The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning

Siyuan Ma, Raef Bassily, Mikhail Belkin
Department of Computer Science and Engineering
The Ohio State University
{ma.588, bassily.1}@osu.edu, *mbelkin@cse.ohio-state.edu*

December 18, 2017

Abstract

Stochastic Gradient Descent (SGD) with small mini-batch is a key component in modern large-scale machine learning. However, its efficiency has not been easy to analyze as most theoretical results require adaptive rates and show convergence rates far slower than that for gradient descent, making computational comparisons difficult.

In this paper we aim to clarify the issue of fast SGD convergence. The key observation is that most modern architectures are over-parametrized and are trained to interpolate the data by driving the empirical loss (classification and regression) close to zero. While it is still unclear why these interpolated solutions perform well on test data, these regimes allow for very fast convergence of SGD, comparable in the number of iterations to gradient descent.

Specifically, consider the setting with quadratic objective function, or near a minimum, where the quadratic term is dominant. We show that:

- Mini-batch size 1 with constant step size is optimal in terms of computations to achieve a given error.
- There is a critical mini-batch size such that:
  - (linear scaling) SGD iteration with mini-batch size m smaller than the critical size is nearly equivalent to m iterations of mini-batch size 1.
  - (saturation) SGD iteration with mini-batch larger than the critical size is nearly equivalent to a gradient descent step.

The critical mini-batch size can be viewed as the limit for effective mini-batch parallelization. It is also nearly independent of the data size, implying $O(n)$ acceleration over GD per unit of computation. We give experimental evidence on real data, with the results closely following our theoretical analyses.

Finally, we show how the interpolation perspective and our results fit with recent developments in training deep neural networks and discuss connections to adaptive rates for SGD and variance reduction.

1 Introduction

Most machine learning techniques for supervised learning are based on Empirical Loss Minimization (ERM), i.e., minimizing the loss $\mathscr{L}(w) \triangleq \frac{1}{n}\sum_{i=1}^n \ell_i(w)$ over some parametrized space of functions f_w. Here $\ell_i(w) = L(f_w(x_i), y_i)$, where (x_i, y_i) are the data and L could, for example, be the square loss $L(f_w(x), y) = (f_w(x) - y)^2$.

In recent years Stochastic Gradient Descent (SGD) with a small mini-batch size has become the backbone of machine learning, used in nearly all large-scale applications of machine learning methods, notably in conjunction with deep neural networks. Mini-batch SGD is a first order method which, instead of computing the full gradient of $\mathscr{L}(w)$, computes the gradient with respect to a certain subset of the data points, often chosen sequentially. While in practice SGD consistently outperforms full gradient descent (GD) by a large factor, the theoretical evidence is mixed. SGD requires less computations per iteration, however most analyses suggest that it needs adaptive step sizes and has the rate of convergence that is far slower than that of GD, making computational comparisons difficult.

In this paper we aim to clarify the reasons for the effectiveness of SGD by taking a different perspective. Our key observation is that most of modern machine learning, especially deep learning, relies on classifiers which are trained to achieve near zero classification and regression losses on the training data. Indeed, the goal of achieving near-perfect fit on the training set is stated explicitly by the practitioners as a best practice in supervised learning[1], see, e.g., the tutorial [Sal17]. The ability to achieve near-zero loss is provided by over-parametrization. The number of parameters for most deep architectures is very large and often exceeds by far the size of the datasets used for training (see, e.g., [CPC16] for a summary of different architectures). There is significant theoretical and empirical evidence that in such over-parametrized systems most or all local minima are also global and hence correspond to the regime where the output of the learning algorithm matches the labels exactly [GAGN15, CCSL16, ZBH+16, HLWvdM16, SEG+17, BFT17]. Since continuous loss functions are typically used for training, the resulting function *interpolates* the data[2], i.e., $f_{w^*}(x_i) \approx y_i$.

While we do not yet understand why these interpolated classifiers generalize so well to unseen data, there is ample evidence for their excellent performance in deep neural networks [GAGN15, CCSL16, ZBH+16, HLWvdM16, SEG+17, BFT17], kernels [MB17] and boosting [SFBL98]. In this paper we look at the significant computational implications of this startling phenomenon for stochastic gradient descent.

We start by observing that in the interpolated regime, SGD converges exponentially fast for convex functions. This observation is not new, going back to the Kaczmarz method [Kac37] for quadratic functions, more recently analyzed in [SV09]. For the general convex case it was first proved in [MB11], the rate was later improved in [NWS14]. However, to the best of our knowledge, exponential convergence in that regime has not yet been connected to over-parametrization in modern machine learning. Still, exponential convergence on its own does not allow us to make any comparisons between the computational efficiency of SGD with different mini-batch sizes and full gradient descent as the rate of convergence depends on the mini-batch size m. This is a key issue as full gradient descent is rarely, if ever, used in practice due to its inefficiency. There have been some attempts to understand the interplay between the mini-batch size and computational efficiency, e.g., [TBRS13, LZCS14] in the standard non-interpolated regime. However, in that setting the issue of bridging the exponential convergence of full GD and the much slower convergence rates of mini-batch SGD is difficult to resolve.

We will provide a detailed analysis for the rates of convergence and computational efficiency for different mini-batch sizes and a discussion of its implications in the context of modern machine learning. Specifically, we will consider the setting when the loss function is either quadratic (i.e., kernel or linear methods with the square loss) or a smooth function in a neighborhood of a minimum, where the higher order terms can be ignored[3]. The second setting is likely to be the case in many non-linear situations, including neural networks, once the iteration approaches a global minimum.

Under that assumption we show that:

1. Mini-batch of size 1 with a constant step size is optimal in terms of computations required to achieve a given error. Thus, in a fully sequential computer, first order optimization should

[1]Potentially using regularization at a later stage, after a near-perfect fit is achieved.

[2]Most of these architectures should be able to achieve perfect interpolation, $f_{w^*}(x_i) = y_i$. In practice, of course, it is not possible even for linear systems due to the computational and numerical limitations.

[3]In deep learning, pre-training/warmup techniques attempt to find an initial point near an optimal solution [Sal17, GDG+17].

be implemented as a sequence of gradient descents one data point at a time (cf. the classical Perceptron algorithm and the Kaczmarz method).

2. Let $m^* = \frac{trH}{\lambda_1(H)}$, where H is the Hessian at the critical point. Then the following holds:
   (a) (linear scaling) One SGD iteration with mini-batch of size $m \leq m^*$ is equivalent to m iterations of mini-batch of size 1 up to a multiplicative constant close to 1.
   (b) (saturation) One SGD iteration with a mini-batch of size $m > m^*$ is equivalent to an iteration of full gradient descent up to a multiplicative constant less than 4.

We see that the critical mini-batch size m^* can be viewed as the limit for the effective parallelization of mini-batch computations. If an iteration with mini-batch of size $m \leq m^*$ can be computed in parallel, it is nearly equivalent to m sequential steps with mini-batch of size 1. For $m > m^*$ parallel computation has limited added value. Our theoretical results are based on upper bounds which we show to be tight.

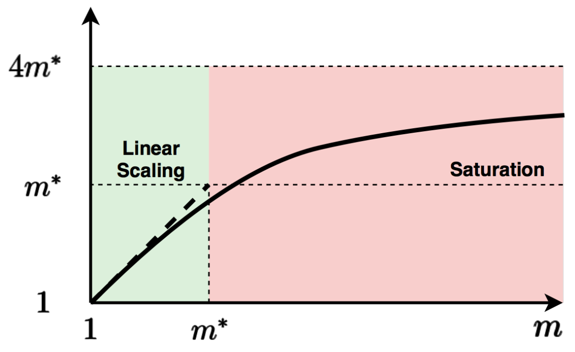


Figure 1: Number of iterations with batch size 1 (the y axis) equivalent to one iteration with batch size m.

