw random batch with size $|\mathcal{B}| = K$
  Calculate $V_{\mathcal{B}}$ and $\nabla \ell_{\mathcal{B}}(x_t)$ using (6)
  while $\|\nabla \ell_{\mathcal{B}}(x_t)\|^2 \leq V_{\mathcal{B}}/K$ do
    Increase batch size $K \leftarrow K + \delta_K$
    Sample more gradients
    Update $V_{\mathcal{B}}$ and $\nabla \ell_{\mathcal{B}}(x_t)$
    Set flag $F = 1$
  end while
  if flag $F == 1$ then
    $\alpha \leftarrow \alpha * 2$
    Reset flag $F = 0$
  end if
  while $\ell_{\mathcal{B}}(x_t - \alpha \nabla \ell_{\mathcal{B}}(x_t)) > \ell_{\mathcal{B}}(x_t) - c\alpha_t \|\nabla \ell_{\mathcal{B}}(x_t)\|^2$ do
    $\alpha \leftarrow \alpha/2$
  end while
  $x_{t+1} = x_t - \alpha \nabla \ell_{\mathcal{B}}(x_t)$
end while
```

Further, notice that:

$$\begin{aligned} \mathbb{E}_{\phi}[\nabla \ell(x)] &= \nu(x - x^\star), \quad \text{and} \\ \operatorname{Tr}\operatorname{Var}_{\phi}[\nabla \ell(x)] &= d\nu^2\sigma^2. \end{aligned}$$

Now, we can rewrite the big batch SGD update as:

$$\begin{aligned} x_{t+1} &= x_t - \alpha_t \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nu(x_t - \phi_i) \\ &= (1 - \nu\alpha_t)x_t + \nu\alpha_t x^\star + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \xi_i, \end{aligned}$$

where we write $\phi_i = x^\star + \sigma\xi_i$ with $\xi_i \sim \mathcal{N}(0, 1)$. Thus, the expected value of the function is:

$$\mathbb{E}[\ell(x_{t+1})] = \frac{\nu}{2}\big(\|(1 - \nu\alpha_t)(x_t - x^\star)\|^2 + (1 + \frac{\nu^2\alpha_t^2}{|\mathcal{B}|})d\sigma^2\big).$$

Minimizing $\mathbb{E}[\ell(x_{t+1})]$ w.r.t. α_t we get:

$$\begin{aligned} \alpha_t &= \frac{1}{\nu} \cdot \frac{\mathbb{E}\big\|\nabla \ell_{\mathcal{B}_t}(x_t)\big\|^2 - \frac{1}{|\mathcal{B}_t|} \operatorname{Tr}\operatorname{Var}[\nabla f(x_t)]}{\mathbb{E}\big\|\nabla \ell_{\mathcal{B}_t}(x_t)\big\|^2} \\ &= \frac{1}{\nu} \cdot \left(1 - \frac{\frac{1}{|\mathcal{B}_t|} \operatorname{Tr}\operatorname{Var}[\nabla f(x_t)]}{\mathbb{E}\big\|\nabla \ell_{\mathcal{B}_t}(x_t)\big\|^2}\right). \end{aligned} \tag{8}$$

Here ν denotes the curvature of the quadratic approximation. Note that, in the case of *deterministic* gradient descent, the optimal stepsize is simply $1/\nu$ (Goldstein et al., 2014).

We estimate the curvature ν_t on each iteration using the BB least-squares rule (Barzilai and Borwein, 1988):

$$\nu_t = \frac{\langle x_t - x_{t-1}, \nabla \ell_{\mathcal{B}_t}(x_t) - \nabla \ell_{\mathcal{B}_t}(x_{t-1})\rangle}{\|x_t - x_{t-1}\|^2}. \tag{9}$$

Thus, each time we sample a batch $\mathcal{B}_t$ on the t-th iteration, we calculate the gradient on that batch in the previous iterate, i.e., we calculate $\nabla \ell_{\mathcal{B}_t}(x_{t-1})$. This gives us an approximate curvature estimate, with which we derive the stepsize α_t using (8).

5.1 Convergence Proof

Here we prove convergence for the adaptive stepsize method described above. For the convergence proof, we first state two assumptions:

Assumption 4. *Each f has L-Lipschitz gradients:*

$$f(x) \leq f(y) + \nabla f(y)^T(x-y) + \frac{L}{2}\|x-y\|^2.$$

Assumption 5. *Each f is μ-strongly convex:*

$$\langle \nabla f(x) - \nabla f(y), x-y \rangle \geq \mu \|x-y\|^2.$$

Note that both assumptions are stronger than Assumptions 2 and 3, i.e., Assumption 4 implies 2 and Assumption 5 implies 3 (Karimi et al., 2016). Both are very standard assumptions frequently used in the convex optimization literature.