Significantly, we show that m^* is nearly independent of the data size n (depending only on the ratio of the trace and the top eigenvalue of the Hessian). Thus SGD with mini-batch size m^* (typically a small constant) gives essentially the same convergence per iteration as full gradient descent, implying speed up factor of $O(n)$ over GD per unit of computation.

We provide experimental evidence corroborating this on real data. In particular, we demonstrate the regimes of linear scaling and saturation and also show that on real data m^* is in line with our estimate. It is typically several orders of magnitude smaller than the data size n implying a computational advantage of at least 10^3 factor over full gradient descent in realistic scenarios in the over-parametrized (or fully parametrized) setting. We believe this sheds light on the impressive effectiveness of SGD observed in many real-world situation and is the reason why full gradient descent is rarely, if ever, used. In particular, "linear scaling rule" recently used in deep convolutional networks [Kri14, GDG$^+$17, YGG17, SKL17] seems consistent with our theoretical analyses.

The rest of the paper is organized as follows: In Section 3 we discuss exponential convergence of SGD and some implications for the variance reduction techniques. It turns out that in the interpolated regime, simple SGD with constant step size is equally or more effective than the more complex variance reduction methods.

Section 4 contains the main results of our paper analyzing optimal convergence, computational efficiency and optimal step sizes for different mini-batch sizes.

In Section 5 we provide experimental evidence using several datasets. We show that the experimental results correspond closely to the behavior predicted by our bounds. We also briefly discuss the connection to the linear scaling rule in neural networks.

2 Preliminaries

Before we start our technical discussion, we briefly overview some standard notions in convex analysis. Here, we will focus on differentiable convex functions, however, the definitions below extend to general functions simply by replacing the gradient of the function at a given point to by the set of all sub-gradients at that point. In fact, since in this paper we only consider smooth functions, differentiability is directly implied.

- A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ is *convex* on $\mathbb{R}^d$ if, for all $w, v \in \mathbb{R}^d$, we have $\ell(v) \geq \ell(w) + \langle \nabla \ell(w), v - w \rangle$.
- Let $\beta > 0$. A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ is β-*smooth* on $\mathbb{R}^d$ if, for all $w, v \in \mathbb{R}^d$, we have $\ell(v) \leq \ell(w) + \langle \nabla \ell(w), v - w \rangle + \frac{\beta}{2} \|v - w\|^2$, where $\nabla \ell(w)$ denotes the gradient of ℓ at w.
- Let $\alpha > 0$. A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ is α-*strongly convex* on $\mathbb{R}^d$ if, for all $w, v \in \mathbb{R}^d$, we have $\ell(v) \geq \ell(w) + \langle \nabla \ell(w), v - w \rangle + \frac{\alpha}{2} \|v - w\|^2$. (Clearly, for any $\alpha \geq 0$, α-strong convexity implies convexity).

The problem of unconstrained Empirical Risk Minimization (ERM) can be described as follows: Given a set of n loss functions $\ell_i : \mathbb{R}^d \to \mathbb{R}$, $i \in \{1, \ldots, n\}$, the goal is to minimize the empirical loss function defined as

$$\mathscr{L}(w) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell_i(w),\ w \in \mathbb{R}^d.$$

In particular, we want to find a minimizer $w^* \triangleq \arg\min_{w \in \mathbb{R}^d} \mathscr{L}(w)$. In the context of supervised learning, given a training set $\{(x_i, y_i) : 1 \leq i \leq n\}$ of n (feature vector, target) pairs, one can think of $\ell_i(w)$ as the cost incurred in choosing a parameter vector w to fit the data point (x_i, y_i). In particular, in this context, minimizing $\mathscr{L}$ over $w \in \mathbb{R}^d$ is equivalent to minimizing $\mathscr{L}$ over a parameterized space of functions $\{f_w : w \in \mathbb{R}^d\}$, where each f_w maps a feature vector x to a target y. Thus, in this case, for each i, $\ell_i(w)$ can be written as $L(f_w(x_i), y_i)$ where L is some cost function that represents how far is $f_w(x_i)$ from y_i, for example, $L(\cdot, \cdot)$ could be the squared loss $L(f_w(x), y) = (f_w(x) - y)^2$.

3 Interpolation and Fast SGD: Warm Up

Before proceeding with our analysis of the optimal convergence and computational efficiency for mini-batch SGD, we revisit a result from [MB11, NWS14] that implies exponential convergence for SGD in the interpolation setting. This result is important for our further developments. It also has a number of interesting implications for variance reduction techniques that, to the best of our knowledge, have not been observed before. We discuss these below in Subsection 3.1.

We consider a standard setting of ERM where for all $1 \leq i \leq n$, ℓ_i is non-negative, β-smooth and convex. Moreover, $\mathscr{L}(w) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(w)$ is α-strongly convex.

The above setting is naturally satisfied in many problems, for example, in least-squares linear regression with full rank sample covariance matrix.

Next, we state our key assumption in this work. This assumption describes the interpolation setting, which is aligned with what we usually observe in over-parametrized settings in modern machine learning.

Assumption 1 (Interpolation)**.** Let $w^* \in \operatorname{argmin}_{w \in \mathbb{R}^d} \mathscr{L}(w)$. Then, for all $1 \leq i \leq n$, $\ell_i(w^*) = 0$.

Note that instead of assuming that $\ell_i(w^*) = 0$, it suffices to assume that w^* is the minimizer of all ℓ_i. By subtracting from each ℓ_i the offset $\ell_i(w^*)$, we get an equivalent minimization problem where the new losses are all non-negative, and are all zero at w^*.

Consider the SGD algorithm that starts at an arbitrary $w_0 \in \mathbb{R}^d$, and at each iteration t draws an index $i_t \leftarrow \{1, \ldots, n\}$ uniformly at random, then takes a step:

$$w_{t+1} = w_t - \frac{1}{\beta} \nabla \ell_{i_t}(w_t). \tag{1}$$

The theorem below shows exponential convergence for SGD in the interpolated regime. It is a special (and simplified) case of Theorem 2.1 in [NWS14], which is a sharper version of Theorem 1 in [MB11]. For completeness, we provide a simple proof in the Appendix 1.

Theorem 1. *For the setting described above and under Assumption 1, for any $t \in \mathbb{N}$, the SGD with update step 1 gives the following guarantee*

$$\mathop{\mathbb{E}}_{w_t} \left[\mathscr{L}(w_t) \right] \leq \frac{\beta}{2} \mathop{\mathbb{E}}_{w_t} \left[\|w_t - w^*\|^2 \right] \leq \frac{\beta}{2} (1 - \frac{\alpha}{\beta})^t \|w_0 - w^*\|^2$$

3.1 Variance reduction methods in the interpolation regime

For general convex optimization, a set of important stochastic methods [RSB12, JZ13, DBLJ14, XZ14, AZ16] have been proposed to achieve exponential (linear) convergence rate with constant step size. The effectiveness of these methods derives from their ability to reduce the stochastic variance caused by sampling. In a general convex setting, this variance prevents SGD from both adopting a constant step size and achieving an exponential convergence rate.

Remarkably, in the interpolated regime, Theorem 1 implies that SGD obtains the benefits of variance reduction "for free" without the need for any modification or extra information (e.g., full gradient computations for variance reduction). The table on the right compares the convergence of SGD in the interpolation setting with several popular variance reduction methods. Overall, SGD has the largest step size and achieves the fastest convergence rate without the need for any further assumptions. The only comparable or faster rate is given by Katyusha, which is an accelerated SGD method combining momentum and variance reduction for faster convergence.

Method	Step size	#Iterations to reach a given error
SGD (Eq. 1)	$\frac{1}{\beta}$	$O(\frac{\beta}{\alpha})$
SAG [RSB12]	$\frac{1}{2n\cdot\beta}$	$O(\frac{n\cdot\beta}{\alpha})$
SVRG [JZ13]	$\frac{1}{10\beta}$	$O(n+\frac{\beta}{\alpha})$
SAGA [DBLJ14]	$\frac{1}{3\beta}$	$O(n+\frac{\beta}{\alpha})$
Katyusha [AZ16] (momentum)	adaptive	$O(n+\sqrt{\frac{n\cdot\beta}{\alpha}})$

4 How fast is fast SGD: analysis of step, mini-batch sizes and computational efficiency

In this section, we state and discuss our main technical results. We will analyze the efficiency of SGD in the interpolation regime as a function of the mini-batch size m given some additional information about the Hessian. We will consider the following key questions:

- What is the optimal convergence rate of mini-batch SGD and the corresponding step size as a function of m?
- What is the computational efficiency of different batch sizes and how do they compare to full GD?

We will consider the standard update rule with a constant step size η:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla \frac{1}{m} \sum_{j=1}^{m} \ell_{i_t^{(j)}}(\boldsymbol{w}_t) \tag{2}$$

where m is the size of a mini-batch of data points whose indices $\{i_t^{(1)}, \ldots, i_t^{(m)}\}$ are drawn uniformly with replacement at each iteration t from $\{1, \ldots, n\}$.

We will explore the setting, where the loss function is quadratic. This case covers over-parametrized linear or kernel regression with a positive definite kernel. The quadratic case also applies to general smooth convex functions in the neighborhood of a minimum where higher order terms can be ignored. Thus the results of our analysis would hold for SGD in the proximity of an interpolating minimum of a general smooth function.

Quadratic loss. Consider the problem of minimizing the sum of least squares

$$\mathscr{L}(\boldsymbol{w}) \triangleq \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 \tag{3}$$

where $(\boldsymbol{x}_i, y_i) \in \mathscr{H} \times \mathbb{R}, i = 1, \ldots, n$ are labeled data points sampled from some (unknown) distribution. In the interpolation setting, there exists $\boldsymbol{w}^* \in \mathscr{H}$ such that $L(\boldsymbol{w}^*) = 0$. In this setting, we assume, w.l.o.g., that the covariance $H \triangleq \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^T$ has eigen decomposition $\sum_{i=1}^{n}\lambda_i\boldsymbol{e}_i\boldsymbol{e}_i^T$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. We further assume that for all feature vectors $\boldsymbol{x}_i, i = 1, \ldots, n$, we have $\|\boldsymbol{x}_i\| \leq \beta$. Note that this implies that the trace of H is bounded from above by β, that is, $\mathrm{tr}(H) \leq \beta$. Thus, we have $\beta > \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.

To minimize the loss in this setting, consider the following SGD with mini-batch of size m and step size η:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta H_m(\boldsymbol{w}_t - \boldsymbol{w}^*) \tag{4}$$

where $H_m \triangleq \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)T}$ is a subsample covariance corresponding to subsamples $(\boldsymbol{x}^{(i)}, y^{(i)})$.

4.1 Upper bound on the expected empirical loss

The following theorem provides an upper bound on the expected empirical loss after t iterations of mini-batch SGD whose update step is given by (4).

Theorem 2. *For any $\lambda \in [\lambda_n, \lambda_1], m \in \mathbb{N}$, and $0 < \eta < \frac{2m}{\beta + (m-1)\lambda_1}$ define*

$$g(\lambda; m, \eta) \triangleq (1 - \eta\lambda)^2 + \frac{\eta^2\lambda}{m}(\beta - \lambda).$$

Let $g(m, \eta) \triangleq \max_{\lambda \in [\lambda_n, \lambda_1]} g(\lambda; m, \eta)$. In the interpolation setting, for any $t \geq 1$, the mini-batch SGD with update step (4) yields the following guarantee

$$\mathbb{E}[\mathscr{L}(w_t)] \leq \lambda_1 \cdot \mathbb{E}\Big[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2\Big] \leq \lambda_1 \cdot (g(m, \eta))^t \cdot \mathbb{E}\Big[\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2\Big]$$

Proof. By reordering terms in the update equation 4 and using the independence of H_m and $\boldsymbol{w}_t$, the variance in the parameter update can be written as

$$\mathbb{E}\Big[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2\Big] = \mathbb{E}\big[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T(I - 2\eta H + \eta^2\mathbb{E}\big[H_m^2\big])(\boldsymbol{w}_t - \boldsymbol{w}^*)\big]$$

To obtain an upper bound, we need to bound $\mathbb{E}\big[H_m^2\big]$. Notice that $H_m = \sum_{i=1}^{m}H_1^{(i)}$ where $H_1^{(i)}, i = 1, \ldots, m$ are rank-1 independent subsample covariances. Expanding H_m accordingly yield

$$\mathbb{E}\big[H_m^2\big] = \frac{1}{m}\mathbb{E}\big[H_1^2\big] + \frac{m-1}{m}H^2 \preceq \frac{\beta}{m}H + \frac{m-1}{m}H^2 \tag{5}$$

Let $G_{m,\eta} \triangleq I - 2\eta H + \eta^2(\frac{\beta}{m}H + \frac{m-1}{m}H^2)$. Then the variance is bounded as

$$\mathbb{E}\Big[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2\Big] \leq \mathbb{E}\big[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T G_{m,\eta}(\boldsymbol{w}_t - \boldsymbol{w}^*)\big]$$

Clearly, $\lim_{t\to\infty}\mathbb{E}\Big[\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2\Big] = 0$ if

$$\|G_{m,\eta}\| < 1 \Leftrightarrow \eta < \eta_1(m) \triangleq \frac{2m}{\beta + (m-1)\lambda_1} \tag{6}$$

Furthermore, the convergence rate relies on the eigenvalues of $G_{m,\eta}$. Let λ be an eigenvalue of H, then the corresponding eigenvalue of $G_{m,\eta}$ is given by

$$g(\lambda; m, \eta) = 1 - 2\eta\lambda + \eta^2[\frac{\beta}{m}\lambda + (1 - \frac{1}{m})\lambda^2] = (1 - \eta\lambda)^2 + \frac{\eta^2\lambda}{m}(\beta - \lambda)$$

When the step size η and mini-batch size m are chosen (satisfying Eq. 6), we have

$$\mathbb{E}\left[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2\right] \leq g(m, \eta) \cdot \mathbb{E}\left[\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2\right]$$

where

$$g(m, \eta) \triangleq \max_{\lambda \in \{\lambda_1, \ldots, \lambda_n\}} g(\lambda; m, \eta).$$

Finally, observe that

$$\mathscr{L}(\boldsymbol{w}_{t+1}) \leq \lambda_1 \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 \tag{7}$$

since the quadratic loss $\mathscr{L}$ is obviously $2\lambda_1$-smooth. □

4.2 Lower bound on the expected empirical loss

We now show that our upper bound given above is indeed tight in the interpolation setting for the class of quadratic loss functions defined in (3). Namely, we give a specific instance of (3) where the upper bound in Theorem 2 is tight.

Theorem 3. *There is a data set $\{(\boldsymbol{x}_i, y_i) \in \mathscr{H} \times \mathbb{R} : 1 \leq i \leq n\}$ such that the mini-batch SGD with update step (4) yields the following lower bound on the expected empirical quadratic loss $\mathscr{L}(\boldsymbol{w})$ (3)*

$$\mathbb{E}[\mathscr{L}(w_t)] \geq \lambda_1 \cdot \mathbb{E}\left[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2\right] = \lambda_1 \cdot (g(m, \eta))^t \cdot \mathbb{E}\left[\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2\right]$$

Proof. We start the proof by observing that there are only two places in the proof of Theorem 2 where the upper bound may not be tight, namely, the last inequality in 5 and inequality (7). Consider a data set where all the feature vectors $\boldsymbol{x}_i, i = 1, \ldots, n$, lie on the sphere of radius β, that is, $\|\boldsymbol{x}_i\| = \beta$, $\forall i = 1, \ldots, n$. We note that the last inequality in 5 in the proof of Theorem 2 is tight in that setting. Suppose that, additionally, we choose the feature vectors such that the eigenvalues of the sample covariance matrix H are all equal, that is, $\lambda_1 = \lambda_2 = \ldots = \lambda_n$. This can be done, for example, by choosing all the feature vectors to be orthogonal (note that this is possible in the fully parametrized setting). Hence, in this case, since $\mathscr{L}$ is λ_1-strongly convex, we have

$$\mathscr{L}(\boldsymbol{w}_{t+1}) \geq \lambda_1 \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2$$

which shows that that inequality (7) is also achieved with equality in that setting. This completes the proof. □

Remark. From the experimental results, it appears that our upper bound can be close to tight even in some settings when the eigenvalues are far apart. We plan to investigate this phenomenon further.