Also note that from (8), we can lower bound the stepsize as:

$$\alpha_t \geq (1-\theta^2)/\nu.$$

Thus, the stepsize for big batch SGD is scaled down by *at most* $1-\theta^2$. For simplicity, we assume that the stepsize is set to this lower bound: $\alpha_t = (1-\theta^2)/\nu_t$. Thus, from Assumptions 4 and 5, we can bound ν_t, and also α_t, as follows:

$$\mu \leq \nu_t \leq L \quad \implies \quad \frac{1-\theta^2}{L} \leq \alpha_t \leq \frac{1-\theta^2}{\mu}.$$

From Theorem 2, we see that we have linear convergence with the adaptive stepsize method when:

$$\begin{aligned} 1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big) \leq 1 - \frac{2(1-\theta^2)}{\kappa} + \beta(1-\theta^2)^2\kappa < 1, \\ \implies \kappa^2 < \frac{2}{\beta(1-\theta^2)}, \end{aligned}$$

where $\kappa = L/\mu$ is the condition number. We see that the adaptive stepsize method enjoys a linear convergence rate when the problem is well-conditioned. In the next section, we talk about ways to deal with poorly-conditioned problems.

5.2 Practical Implementation

To achieve robustness of the algorithm for poorly conditioned problems, we include a backtracking line search after calculating (8), to ensure that the stepsizes do not blow up. Further, instead of calculating two gradients on each iteration ($\nabla \ell_{\mathcal{B}_t}(x_t)$ and $\nabla \ell_{\mathcal{B}_t}(x_{t-1})$), our implementation uses the same batch (and stepsize) on two consecutive iterations. Thus, one parameter update takes place for each gradient calculation.

We found the stepsize calculated from (8) to be noisy when the batch is small. While this did not affect long-term performance, we perform a smoothing operation to even out the stepsizes and make performance more predictable. Let $\tilde{\alpha}_t$ denote the stepsize calculated from (8). Then, the stepsize on each iteration is given by

$$\alpha_t = \Big(1 - \frac{|\mathcal{B}|}{N}\Big)\alpha_{t-1} + \frac{|\mathcal{B}|}{N}\tilde{\alpha}_t.$$

This ensures that the update is proportional to how accurate the estimate on each iteration is. This simple smoothing operation seemed to work very well in practice as shown in the experimental section. Note that when $|\mathcal{B}_t| = N$, we just use $\alpha_t = 1/\nu_t$. Since there is no noise in the algorithm in this case, we use the optimal stepsize for a deterministic algorithm. Algorithm 3 shows the complete details.

Algorithm 3 Big batch SGD: with BB stepsizes

```
initialize starting pt. $x$, initial stepsize $\alpha$, initial batch size $K > 1$, batch size increment $\delta_k$, backtracking line search parameter $c$
while not converged do
  Draw random batch with size $|\mathcal{B}| = K$
  Calculate $V_\mathcal{B}$ and $G_\mathcal{B} = \nabla \ell_\mathcal{B}(x)$ using (6)
  while $\|G_\mathcal{B}\|^2 \leq V_\mathcal{B}/K$ do
    Increase batch size $K \leftarrow K + \delta_K$
    Sample more gradients
    Update $V_\mathcal{B}$ and $G_\mathcal{B}$
  end while
  while $\ell_\mathcal{B}(x - \alpha \nabla \ell_\mathcal{B}(x)) > \ell_\mathcal{B}(x) - c\alpha \|\nabla \ell_\mathcal{B}(x)\|^2$ do
    $\alpha \leftarrow \alpha/2$
  end while
  $x \leftarrow x - \alpha \nabla \ell_\mathcal{B}(x)$
  if $K < N$ then
    Calculate $\tilde{\alpha} = (1 - V_B/(K\|G_\mathcal{B}\|^2))/\nu$ using (8) and (9)
  else
    Calculate $\tilde{\alpha} = 1/\nu$ using (9)
  end if
  Stepsize smoothing: $\alpha \leftarrow \alpha(1 - K/N) + \tilde{\alpha}K/N$
  while $\ell_\mathcal{B}(x - \alpha \nabla \ell_\mathcal{B}(x)) > \ell_\mathcal{B}(x) - c\alpha \|\nabla \ell_\mathcal{B}(x)\|^2$ do
    $\alpha \leftarrow \alpha/2$
  end while
  $x \leftarrow x - \alpha \nabla \ell_\mathcal{B}(x)$
end while
```

6 Experiments

In this section, we present our experimental results. We explore big batch methods with both convex and non-convex (neural network) experiments on large and high-dimensional datasets.

6.1 Convex Experiments

For the convex experiments, we test big batch SGD on a binary classification problem with logistic regression and a linear regression problem:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-b_i a_i^T x)),$$

$$\min_x \frac{1}{n} \sum_{i=1}^{n} (a_i^T x - b_i)^2.$$

Figure 1 presents the results of our convex experiments on three standard real world datasets: IJCNN1 (Prokhorov, 2001) and COVERTYPE (Blackard and Dean, 1999) for logistic regression, and MILLIONSONG

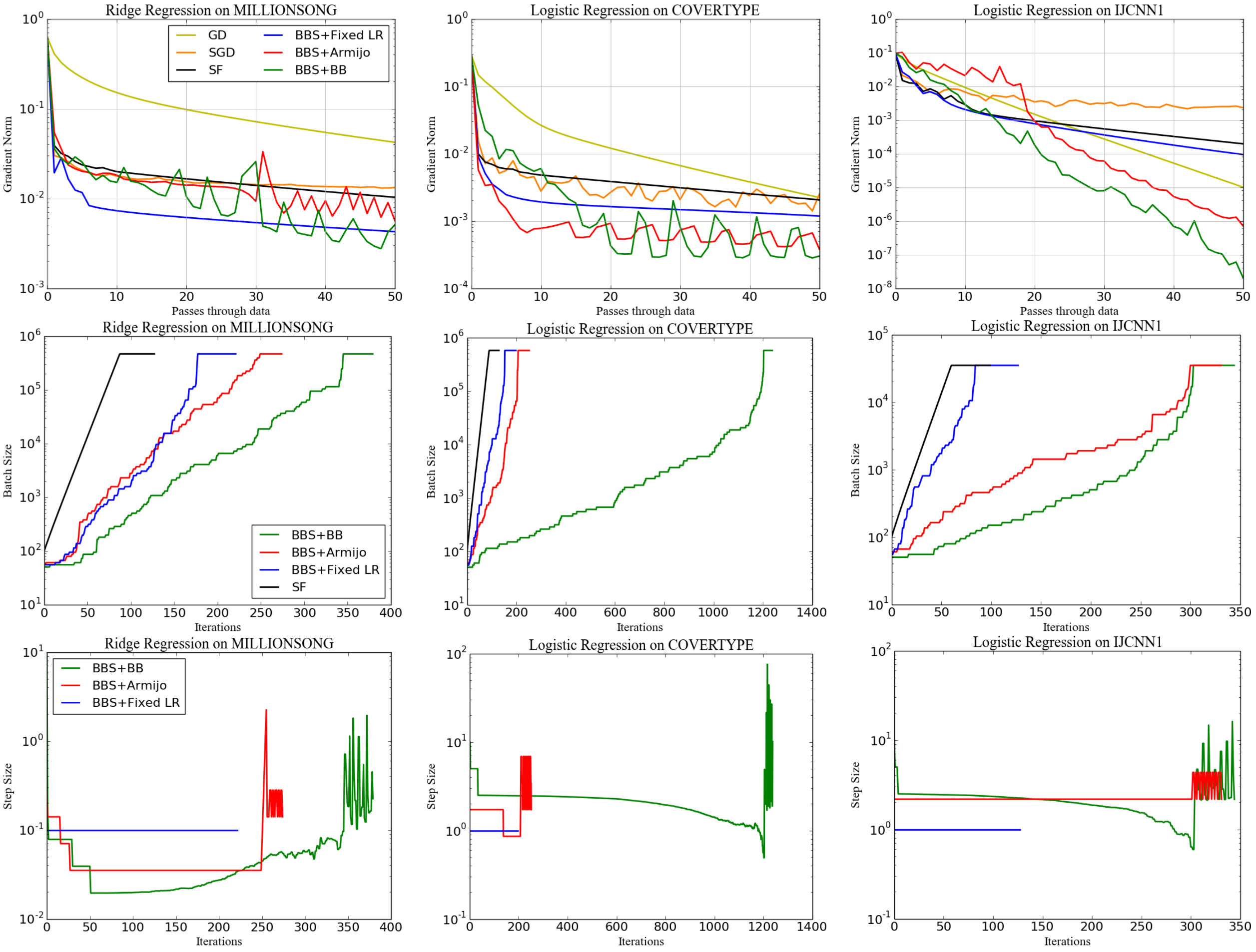

Figure 1: Convex experiments. Left to right: Ridge regression on MILLIONSONG; Logistic regression on COVERTYPE; Logistic regression on IJCNN1. The top row shows how the norm of the true gradient decreases with the number of epochs, the middle and bottom rows show the batch sizes and stepsizes used on each iteration by the big batch methods. Here 'passes through the data' indicates number of epochs, while 'iterations' refers to the number of parameter updates used by the method (there may be multiple iterations during one epoch).

(Bertin-Mahieux et al., 2011) for linear regression. As a preprocessing step, we normalize the features for each dataset. We compare deterministic gradient descent (GD) and SGD with stepsize decay ($\alpha_t = a/(b+t)$) to big batch SGD using a fixed stepsize (BBS+Fixed LR), with backtracking line search (BBS+Armijo) and with the adaptive stepsize (8) (BBS+BB), as well as the growing batch method described in Friedlander and Schmidt (2012) (denoted as SF; while the authors propose a quasi-Newton method, we adapt their algorithm to a first-order method). We selected stepsize parameters using a comprehensive grid search for all algorithms, except BBS+Armijo and BBS+BB, which require no parameter tuning.

We see that across all three problems, the big batch methods outperform the other algorithms. We also see that both fully automated methods are always comparable to or better than fixed stepsize methods. The automated methods increase the batch size more slowly than BBS+Fixed LR and SF, and thus, these methods can take more steps with smaller batches, leveraging its advantages longer. Further, note that the stepsizes derived by the automated methods are very close to the optimal fixed stepsize rate.