4.3 Optimal step size for a given batch size

To fully answer the first question we posed at the beginning of this section, we will derive an optimal rule for choosing the step size as a function of the batch size. Specifically, we want to find step size $\eta^*(m)$ to achieve fastest convergence. Given Theorem 2, our task reduces to finding the minimizer

$$\eta^*(m) \arg \min_{\eta < \frac{2}{\frac{\beta}{m} + \frac{m-1}{m}\lambda_1}} g(m, \eta) \tag{8}$$

Let $g^*(m)$ denote the resulting minimum, that is, $g^*(m) = g(m, \eta^*(m))$.

The resulting expression for the minimizer $\eta^*(m)$ generally depends on the least eigenvalue of the Hessian matrix. In situations where we don't have a good estimate for this eigenvalue (which can be close to zero in practice), one would rather have a step size that is independent of the smallest eigenvalue. We obtain a near-optimal approximation for step size with no dependence on the smallest eigenvalue.

The following theorem characterizes the optimal step size and the resulting $g^*(m)$.

Theorem 4 (Optimal step size role as function of batch size)**.** *For every batch size m, the optimal step size function $\eta^*(m)$ and convergence rate function $g^*(m)$ are given by:*

$$\eta^*(m) = \begin{cases} \frac{m}{\beta+(m-1)\lambda_n} & m \le \frac{\beta}{\lambda_1-\lambda_n}+1 \\ \frac{2m}{\beta+(m-1)(\lambda_1+\lambda_n)} & m > \frac{\beta}{\lambda_1-\lambda_n}+1 \end{cases} \tag{9}$$

$$g^*(m) = \begin{cases} 1-\frac{m\lambda_n}{\beta+(m-1)\lambda_n} & m \le \frac{\beta}{\lambda_1-\lambda_n}+1 \\ 1-4\frac{m(m-1)\lambda_1\lambda_n}{(\beta+(m-1)(\lambda_1+\lambda_n))^2} & m > \frac{\beta}{\lambda_1-\lambda_n}+1 \end{cases} \tag{10}$$

Note that if $\lambda_1 = \lambda_n$, then the first case in each expression will be valid for all $m \ge 1$.

The proof of the above theorem follows from the following two lemmas.

Lemma 1. *Let $\eta_0(m) \triangleq \frac{2m}{\beta+(m-1)(\lambda_1+\lambda_n)}$, and let $\eta_1(m) \triangleq \frac{2m}{\beta+(m-1)\lambda_1}$. Then,*

$$g(m,\eta) = \begin{cases} g^{\mathrm{I}}(m,\eta) \triangleq g(\lambda_n; m,\eta) & \eta \le \eta_0(m) \\ g^{\mathrm{II}}(m,\eta) \triangleq g(\lambda_1; m,\eta) & \eta_0(m) < \eta \le \eta_1(m) \end{cases}$$

Proof. For any fixed $m \ge 1$ and $\eta < \eta_1(m)$, observe that $g(\lambda; m, \eta)$ is a quadratic function of λ. Hence, the maximum must occur at either $\lambda = \lambda_n$ or $\lambda = \lambda_1$. Define $g^{\mathrm{I}}(m,\eta) \triangleq g(\lambda_n; m,\eta)$ and $g^{\mathrm{II}}(m,\eta) \triangleq g(\lambda_1; m,\eta)$. Now, depending on the value of m and η, we would either have $g^{\mathrm{I}}(m,\eta) \ge g^{\mathrm{II}}(m,\eta)$ or $g^{\mathrm{I}}(m,\eta) < g^{\mathrm{II}}(m,\eta)$. In particular, it is not hard to show that

$$g^{\mathrm{I}}(m,\eta) \ge g^{\mathrm{II}}(m,\eta) \Leftrightarrow \eta \le \eta_0(m),$$

where $\eta_0(m) \triangleq \frac{2m}{\beta+(m-1)(\lambda_1+\lambda_n)}$. This completes the proof. □

Lemma 2. *Given the quantities defined in Lemma 1, let $\eta^{\mathrm{I}}(m) = \underset{\eta \le \eta_0(m)}{\operatorname{argmin}}\, g^{\mathrm{I}}(m,\eta)$, and $\eta^{\mathrm{II}}(m) = \underset{\eta_0(m)<\eta\le\eta_1(m)}{\operatorname{argmin}}\, g^{\mathrm{II}}(m,\eta)$. Then, we have*

1. *For all $m \ge 1$, $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right) \le g^{\mathrm{II}}\left(m, \eta^{\mathrm{II}}(m)\right)$.*
2. *For all $m \ge 1$, $\eta^{\mathrm{I}}(m) = \eta^*(m)$ and $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right) = g^*(m)$, where $\eta^*(m)$ and $g^*(m)$ are as given by (9) and (10), respectively, (in Theorem 4).*

Proof. First, consider $g^{\mathrm{I}}(m,\eta)$. For any fixed m, it is not hard to show that the minimizer of $g^{\mathrm{I}}(m,\eta)$ as a function of η, constrained to $\eta \le \eta_0(m)$, is given by $\min\left(\eta_0(m), \frac{m}{\beta+(m-1)\lambda_n}\right) \triangleq \eta^{\mathrm{I}}(m)$. That is,

$$\eta^{\mathrm{I}}(m) = \begin{cases} \frac{m}{\beta+(m-1)\lambda_n} & m \le \frac{\beta}{\lambda_1-\lambda_n}+1 \\ \eta_0(m) = \frac{2m}{\beta+(m-1)(\lambda_1+\lambda_n)} & m > \frac{\beta}{\lambda_1-\lambda_n}+1 \end{cases}$$

Substituting $\eta = \eta^{\mathrm{I}}(m)$ in $g^{\mathrm{I}}(m,\eta)$, we get

$$g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right) = \begin{cases} 1-\frac{m\lambda_n}{\beta+(m-1)\lambda_n} & m \le \frac{\beta}{\lambda_1-\lambda_n}+1 \\ 1-4\frac{m(m-1)\lambda_1\lambda_n}{(\beta+(m-1)(\lambda_1+\lambda_n))^2} & m > \frac{\beta}{\lambda_1-\lambda_n}+1 \end{cases}$$

Note that $\eta^{\mathrm{I}}(m)$ and $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right)$ are equal to $\eta^*(m)$ and $g^*(m)$ given in Theorem 4, respectively. This proves item 2 of the lemma.

Next, consider $g^{\mathrm{II}}(m, \eta)$. Again, for any fixed m, one can easily show that the minimum of $g^{\mathrm{II}}(m, \eta)$ as a function of η, constrained to $\eta_0(m) < \eta \leq \eta_1(m)$, is actually achieved at the boundary $\eta = \eta_0(m)$. Hence, $\eta^{\mathrm{II}}(m) = \eta_0(m)$. Substituting this in $g^{\mathrm{II}}(m, \eta)$, we get

$$g^{\mathrm{II}}\left(m, \eta^{\mathrm{II}}(m)\right) = 1 - 4\frac{m(m-1)\lambda_1\lambda_n}{\left(\beta + (m-1)(\lambda_1 + \lambda_n)\right)^2}, \quad \forall m \geq 1.$$

We conclude the proof by showing that for all $m \geq 1$, $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right) \leq g^{\mathrm{II}}\left(m, \eta^{\mathrm{II}}(m)\right)$. Note that for $m > \frac{\beta}{\lambda_1 - \lambda_n} + 1$, $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right)$ and $g^{\mathrm{II}}\left(m, \eta^{\mathrm{II}}(m)\right)$ are identical. For $m \leq \frac{\beta}{\lambda_1 - \lambda_n} + 1$, given the expressions above, one can verify that $g^{\mathrm{I}}\left(m, \eta^{\mathrm{I}}(m)\right) \leq g^{\mathrm{II}}\left(m, \eta^{\mathrm{I}}(m)\right)$.