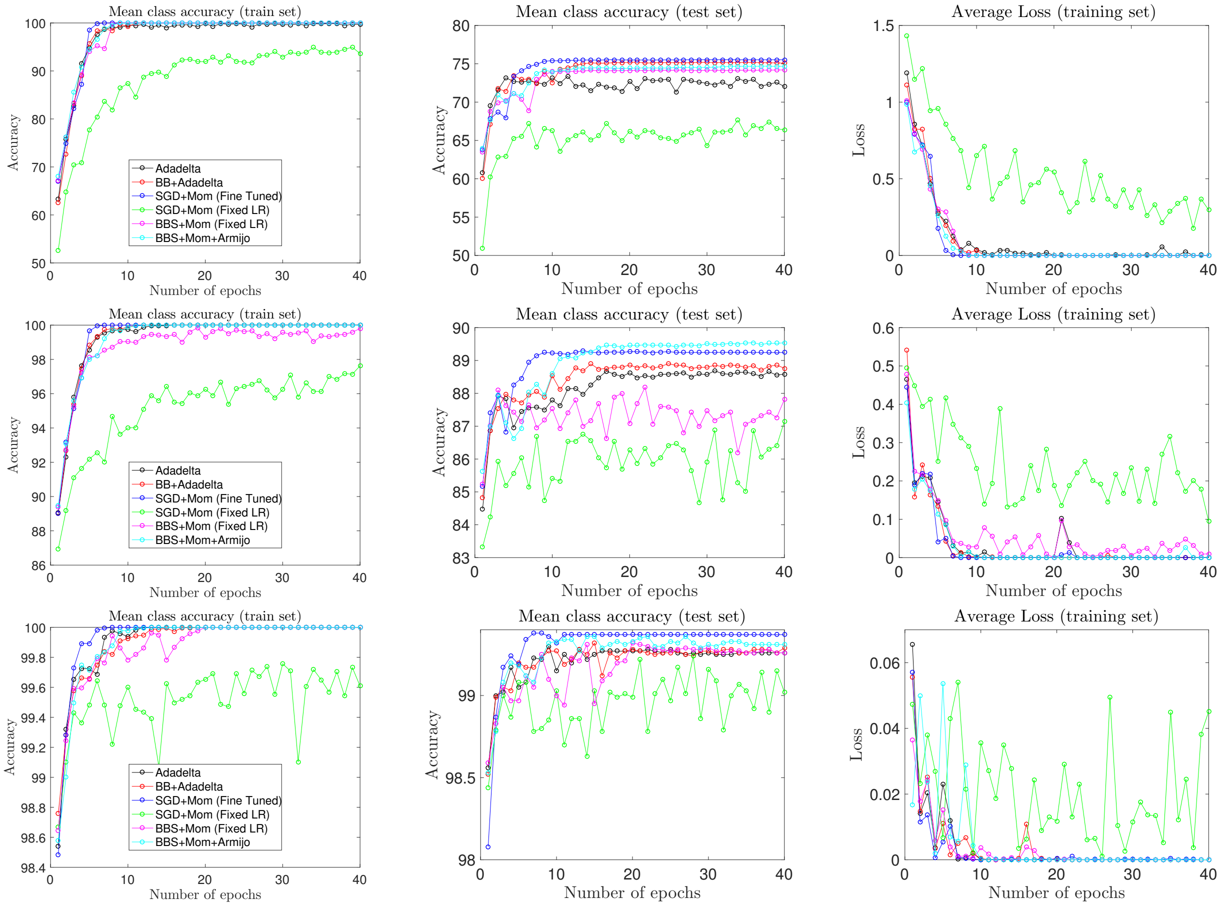

Figure 2: Neural Network Experiments. Top row presents results for CIFAR-10, middle row for SVHN, and bottom row for MNIST. The first column presents classification accuracies on the training set, the middle column presents classification accuracies on the test set, and the last column shows the change in the loss function.

6.2 Neural Network Experiments

To demonstrate the versatility of the big batch SGD framework, we also present results on neural network experiments. We compare big batch SGD against SGD with finely tuned stepsize schedules and fixed stepsizes. We also compare with Adadelta (Zeiler, 2012), and combine the big batch method with AdaDelta (BB+AdaDelta) to show that more complex SGD variants can benefit from growing batch sizes. In addition, we had also compared big batch methods with L-BFGS. However, we found L-BFGS to consistently yield poorer generalization error on neural networks, and thus we omitted these results.

We train a convolutional neural network (LeCun et al., 1998) (ConvNet) to classify three benchmark image datasets: CIFAR-10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), and MNIST (LeCun et al., 1998). Our ConvNet is composed of 4 layers. We use 32×32 pixel images as input. The first layer of the ConvNet contains $16 \times 3 \times 3$, and the second layer contains $256 \times 3 \times 3$ filters. The third and fourth layers are fully connected (LeCun et al., 1998) with 256 and 10 outputs respectively. Each layer except the last one is followed by a ReLu non-linearity (Krizhevsky et al., 2012) and a max pooling stage (Ranzato et al., 2007) of size 2×2. This ConvNet has over 4.3 million weights.

To compare against fine-tuned SGD, we used a comprehensive grid search on the stepsize schedule to identify optimal parameters (up to a factor of 2 accuracy). For CIFAR10, the stepsize starts from 0.5 and is divided by 2 every 5 epochs with 0 stepsize decay. For SVHN, the stepsize starts from 0.5 and is divided by 2 every 5 epochs with $1e{-}05$ learning rate decay. For MNIST, the learning rate starts from 1 and is divided by 2 every 3 epochs with 0 stepsize decay. All algorithms use a momentum parameter of 0.9, and SGD and AdaDelta use mini-batches of size 128.

Fixed stepsize methods use the default decay rule of the *Torch* library: $\alpha_t = \alpha_0/(1 + 10^{-7}t)$, where α_0 was chosen to be the stepsize used in the fine-tuned experiments. We also tune the hyper-parameter ρ in the Adadelta algorithm, and we found 0.9, 0.9 and 0.8 to be best-performing parameters for CIFAR10, SVHN and MNIST respectively.

We plot the accuracy on the train and test set vs the number of epochs (full passes through the dataset) in Figure 2. We notice that the big batch SGD with backtracking performs better than both Adadelta and SGD (Fixed LR) in terms of both train and test error. Big batch SGD even performs comparably to fine tuned SGD but without the trouble of fine tuning. This is interesting because most state-of-the-art deep networks (like AlexNet (Krizhevsky et al., 2012), VGG Net (Simonyan and Zisserman, 2014), ResNets (He et al., 2016)) were trained by their creators using standard SGD with momentum, and training parameters were tuned over long periods of time (sometimes months). Finally, we note that the big batch AdaDelta performs consistently better than plain AdaDelta on both large scale problems (SVHN and CIFAR-10), and performance is nearly identical on the small-scale MNIST problem.

7 Conclusion

We analyzed and studied the behavior of alternative SGD methods in which the batch size increases over time. Unlike classical SGD methods, in which stochastic gradients quickly become swamped with noise, these "big batch" methods maintain a nearly constant signal to noise ratio of the approximate gradient. As a result, big batch methods are able to adaptively adjust batch sizes without user oversight. The proposed automated methods are shown to be empirically comparable or better performing than other standard methods, but without requiring an expert user to choose learning rates and decay parameters.

Acknowledgements

This work was supported by the US Office of Naval Research (N00014-17-1-2078), and the National Science Foundation (CCF-1535902 and IIS-1526234). A. Yadav and D. Jacobs were supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *arXiv preprint arXiv:1602.02283*, 2016.

Soham De and Tom Goldstein. Efficient distributed SGD with variance reduction. In *2016 IEEE International Conference on Data Mining*. IEEE, 2016.

Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated inference with adaptive batches. In *International Conference on Artificial Intelligence and Statistics*, 2017.

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*, pages 1646–1654, 2014a.

Aaron J Defazio, Tibério S Caetano, and Justin Domke. Finito: A faster, permutable incremental gradient method for big data problems. *arXiv preprint arXiv:1407.2710*, 2014b.

Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Tom Goldstein and Simon Setzer. High-order methods for basis pursuit. *UCLA CAM Report*, pages 10–41, 2010.

Tom Goldstein, Christoph Studer, and Richard Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406, 2014. URL `http://arxiv.org/abs/1411.3406`.

Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2251–2259, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. In *Advances In Neural Information Processing Systems*, pages 181–189, 2015.

Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2014.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, Spain, 2011.

Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

Danil Prokhorov. Ijcnn 2001 neural network competition. *Slide presentation in IJCNN*, 1, 2001.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2647–2655, 2015.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *Proceedings of The 30th International Conference on Machine Learning*, pages 343–351, 2013.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

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

Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. *arXiv preprint arXiv:1605.04131*, 2016.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Supplementary Material

A Proof of Lemma 1

Proof. We know that $-\nabla \ell_{\mathcal{B}}(x)$ is a descent direction iff the following condition holds:

$$\nabla \ell_{\mathcal{B}}(x)^T \nabla \ell(x) > 0. \tag{10}$$

Expanding $\|\nabla \ell_{\mathcal{B}}(x) - \nabla \ell(x)\|^2$ we get

$$\begin{aligned} \|\nabla \ell_{\mathcal{B}}(x)\|^2 + \|\nabla \ell(x)\|^2 - 2\nabla \ell_{\mathcal{B}}(x)^T \nabla \ell(x) &< \|\nabla \ell_{\mathcal{B}}(x)\|^2, \\ \Longrightarrow -2\nabla \ell_{\mathcal{B}}(x)^T \nabla \ell(x) &< -\|\nabla \ell(x)\|_2^2 \leq 0, \end{aligned}$$

which is always true for a descent direction (10). □

B Proof of Theorem 1

Proof. Let $\bar{z} = \mathbb{E}[z]$ be the mean of z. Given the current iterate x, we assume that the batch $\mathcal{B}$ is sampled uniformly with replacement from p. We then have the following bound:

$$\begin{aligned} \|\nabla f(x; z) - \nabla \ell(x)\|^2 &\leq 2\|\nabla f(x; z) - \nabla f(x, \bar{z})\|^2 + 2\|\nabla f(x, \bar{z}) - \nabla \ell(x)\|^2 \\ &\leq 2L_z^2 \|z - \bar{z}\|^2 + 2\|\nabla f(x, \bar{z}) - \nabla \ell(x)\|^2 \\ &= 2L_z^2 \|z - \bar{z}\|^2 + 2\|\mathbb{E}_z[\nabla f(x, \bar{z}) - \nabla f(x, z)]\|^2 \\ &\leq 2L_z^2 \|z - \bar{z}\|^2 + 2\mathbb{E}_z\|\nabla f(x, \bar{z}) - \nabla f(x, z)\|^2 \\ &\leq 2L_z^2 \|z - \bar{z}\|^2 + 2L_z^2 \mathbb{E}_z\|\bar{z} - z\|^2 \\ &= 2L_z^2 \|z - \bar{z}\|^2 + 2L_z^2 \operatorname{Tr} \operatorname{Var}_z(z), \end{aligned}$$