□

Proof of Theorem 4: Given Lemma 1 and item 1 of Lemma 2, it follows that $\eta^{\mathrm{I}}(m)$ is the minimizer $\eta^*(m)$ given by (8). Item 2 of Lemma 2 concludes the proof of the theorem.

Nearly optimal step size with no dependence on λ_n: In practice, it is usually easy to obtain a good estimate for λ_1, but it is hard to estimate λ_n. That is why, one would typically want to avoid dependence on λ_n in the SGD algorithm. Following the above derivation of optimal $\eta^*(m)$, we can easily find a close approximation that has no dependence on λ_n and essentially does not cost us anything in terms of the convergence rate. The following theorem provides such approximation.

Theorem 5. *Let $\hat{\eta}(m)$ be defined as:*

$$\hat{\eta}(m) = \begin{cases} \frac{m}{\beta(1+(m-1)/n)} & m \leq \frac{\beta}{\lambda_1 - \beta/n} + 1 \\ \frac{2m}{\beta + (m-1)(\lambda_1 + \beta/n)} & m > \frac{\beta}{\lambda_1 - \beta/n} + 1 \end{cases} \tag{11}$$

The step size $\hat{\eta}(m)$ yields the following upper bound on $g\left(m, \hat{\eta}(m)\right)$, denoted as $\hat{g}(m)$:

$$\hat{g}\left(m\right) = \begin{cases} 1 - \frac{m\lambda_n}{\beta(1+(m-1)/n)} & m \leq \frac{\beta}{\lambda_1 - \beta/n} + 1 \\ 1 - 4\frac{m(m-1)\lambda_1\lambda_n}{(\beta+(m-1)(\lambda_1+\beta/n))^2} & m > \frac{\beta}{\lambda_1 - \beta/n} + 1 \end{cases} \tag{12}$$

Proof. The proof easily follows from the fact that $\hat{\eta}(m)$ lies in the feasible region for the minimization problem in (8). In particular, $\hat{\eta}(m) \leq \eta_0(m)$, where $\eta_0(m)$ is as defined in Lemma 1. The upper bound $\hat{g}\left(m\right)$ follows from substituting $\hat{\eta}(m)$ in $g^{\mathrm{I}}(m, \eta)$ defined in Lemma 1, then upper-bounding the resulting expression. □

It is easy to see that the convergence rate resulting from $\hat{\eta}$ is at most factor $O(1 + 1/n)$ slower than the optimal rate resulting from $\eta^*(m)$ (assuming $m \ll n$, in particular, assuming m is a constant that does not scale with n). This is clearly negligible difference. Note that since n is typically very large, the near-optimal step size is approximately given by $\hat{\eta}(m) \approx m/\beta$ when $m \lessapprox \beta/\lambda_1$ and $\approx \frac{2m}{\beta + (m-1)\lambda_1}$ when $m \gtrapprox \beta/\lambda_1$.

4.4 Batch size selection

Now we will derive the optimal batch size given a fixed computational budget in terms of the computational efficiency defined as the number of gradient computations to obtain a fixed desired accuracy. We will show that single point batch is in fact optimal in that setting. Moreover, we will show that any mini-batch size in the range from 1 to a certain constant m^* independent of n, is nearly optimal in terms of gradient computations. Interestingly for values beyond m^* the computational efficiency drops sharply. This result has direct implications for the batch size selection in parallel computation.

4.4.1 Optimality of a single-point batch (standard SGD)

Suppose we are limited by a fixed number of gradient computations. Then, what would be the batch size that yields the least approximation error? Equivalently, suppose we are required to achieve a certain target accuracy ϵ (i.e., want to reach parameter $\hat{\boldsymbol{w}}$ such that $\mathscr{L}(\hat{\boldsymbol{w}}) - \mathscr{L}(\boldsymbol{w}^*) \leq \epsilon$). Then, again, what would be the optimal batch size that yields the least amount of computation.

Suppose we are being charged a unit cost for each gradient computation, then it is not hard to see that the cost function we seek to minimize is $g^*(m)^{\frac{1}{m}}$, where $g^*(m)$ is as given by Theorem 4. To see this, note that for a batch size m, the number of iterations to reach a fixed desired accuracy is $t(m) = \frac{\text{constant}}{\log(1/g^*(m))}$. Hence, the computation cost is $m \cdot t(m) = \frac{\text{constant}}{\log(1/g^*(m))^{1/m}}$. Hence, minimizing the computation cost is tantamount to minimizing $g^*(m)^{1/m}$. The following theorem shows that the exact minimizer is $m = 1$. Later, we will see that *any* value for m from 2 to $\approx \beta/\lambda_1$ is actually not far from optimal. So, if we have cheap or free computation available (e.g., parallel computation), then it would make sense to choose $m \approx \beta/\lambda_1$. We will provide more details in the following subsection.

Theorem 6 (Optimal batch size under a limited computational budget)**.** *When we are charged a unit cost per gradient computation, the batch size that minimizes the overall computational cost required to achieve a fixed accuracy (i.e., maximizes the computational efficiency) is $m = 1$. Namely,*

$$\arg\min_{m \in \mathbb{N}} g^*(m)^{\frac{1}{m}} = 1$$

The detailed and precise proof is deferred to the appendix. Here, we give a less formal but more intuitive argument based on a reasonable approximation for the objective function. Such approximation in fact is valid in most of the practical settings. In the appendix, we give the exact and detailed analysis. Note that $g^*(m)$ can be written as $1 - \frac{\lambda_n}{\beta} s(m)$, where $s(m)$ (which we call the speed-up factor) is given by

$$s(m) = \begin{cases} \frac{m}{1+(m-1)\frac{\lambda_n}{\beta}} & m \leq \frac{\beta}{\lambda_1 - \lambda_n} + 1 \\ \frac{4m(m-1)\lambda_1}{\beta\left(1+(m-1)\frac{\lambda_1+\lambda_n}{\beta}\right)^2} & m > \frac{\beta}{\lambda_1 - \lambda_n} + 1 \end{cases} \tag{13}$$

Proof outline: Note that $s(m)$ defined in (13) indeed captures the speed-up factor we gain in convergence relative to standard SGD (with $m = 1$) where the convergence is dictated by λ_n/β. Now, note that $g^*(m)^{1/m} \approx e^{-s(m)/m}$. This approximation becomes very accurate when $\lambda_n \ll \lambda_1$, which is typically the case for most of the practical settings where n is very large. Assuming this is the case (for the sake of this intuitive outline), minimizing $g^*(m)^{1/m}$ becomes equivalent to maximizing $s(m)/m$. Now, note that when $m \leq \frac{\beta}{\lambda_1 - \lambda_n} + 1$, then $s(m)/m \geq \frac{1}{1+\frac{m-1}{n}}$ (where the lower bound follows from the fact that $\beta/\lambda_n \leq 1/n$ since β is an upper bound on the trace of H and λ_n is the least eigen value of H). Hence, for $m = 1$, we have $s(m)/m = s(1) \geq 1$. On the other hand, when $m > \frac{\beta}{\lambda_1 - \lambda_n} + 1$, we have

$$s(m)/m = \frac{4(m-1)\lambda_1}{\beta\left(1 + (m-1)\frac{\lambda_1+\lambda_n}{\beta}\right)^2},$$

which is decreasing in m, and hence, it's upper bounded by its value at $m = \frac{\beta}{\lambda_1 - \lambda_n} + 1$. By direct substitution and simple cancellations, we reach an upper bound of $\frac{\lambda_1 - \lambda_n}{\lambda_1} < 1$. Thus, $m = 1$ is optimal.