where the first inequality uses the property $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, the second and fourth inequalities use Assumption 1, and the third line uses Jensen's inequality. This bound is *uniform* in x. We then have

$$\begin{aligned} \mathbb{E}_z\|\nabla f(x; z) - \nabla \ell(x)\|^2 &\leq 2L_z^2 \mathbb{E}_z\|z - \bar{z}\|^2 + 2L_z^2 \operatorname{Tr} \operatorname{Var}_z(z) \\ &= 4L_z^2 \operatorname{Tr} \operatorname{Var}_z(z) \end{aligned}$$

uniformly for all x. The result follows from the observation that

$$\mathbb{E}_{\mathcal{B}}\|\nabla f_{\mathcal{B}}(x) - \nabla \ell(x)\|^2 = \frac{1}{|\mathcal{B}|} E_z\|\nabla f(x; z) - \nabla \ell(x)\|^2.$$

□

C Proof of Lemma 2

Proof. From (5) and Assumption 2 we get

$$\ell(x_{t+1}) \leq \ell(x_t) - \alpha {g_t}^T \nabla \ell(x_t) + \frac{L\alpha^2}{2}\|g_t\|^2.$$

Taking expectation with respect to the batch $\mathcal{B}_t$ and conditioning on x_t, we get

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \ell(x_t) - \ell(x^\star) - \alpha \mathbb{E}[g_t]^T \nabla \ell(x_t) + \frac{L\alpha^2}{2}\mathbb{E}\|g_t\|^2$$

$$
\begin{aligned}
&= \ell(x_t) - \ell(x^\star) - \alpha\|\nabla\ell(x_t)\|^2 + \frac{L\alpha^2}{2}(\|\nabla\ell(x_t)\|^2 + \mathbb{E}\|e_t\|^2 + \mathbb{E}[e_t]^T\nabla\ell(x_t)) \\
&= \ell(x_t) - \ell(x^\star) - \big(\alpha - \frac{L\alpha^2}{2}\big)\|\nabla\ell(x_t)\|^2 + \frac{L\alpha^2}{2}\mathbb{E}\|e_t\|^2 \\
&\le \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2}{2}\big)\Big)(\ell(x_t) - \ell(x^\star)) + \frac{L\alpha^2}{2}\mathbb{E}\|e_t\|^2,
\end{aligned}
$$

where the second inequality follows from Assumption 3. Taking expectation, the result follows. □

D Proof of Theorem 2

Proof. We begin by applying the reverse triangle inequality to (4) to get

$$(1-\theta)\mathbb{E}\|\nabla\ell_{\mathcal{B}}(x)\| \le \mathbb{E}\|\nabla\ell(x)\|$$

which applied to (4) yields

$$\frac{\theta^2}{(1-\theta)^2}\mathbb{E}\|\nabla\ell(x_t)\|^2 \ge \mathbb{E}\|\nabla\ell_{\mathcal{B}}(x_t) - \nabla\ell(x_t)\|^2 = \mathbb{E}\|e_t\|^2. \tag{11}$$

Now, we apply (11) to the result in Lemma 2 to get

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \le \mathbb{E}[\ell(x_t) - \ell(x^\star)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla\ell(x_t)\|^2,$$

where $\beta = \frac{\theta^2+(1-\theta)^2}{(1-\theta)^2} \ge 1$. Assuming $\alpha - \frac{L\alpha^2\beta}{2} \ge 0$, we can apply Assumption 3 to write

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \le \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big)\Big)\mathbb{E}[\ell(x_t) - \ell(x^\star)],$$

which proves the theorem. Note that $\max_\alpha\{\alpha - \frac{L\alpha^2\beta}{2}\} = \frac{1}{2L\beta}$, and $\mu \le L$. It follows that

$$0 \le \Big(1 - 2\mu\big(\alpha - \frac{L\alpha^2\beta}{2}\big)\Big) < 1.$$

The second result follows immediately. □

E Proof of Theorem 3

Proof. Applying the reverse triangle inequality to (4) and using Lemma 2 we get, as in Theorem 2:

$$\mathbb{E}[\ell(x_{t+1})] \le \mathbb{E}[\ell(x_t)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla\ell(x_t)\|^2, \tag{12}$$

where $\beta = \frac{\theta^2+(1-\theta)^2}{(1-\theta)^2} \ge 1$. Note that $\alpha - \frac{L\alpha^2\beta}{2} > 0$ if $\alpha < \frac{2}{L\beta}$.

From (5), taking norm on both sides and taking expectation, conditioned on all x_k, with $k = 0, 1, \cdots, t$, we get

$$
\begin{aligned}
\mathbb{E}\|x_{t+1} - x^\star\|^2 &= \|x_t - x^\star\|^2 - 2\alpha\mathbb{E}\langle x_t - x^\star, \nabla\ell(x_t) + \epsilon_t\rangle + \alpha^2\mathbb{E}\|\nabla\ell(x_t) + \epsilon_t\|^2 \\
&= \|x_t - x^\star\|^2 - 2\alpha\langle x_t - x^\star, \nabla\ell(x_t)\rangle + \alpha^2\|\nabla\ell(x_t)\|^2 + \alpha^2\mathbb{E}\|\epsilon_t\|^2 \\
&\le \|x_t - x^\star\|^2 - 2\alpha\langle x_t - x^\star, \nabla\ell(x_t)\rangle + \alpha^2\|\nabla\ell(x_t)\|^2 + \alpha^2\frac{\theta^2}{(1-\theta)^2}\|\nabla\ell(x_t)\|^2
\end{aligned}
$$

$$
\begin{aligned}
&= \|x_t - x^\star\|^2 - 2\alpha\langle x_t - x^\star, \nabla\ell(x_t)\rangle + \alpha^2\beta\|\nabla\ell(x_t)\|^2 \\
&\leq \|x_t - x^\star\|^2 - 2\alpha(\ell(x_t) - \ell(x^\star)) + \alpha^2\beta\|\nabla\ell(x_t)\|^2 \\
&\leq \|x_t - x^\star\|^2 - 2\alpha(\ell(x_t) - \ell(x^\star)) + 2L\alpha^2\beta(\ell(x_t) - \ell(x^\star)) \\
&= \|x_t - x^\star\|^2 - (2\alpha - 2L\alpha^2\beta)(\ell(x_t) - \ell(x^\star)),
\end{aligned}
$$

where we use the property that $\mathbb{E}[\epsilon_t] = 0$, and the properties $\ell(x) \leq \ell(x^\star) + \langle x - x^\star, \nabla\ell(x)\rangle$ (which follows from the convexity of ℓ) and $\|\nabla\ell(x)\|^2 \leq 2L(\ell(x) - \ell(x^\star))$ (a proof for this identity can be found in Nesterov (2013)).