One may wonder whether the above result is valid if the near-optimal step size $\hat{\eta}(m)$ (that does not depend on λ_n) is used. That is, one may ask whether the same optimality result is valid if the near optimal error rate function $\hat{g}(m)$ is used instead of $g^*(m)$ in Theorem 6. Indeed, we show that the same optimality remains true even if computational efficiency is measured with respect to $\hat{g}(m)$. This is formally stated in the following theorem.

Theorem 7. *When the near-optimal step size $\hat{\eta}(m)$ is used, the batch size that minimizes the overall computational cost required to achieve a fixed accuracy is $m = 1$. Namely,*

$$\arg\min_{m\in\mathbb{N}} \hat{g}(m)^{\frac{1}{m}} = 1$$

The proof of the above theorem follows similar lines of the proof of Theorem 6.

4.4.2 Near optimal larger batch sizes

Suppose that several gradient computations can be performed in parallel. Sometimes doubling the number of machines used in parallel can halve the number of iterations needed to reach a fixed desired accuracy. Such observation has motivated many works to use large batch size with distributed synchronized SGD [CMBJ16, GDG+17, YGG17, SKL17]. One critical problem in this large batch setting is how to choose the step size. To keep the same covariance, [BCN16, Li17, HHS17] choose the step size $\eta \sim \sqrt{m}$ for batch size m. While [Kri14, GDG+17, YGG17, SKL17] have observed that rescaling the step size $\eta \sim m$ works well in practice for not too large m. To explain these observations, we directly connect the parallelism, or the batch size m, to the required number of iterations $t(m)$ defined previously. It turns out that (a) when the batch size is small, doubling the size will almost halve the required iterations; (b) after the batch size surpasses certain value, increasing the size to any amount would only reduce the required iterations by at most a constant factor.

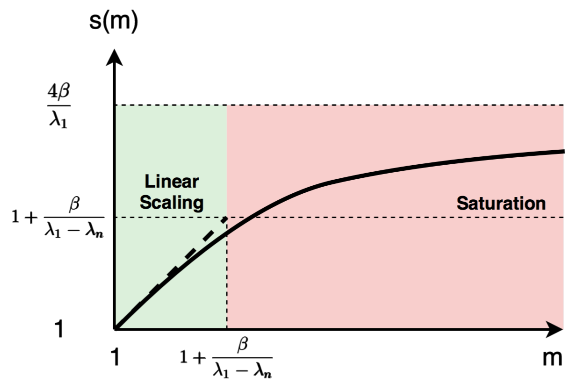


Figure 2: Factor of iterations saved by large batch $\frac{t(1)}{t(m)} \approx s(m)$

Our analysis uses the optimal step size and convergence rate in Theorem 4. Now consider the factor by which we save the number of iterations when increasing the batch size from 1 to m. Using the approximation $g^*(m)^{1/m} \approx e^{-s(m)/m}$, we have $\frac{t(1)}{t(m)} \approx s(m)$, the speed up factor defined in Eq. 13. The change of $s(m)$ is illustrated in Figure 2 where two regimes are highlighted:

Linear scaling regime ($m \le \frac{\beta}{\lambda_1-\lambda_n} + 1$): This is the regime where increasing the batch size m will quickly drive down $t(m)$ needed to reach certain accuracy. When $\lambda_n \ll \lambda_1$, $s(m) \approx m$, which suggests $t(m/2) \approx 2 \cdot t(m)$. In another word, doubling the batch size in this region will roughly halve the number of iterations needed. Note that we choose step size $\eta \leftarrow \frac{m}{\beta+(m-1)\lambda_n}$ in this region. When $\lambda_n \ll \lambda_1 \le \beta$, $\eta \sim m$, which is consistent with the linear scaling heuristic used in [Kri14, GDG+17, SKL17]. In this case, the largest batch size in the linear scaling regime can be practically calculated through

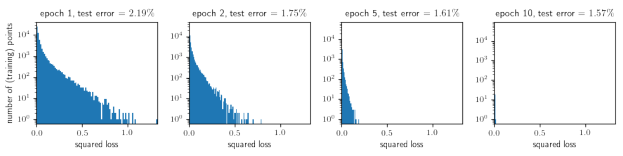
$$m^* = \frac{\beta}{\lambda_1-\lambda_n} + 1 \approx \frac{\beta}{\lambda_1-\beta/n} + 1 \approx \frac{\beta}{\lambda_1} + 1 \tag{14}$$

Saturation regime ($m > \frac{\beta}{\lambda_1 - \lambda_n} + 1$). Increasing batch size in this regime becomes much less beneficial. Although $s(m)$ is monotonically increasing, it is upper bounded by $\lim_{m\to\infty} s(m) = \frac{4\beta}{\lambda_1}$. In fact, since $t(\frac{\beta}{\lambda_1 - \lambda_n} + 1) / \lim_{m\to\infty} t(m) < 4$ for small λ_n, no batch size in this regime can reduce the needed iterations by a factor of more than 4.

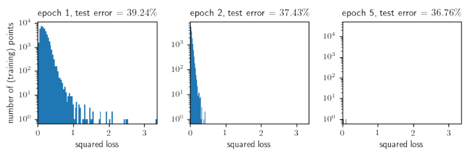
5 Experimental Results

This section will provide empirical evidence for interpolation and our theoretical results on the effectiveness of different mini-batch sizes in the kernel least squares regression setting for some standard positive definite kernels. In that case $\beta = 1$, λ_1 and m^* can be computed efficiently (see [MB17] for details).

5.1 Interpolation by kernels



(a) MNIST ($\sigma = 10$), 10 classes

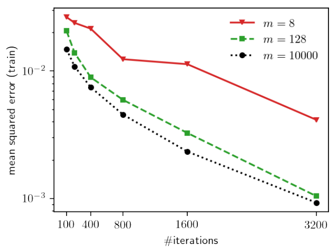


(b) TIMIT ($\sigma = 20$), 144 classes

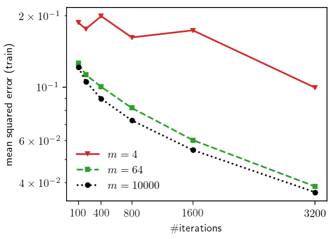
Figure 3: Histogram of training loss on data points at each epoch

We start by presenting an example of the interpolation phenomenon through training the EigenPro-modified Laplace kernel [MB17] on MNIST [LBBH98] and a subset (of size $5 \cdot 10^4$) of TIMIT [GLF+93]. The histograms in Fig. 3 show the number of points with a given loss calculated as $\|\boldsymbol{y}_i - f(\boldsymbol{x}_i)\|^2$ (on feature vector $\boldsymbol{x}_i$ and corresponding binary label vector $\boldsymbol{y}_i$). We see that we enter the interpolation regime after only a few epochs of SGD training, but the test loss is consistently decreasing.

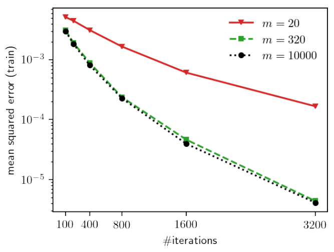
5.2 Comparison of SGD with the critical batch size m^* to full gradient descent



(a) MNIST (Gaussian, $\sigma = 5$) $\beta = 1, \lambda_1 = 0.15, m^* \approx 8$



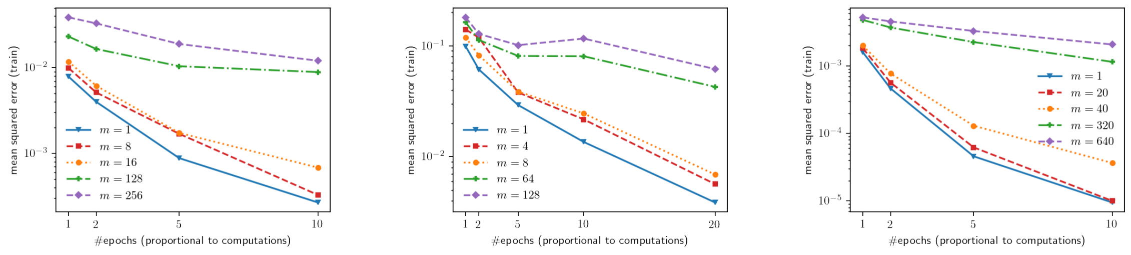
(b) HINT-S (Gaussian, $\sigma = 16$) $\beta = 1, \lambda_1 = 0.3, m^* \approx 4$



(c) TIMIT (Gaussian, $\sigma = 11$) $\beta = 1, \lambda_1 = 0.054, m^* \approx 20$

Figure 4: Comparison of training error ($n = 10^4$) for different mini-batch sizes (m) vs. number of iterations

Theorem 4 suggests that for SGD with batch size m^* (by Eq. 14) can reach the same error as GD using at most 4 times the number of iterations. For both MNIST and TIMIT in Fig. 4, SGD with batch size m^* reaches the error of GD using less than 4 times iterations. For HINT-S [HYWW13], it takes less than 8 times iterations to reach the same error. Moreover, in line with our analysis, SGD with the larger batch size $16 \cdot m^*$ (but still much smaller than the data size) converges nearly identical to full gradient descent.



(a) MNIST (Gaussian, $\sigma = 5$) $\beta = 1, \lambda_1 = 0.15, m^* \approx 8$

(b) HINT-S (Gaussian, $\sigma = 16$) $\beta = 1, \lambda_1 = 0.3, m^* \approx 4$

(c) TIMIT (Gaussian, $\sigma = 11$) $\beta = 1, \lambda_1 = 0.054, m^* \approx 20$

Figure 5: Comparison of training error ($n = 10^4$) for different mini-batch sizes (m) vs. number of epochs (proportional to computation, note for n data points, $n \cdot N_{epoch} = m \cdot N_{iter}$)

Note that only the training results are reported here as our analysis applies to them. For completeness we report the test performance in the Appendix C. The test results are consistent with training.

5.3 Optimality of batch size $m = 1$

When different mini-batch sizes are compared, Theorem 6 and Theorem 7 show that $m = 1$ achieves the optimal efficiency for a fixed computational budget. Note for a given batch size, the corresponding optimal step size is chosen according to Eq. 11. The experiments in Fig. 5 show that $m = 1$ indeed achieves the lowest error for any fixed number of epochs.

5.4 Linear scaling and saturation regimes

In the interpolation regime, Theorem 6 show linear scaling for mini-batch sizes up to (typically small) "critical" batch size m^* (Eq. 14) followed by the saturation regime. The linear scaling regime ($1 \le m \le m^*$) is reflected in the small difference in the training error for $m = 1$ and $m = m^*$ in Fig. 5. The top two lines in the figures show saturation.

Relation to the "linear scaling rule" in neural networks. A number of recent large scale neural network methods including [Kri14, CMBJ16, GDG+17] use the "linear scaling rule" to accelerate training using parallel computation. After the initial "warmup" stage to find a good region of parameters, this rule suggest increasing the step size to a level proportional to the mini-batch size m. In spite of the wide adoption and effectiveness of this technique, there has been no satisfactory explanation [Kri14] as the usual variance-based analysis suggests increasing the step size by a factor of $\sqrt{m}$ instead of m [BCN16]. We note that this "linear scaling" can be explained by our analysis, assuming that the warmup stage ends up in a neighborhood of an interpolating minimum.

References

[AZ16] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

[BCN16] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

[BFT17] Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *NIPS*, 2017.

[CCSL16] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, and Yann LeCun. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

[CMBJ16] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[CPC16] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

[GAGN15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *ICML*, 2015.

[GDG$^+$17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[GLF$^+$93] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. *NIST speech disc*, 1-1.1, 1993.

[HHS17] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*, 2017.

[HLWvdM16] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[HYWW13] Eric W Healy, Sarah E Yoho, Yuxuan Wang, and DeLiang Wang. An algorithm to improve speech recognition in noise for hearing-impaired listeners. *The Journal of the Acoustical Society of America*, 134(4), 2013.

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

[Kac37] Stefan Kaczmarz. Angenaherte auflosung von systemen linearer gleichungen. *Bull. Int. Acad. Sci. Pologne, A*, 35, 1937.

[Kri14] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, 1998.

[Li17] Mu Li. *Scaling Distributed Machine Learning with System and Algorithm Co-design*. PhD thesis, 2017.

[LZCS14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *KDD*, 2014.

[MB11] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *NIPS*, 2011.

[MB17] Siyuan Ma and Mikhail Belkin. Diving into the shallows: a computational perspective on large-scale shallow learning. *arXiv preprint arXiv:1703.10622*, 2017.

[NWS14] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *NIPS*, 2014.

[RSB12] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, 2012.

[Sal17] Ruslan Salakhutdinov. Deep learning tutorial at the Simons Institute, Berkeley, https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1, 2017.

[SEG$^+$17] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[SFBL98] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5), 1998.

[SKL17] Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[SV09] Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2), 2009.

[TBRS13] Martin Takác, Avleen Singh Bijral, Peter Richtárik, and Nati Srebro. Mini-batch primal and dual methods for svms. In *ICML*, 2013.

[XZ14] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4), 2014.

[YGG17] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017.

[ZBH$^+$16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A Proof of Theorem 1

The first inequality in the statement of theorem follows directly from β-smoothness of $\mathscr{L}$. Namely, we have

$$\mathscr{L}(w_t) \leq \mathscr{L}(w^*) + \langle \nabla \mathscr{L}(w^*), w_t - w^* \rangle + \frac{\beta}{2} \|w_t - w^*\|^2 = \frac{\beta}{2} \|w_t - w^*\|^2 .$$

Next, we prove the second inequality. To do this, observe that

$$\begin{aligned}\mathop{\mathbb{E}}_{w_t}\left[\|w_t - w^*\|^2\right] &= \mathop{\mathbb{E}}_{w_{t-1},\, i_{t-1}}\left[\langle w_{t-1} - w^* - \frac{1}{\beta}\nabla \ell_{i_{t-1}}(w_{t-1})\ ,\ w_{t-1} - w^* - \frac{1}{\beta}\nabla \ell_{i_{t-1}}(w_{t-1})\rangle\right] \\ &= \mathop{\mathbb{E}}_{w_{t-1},\, i_{t-1}}\left[\|w_{t-1} - w^*\|^2 - \frac{2}{\beta}\langle \nabla \ell_{i_{t-1}}(w_{t-1}), w_{t-1} - w^*\rangle + \frac{1}{\beta^2}\left\|\nabla \ell_{i_{t-1}}(w_{t-1})\right\|^2\right]\end{aligned} \tag{15}$$

Next, we consider the expectation of the middle term in the last expression, and use α- strong convexity of $\mathscr{L}$ to get

$$\begin{aligned}\mathop{\mathbb{E}}_{w_{t-1},\, i_{t-1}}\left[\langle \nabla \ell_{i_{t-1}}(w_{t-1}), w_{t-1} - w^*\rangle\right] &= \mathop{\mathbb{E}}_{w_{t-1}}\left[\mathbb{E}_{i_{t-1}}\left[\langle \nabla \ell_{i_{t-1}}(w_{t-1}), w_{t-1} - w^*\rangle \mid w_{t-1}\right]\right] \\ &= \mathop{\mathbb{E}}_{w_{t-1}}\left[\langle \nabla \mathscr{L}(w_{t-1}), w_{t-1} - w^*\rangle\right] \\ &\geq \mathop{\mathbb{E}}_{w_{t-1}}\left[\mathscr{L}(w_{t-1}) + \frac{\alpha}{2}\|w_{t-1} - w^*\|^2\right]\end{aligned} \tag{16}$$