Note that $2\alpha - 2L\alpha^2\beta > 0$ when $\alpha < \frac{1}{L\beta}$. Taking expectation on all x, we get

$$
\mathbb{E}[\ell(x_t) - \ell(x^\star)] \leq \frac{1}{2\alpha(1 - L\alpha\beta)}(\mathbb{E}\|x_t - x^\star\|^2 - \mathbb{E}\|x_{t+1} - x^\star\|^2). \tag{13}
$$

Summing (13) over all $t = 0, 1, \cdots, T$, and using the telescoping sum in $\|x_t - x^\star\|^2$, we get:

$$
\begin{aligned}
\sum_{t=0}^{T}\mathbb{E}[\ell(x_t) - \ell(x^\star)] &\leq \frac{1}{2\alpha(1 - L\alpha\beta)}(\mathbb{E}\|x_0 - x^\star\|^2 - \mathbb{E}\|x_{T+1} - x^\star\|^2) \\
&\leq \frac{1}{2\alpha(1 - L\alpha\beta)}\|x_0 - x^\star\|^2.
\end{aligned} \tag{14}
$$

From (12) we see that $\mathbb{E}[\ell(x_{t+1})] \leq \mathbb{E}[\ell(x_t)]$ when $\alpha < \frac{2}{L\beta}$. Thus we can write (14) as

$$
\mathbb{E}[\ell(x_T) - \ell(x^\star)] \leq \frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}[\ell(x_t) - \ell(x^\star)] \leq \frac{\|x_0 - x^\star\|^2}{(2\alpha - 2L\alpha^2\beta)(T+1)}.
$$

Choosing the optimal step size of $\alpha = \frac{1}{2L\beta}$, we get

$$
\mathbb{E}[\ell(x_T) - \ell(x^\star)] \leq \frac{2L\beta\|x_0 - x^\star\|^2}{T+1}.
$$

□

F Proof of Theorem 4

Proof. Applying the reverse triangle inequality to (4) and using Lemma 2 we get, as in Theorem 2:

$$
\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \mathbb{E}[\ell(x_t) - \ell(x^\star)] - \big(\alpha - \frac{L\alpha^2\beta}{2}\big)\mathbb{E}\|\nabla\ell(x_t)\|^2, \tag{15}
$$

where $\beta = \frac{\theta^2 + (1-\theta)^2}{(1-\theta)^2} \geq 1$.

We will show that the backtracking condition in (7) is satisfied whenever $0 < \alpha_t \leq \frac{1}{\beta L}$. Notice that:

$$
0 < \alpha_t \leq \frac{1}{\beta L} \quad \implies \quad -\alpha_t + \frac{L\alpha_t^2\beta}{2} \leq -\frac{\alpha_t}{2}.
$$

Thus, we can rewrite (15) as

$$
\begin{aligned}
\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] &\leq \mathbb{E}[\ell(x_t) - \ell(x^\star)] - \frac{\alpha_t}{2}\mathbb{E}\|\nabla\ell(x_t)\|^2 \\
&\leq \mathbb{E}[\ell(x_t) - \ell(x^\star)] - c\alpha_t\mathbb{E}\|\nabla\ell(x_t)\|^2,
\end{aligned}
$$

where $0 < c \leq 0.5$. Thus, the backtracking line search condition (7) is satisfied whenever $0 < \alpha_t \leq \frac{1}{L\beta}$.

Now we know that either $\alpha_t = \alpha_0$ (the initial stepsize), or $\alpha_t \geq \frac{1}{2\beta L}$, where the stepsize is decreased by a factor of 2 each time the backtracking condition fails. Thus, we can rewrite the above as

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \mathbb{E}[\ell(x_t) - \ell(x^\star)] - c\min\Big(\alpha_0, \frac{1}{2\beta L}\Big)\mathbb{E}\|\nabla\ell(x_t)\|^2.$$

Using Assumption 3 we get

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \bigg(1 - 2c\mu\min\Big(\alpha_0, \frac{1}{2\beta L}\Big)\bigg)\mathbb{E}[\ell(x_t) - \ell(x^\star)].$$

Assuming we start off the stepsize at a large value such that $\min(\alpha_0, \frac{1}{2\beta L}) = \frac{1}{2\beta L}$, we can rewrite this as:

$$\mathbb{E}[\ell(x_{t+1}) - \ell(x^\star)] \leq \Big(1 - \frac{c\mu}{\beta L}\Big)\mathbb{E}[\ell(x_t) - \ell(x^\star)].$$

□

G Derivation of Adaptive Step Size

Here we present the complete derivation of the adaptive stepsizes presented in Section 5. Our derivation follows the classical adaptive Barzilai and Borwein (1988) (BB) method. The BB methods fits a quadratic model to the objective on each iteration, and a stepsize is proposed that is optimal for the local quadratic model (Goldstein et al., 2014). To derive the analog of the BB method for stochastic problems, we consider quadratic approximations of the form $\ell(x) = \mathbb{E}_\theta f(x, \theta)$, where we define $f(x, \theta) = \frac{\nu}{2}\|x - \theta\|^2$ with $\theta \sim \mathcal{N}(x^\star, \sigma^2 I)$.

We derive the optimal stepsize for this. We can rewrite the quadratic approximation as

$$\ell(x) = \mathbb{E}_\theta f(x, \theta) = \frac{\nu}{2}\mathbb{E}_\theta\|x - \theta\|^2 = \frac{\nu}{2}[x^T x - 2x^T x^\star - \mathbb{E}(\theta^T\theta)] = \frac{\nu}{2}\big(\|x - x^\star\|^2 + d\sigma^2\big),$$

since we can write

$$\mathbb{E}(\theta^T\theta) = \mathbb{E}\sum_{i=1}^d \theta_i^2 = \sum_{i=1}^d \mathbb{E}\theta_i^2 = \sum_{i=1}^d (x_i^\star)^2 + \sigma^2 = \|x^\star\|^2 + d\sigma^2.$$

Further, notice that:

$$\begin{aligned}\mathbb{E}_\theta[\nabla\ell(x)] &= \mathbb{E}_\theta[\nu(x - \theta)] = \nu(x - x^\star), \quad \text{and}\\ \operatorname{Tr}\operatorname{Var}_\theta[\nabla\ell(x)] &= \mathbb{E}_\theta[\nu^2(x - \theta)^T(x - \theta)] - \nu^2(x - x^\star)^T(x - x^\star) = d\nu^2\sigma^2.\end{aligned}$$

Using the quadratic approximation, we can rewrite the update for big batch SGD as follows:

$$x_{t+1} = x_t - \alpha_t\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\nu(x_t - \theta_i) = (1 - \nu\alpha_t)x_t + \frac{\nu\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\theta_i = (1 - \nu\alpha_t)x_t + \nu\alpha_t x^\star + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\xi_i,$$

where we write $\theta_i = x^\star + \sigma\xi_i$ with $\xi_i \sim \mathcal{N}(0, 1)$. Thus, the expected value of the function is:

$$\begin{aligned}\mathbb{E}[\ell(x_{t+1})] &= \mathbb{E}_\xi\left[\ell\left((1 - \nu\alpha_t)x_t + \nu\alpha_t x^\star + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\xi_i\right)\right]\\ &= \frac{\nu}{2}\mathbb{E}_\xi\left[\left\|(1 - \nu\alpha_t)(x_t - x^\star) + \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\xi_i\right\|^2 + d\sigma^2\right]\end{aligned}$$

$$
\begin{aligned}
&= \frac{\nu}{2}\left(\|(1-\nu\alpha_t)(x_t - x^\star)\|^2 + \mathbb{E}_\xi \left\| \frac{\nu\sigma\alpha_t}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \xi_i \right\|^2 + d\sigma^2\right) \\
&= \frac{\nu}{2}\left(\|(1-\nu\alpha_t)(x_t - x^\star)\|^2 + (1 + \frac{\nu^2\alpha_t^2}{|\mathcal{B}|})d\sigma^2\right).
\end{aligned}
$$

Minimizing $\mathbb{E}[\ell(x_{t+1})]$ w.r.t. α_t we get:

$$
\begin{aligned}
\alpha_t &= \frac{1}{\nu}\cdot\frac{\left\|\mathbb{E}[\nabla\ell_{\mathcal{B}_t}(x_t)]\right\|^2}{\left\|\mathbb{E}[\nabla\ell_{\mathcal{B}_t}(x_t)]\right\|^2 + \frac{1}{|\mathcal{B}_t|}\operatorname{Tr}\operatorname{Var}[\nabla f(x_t)]} \\
&= \frac{1}{\nu}\cdot\frac{\mathbb{E}\left\|\nabla\ell_{\mathcal{B}_t}(x_t)\right\|^2 - \frac{1}{|\mathcal{B}_t|}\operatorname{Tr}\operatorname{Var}[\nabla f(x_t)]}{\mathbb{E}\left\|\nabla\ell_{\mathcal{B}_t}(x_t)\right\|^2} \\
&= \frac{1}{\nu}\cdot\left(1 - \frac{\frac{1}{|\mathcal{B}_t|}\operatorname{Tr}\operatorname{Var}[\nabla f(x_t)]}{\mathbb{E}\left\|\nabla\ell_{\mathcal{B}_t}(x_t)\right\|^2}\right) \\
&\geq \frac{1-\theta^2}{\nu}.
\end{aligned}
$$

Thus, the optimal stepsize for big batch SGD is the optimal stepsize for deterministic gradient descent scaled down by at most $1-\theta^2$.