From (15) and (16), we get

$$\begin{aligned}\mathop{\mathbb{E}}_{w_t}\left[\|w_t - w^*\|^2\right] &\leq (1 - \frac{\alpha}{\beta})\mathop{\mathbb{E}}_{w_{t-1}}\left[\|w_{t-1} - w^*\|^2\right] \\ &\quad - \frac{2}{\beta}\left(\mathop{\mathbb{E}}_{w_{t-1}}\left[\mathscr{L}(w_{t-1})\right] - \frac{1}{2\beta}\mathop{\mathbb{E}}_{w_{t-1},\, i_{t-1}}\left[\left\|\nabla \ell_{i_{t-1}}(w_{t-1})\right\|^2\right]\right) \\ &= (1 - \frac{\alpha}{\beta})\mathop{\mathbb{E}}_{w_{t-1}}\left[\|w_{t-1} - w^*\|^2\right] \\ &\quad - \frac{2}{\beta}\mathop{\mathbb{E}}_{w_{t-1},\, i_{t-1}}\left[\ell_{i_{t-1}}(w_{t-1}) - \frac{1}{2\beta}\left\|\nabla \ell_{i_{t-1}}(w_{t-1})\right\|^2\right]\end{aligned} \tag{17}$$

By β-smoothness of $\ell_{i_{t-1}}$, we have

$$\ell_{i_{t-1}}(w_{t-1}) - \frac{1}{2\beta}\left\|\nabla \ell_{i_{t-1}}(w_{t-1})\right\|^2 \geq \ell_{i_{t-1}}(w_t) \geq 0$$

Hence, the last term in (17) is non-negative. This gives us

$$\mathop{\mathbb{E}}_{w_t}\left[\|w_t - w^*\|^2\right] \leq (1 - \frac{\alpha}{\beta})\mathop{\mathbb{E}}_{w_{t-1}}\left[\|w_{t-1} - w^*\|^2\right]$$

This recurrence gives us the desired bound.

B Proof of Theorem 6

Here, we will provide an exact analysis for the optimality of batch size $m = 1$ for the cost function $g^*(m)^{1/m}$, which, as discussed in Section 4.4.1, captures the total computational cost required to achieve any fixed target accuracy (in a model with no parallel computation).

We prove this theorem by showing that $g^*(m)^{\frac{1}{m}}$ is strictly increasing for $m \geq 1$. We do this via the following two simple lemmas. First, we introduce the following notation.

Let

$$g_1(m) = 1 - \frac{m\lambda_n}{\beta + (m-1)\lambda_n},\ m \geq 1$$

That is, $g_1(m)$ is an extension of $g^*(m),\ m \in [1, \frac{\beta}{\lambda_1 - \lambda_n} + 1]$ (given by the first expression in (10) in Theorem 4) to all $m \geq 1$.

Let $g_2(m)$ denote the extension of $g^*(m),\ m > \frac{\beta}{\lambda_1 - \lambda_n} + 1$ (given by the second expression in (10) in Theorem 4) to all $m \geq 1$. That is,

$$g_2(m) = 1 - 4\frac{m(m-1)\lambda_1\lambda_n}{(\beta + (m-1)(\lambda_1 + \lambda_n))^2},\ m \geq 1.$$

Lemma 3. *$g_1(m)^{\frac{1}{m}}$ is strictly increasing for $m \geq 1$.*

Proof. Define $T(m) \triangleq \frac{1}{m}\ln(1/g_1(m))$. We will show that $T(m)$ is strictly decreasing for $m \geq 1$, which is tantamount to showing that $g_1(m)^{\frac{1}{m}}$ is strictly increasing over $m \geq 1$. For more compact notation, let's define $\tau \triangleq \frac{\beta - \lambda_n}{\beta}$, and $\bar{\tau} = 1 - \tau$. First note that, after straightforward simplification, $g^*(m) = \frac{\tau}{\tau + \bar{\tau}m}$. Hence, $T(m) = \frac{1}{m}\ln(1 + um)$, where $u \triangleq \frac{\bar{\tau}}{\tau} = \frac{\lambda_n}{\beta - \lambda_n} > 0$. Now, it is not hard to see that $T(m)$ is strictly decreasing since the function $\frac{1}{x}\ln(1 + ux)$ is strictly decreasing in x as long as $u > 0$. $\square$

Lemma 4. *$g_1(m) \leq g_2(m)$, for all $m \geq 1$.*

Proof. Proving the lemma is equivalent to proving $\frac{4m(m-1)\lambda_1\lambda_n}{(\beta + (m-1)(\lambda_1 + \lambda_n)^2)} < \frac{m\lambda_n}{\beta + (m-1)\lambda_n}$. After direct manipulation, this is equivalent to showing that

$$(m-1)^2(\lambda_1 - \lambda_n)^2 - 2(m-1)\beta(\lambda_1 - \lambda_n) + \beta^2 \geq 0,$$

which is true for all m since the left-hand side is a complete square: $((m-1)(\lambda_1 - \lambda_n) - \beta)^2$. $\square$

Given these two simple lemmas, observe that

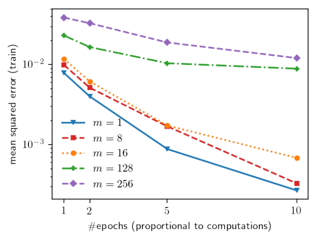
$$g^*(1) = g_1(1) \leq g_1(m) = g^*(m), \qquad \text{for all } m \in [1, \frac{\beta}{\lambda_1 - \lambda_n} + 1], \tag{18}$$

where the first and last equalities follow from the fact that $g_1(m) = g^*(m)$ for $m \in [1, \frac{\beta}{\lambda_1 - \lambda_n} + 1]$, and the second inequality follows from Lemma 3. Also, observe that

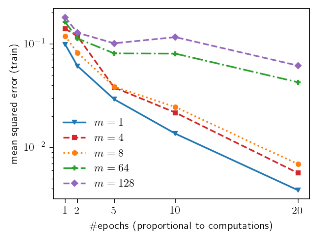
$$g^*(1) = g_1(1) \leq g_1(m) \leq g_2(m) = g^*(m), \qquad \text{for all } m > \frac{\beta}{\lambda_1 - \lambda_n} + 1, \tag{19}$$

where the third inequality follows from Lemma 4, and the last equality follows from the fact that $g_2(m) = g^*(m)$ for $m > \frac{\beta}{\lambda_1 - \lambda_n} + 1$. Putting (18) and (19) together, we have $g^*(1) \leq g^*(m)$, for all $m \geq 1$, which completes the proof.

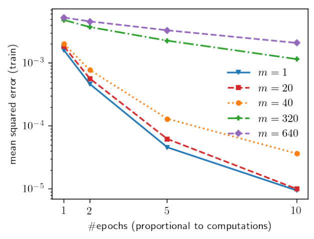
C Experiments: Comparison of Train and Test losses



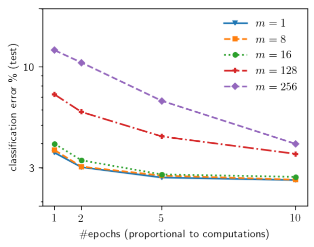
MNIST (train)



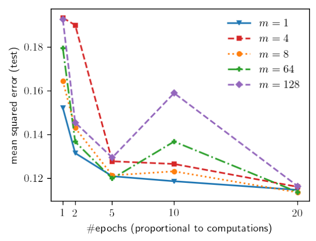
HINT-S (train)



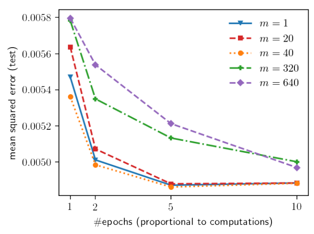
TIMIT (train)



(a) MNIST (test)



(b) HINT-S (test)



(c) TIMIT (test)

Figure 6: Comparison of training error ($n = 10^4$) and testing error for different mini-batch sizes (m) vs. number of epochs (proportional to computation, note for n data points, $n \cdot N_{epoch} = m \cdot N_{iter}$)