Leader Stochastic Gradient Descent for Distributed Training of Deep Learning Models

Yunfei Teng[*,1] yt1208@nyu.edu
Wenbo Gao[*,2] wg2279@columbia.edu
Francois Chalus chalusf3@gmail.com
Anna Choromanska ac5455@nyu.edu
Donald Goldfarb goldfarb@columbia.edu
Adrian Weller aw665@cam.ac.uk

Abstract

We consider distributed optimization under communication constraints for training deep learning models. We propose a new algorithm, whose parameter updates rely on two forces: a regular gradient step, and a corrective direction dictated by the currently best-performing worker (leader). Our method differs from the parameter-averaging scheme EASGD [1] in a number of ways: (i) our objective formulation does not change the location of stationary points compared to the original optimization problem; (ii) we avoid convergence decelerations caused by pulling local workers descending to different local minima to each other (i.e. to the average of their parameters); (iii) our update by design breaks the curse of symmetry (the phenomenon of being trapped in poorly generalizing sub-optimal solutions in symmetric non-convex landscapes); and (iv) our approach is more communication efficient since it broadcasts only parameters of the leader rather than all workers. We provide theoretical analysis of the batch version of the proposed algorithm, which we call Leader Gradient Descent (LGD), and its stochastic variant (LSGD). Finally, we implement an asynchronous version of our algorithm and extend it to the multi-leader setting, where we form groups of workers, each represented by its own local leader (the best performer in a group), and update each worker with a corrective direction comprised of two attractive forces: one to the local, and one to the global leader (the best performer among all workers). The multi-leader setting is well-aligned with current hardware architecture, where local workers forming a group lie within a single computational node and different groups correspond to different nodes. For training convolutional neural networks, we empirically demonstrate that our approach compares favorably to state-of-the-art baselines.

1 Introduction

As deep learning models and data sets grow in size, it becomes increasingly helpful to parallelize their training over a distributed computational environment. These models lie at the core of many modern machine-learning-based systems for image recognition [2], speech recognition [3], natural language processing [4], and more. This paper focuses on the parallelization of the data, not the model, and considers collective communication scheme [5] that is most commonly used nowadays. A typical approach to data parallelization in deep learning [6, 7] uses multiple workers that run variants of SGD [8] on different data batches. Therefore, the effective batch size is increased by the number of workers. Communication ensures that all models are synchronized and critically relies on a scheme where each worker broadcasts its parameter gradients to all the remaining workers.

*,1: Equal contribution. Algorithm development and implementation on deep models.
*,2: Equal contribution. Theoretical analysis and implementation on matrix completion.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

This is the case for DOWNPOUR [9] (its decentralized extension, with no central parameter server, based on the ring topology can be found in [10]) or Horovod [11] methods. These techniques require frequent communication (after processing each batch) to avoid instability/divergence, and hence are communication expensive. Moreover, training with a large batch size usually hurts generalization [12, 13, 14] and convergence speed [15, 16].

Another approach, called Elastic Averaging (Stochastic) Gradient Decent, EA(S)GD [1], introduces elastic forces linking the parameters of the local workers with central parameters computed as a moving average over time and space (i.e. over the parameters computed by local workers). This method allows less frequent communication as workers by design do not need to have the same parameters but are instead periodically pulled towards each other. The objective function of EASGD, however, has stationary points which are not stationary points of the underlying objective function (see Proposition 8 in the Supplement), thus optimizing it may lead to sub-optimal solutions for the original problem. Further, EASGD can be viewed as a parallel extension of the averaging SGD scheme [17] and as such it inherits the downsides of the averaging policy. On non-convex problems, when the iterates are converging to different local minima (that may potentially be globally optimal), the averaging term can drag the iterates in the wrong directions and significantly hurt the convergence speed of both local workers and the master. In symmetric regions of the optimization landscape, the elastic forces related with different workers may cancel each other out causing the master to be permanently stuck in between or at the maximum between different minima, and local workers to be stuck at the local minima or on the slopes above them. This can result in arbitrarily bad generalization error. We refer to this phenomenon as the "curse of symmetry". Landscape symmetries are common in a plethora of non-convex problems [18, 19, 20, 21, 22], including deep learning [23, 24, 25, 26].

This paper revisits the EASGD update and modifies it in a simple, yet powerful way which overcomes the above mentioned shortcomings of the original technique. We propose to replace the elastic force relying on the average of the parameters of local workers by an attractive force linking the local workers and the current best performer among them (leader). Our approach reduces the communication overhead related with broadcasting parameters of all workers to each other, and instead requires broadcasting only the leader parameters. The proposed approach easily adapts to a typical hardware architecture comprising of multiple compute nodes where each node contains a group of workers and local communication, within a node, is significantly faster than communication between the nodes. We propose a multi-leader extension of our approach that adapts well to this hardware architecture and relies on forming groups of workers (one per compute node) which are attracted both to their local and global leader. To reduce the communication overhead, the correction force related with the global leader is applied less frequently than the one related with the local leader.

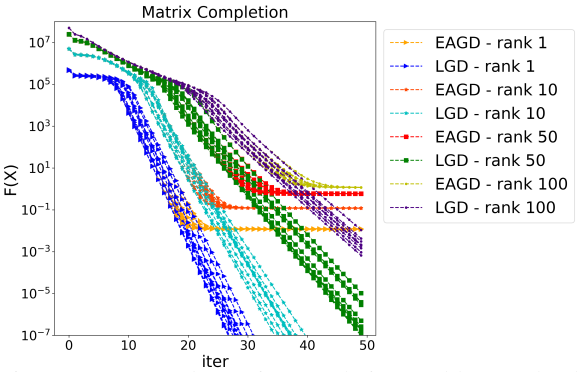

Figure 1: Low-rank matrix completion problems solved with EAGD and LGD. The dimension $d = 1000$ and four ranks $r \in \{1, 10, 50, 100\}$ are used. The reported value for each algorithm is the value of the best worker (8 workers are used in total) at each step.

Finally, our L(S)GD approach, similarly to EA(S)GD, tends to explore wide valleys in the optimization landscape when the pulling force between workers and leaders is set to be small. This property often leads to improved generalization performance of the optimizer [27, 28].

The paper is organized as follows: Section 2 introduces the L(S)GD approach, Section 3 provides theoretical analysis, Section 4 contains empirical evaluation, and finally Section 5 concludes the paper. Theoretical proofs and additional theoretical and empirical results are contained in the Supplement.

2 Leader (Stochastic) Gradient Descent "L(S)GD" Algorithm

2.1 Motivating example

Figure 1 illustrates how elastic averaging can impair convergence. To obtain the figure we applied EAGD (Elastic Averaging Gradient Decent) and LGD to the matrix completion problem of the form: $\min_X \left\{ \frac{1}{4}\|M - XX^T\|_F^2 : X \in \mathbb{R}^{d\times r} \right\}$. This problem is non-convex but is known to have the property that all local minimizers are global minimizers [18]. For four choices of the rank r, we generated 10 random instances of the matrix completion problem, and solved each with EAGD and LGD, initialized from the same starting points (we use 8 workers). For each algorithm, we report the progress of the *best* objective value at each iteration, over all workers. Figure 1 shows the results across 10 random experiments for each rank.

It is clear that EAGD slows down significantly as it approaches a minimizer. Typically, the center $\widetilde{X}$ of EAGD is close to the average of the workers, which is a poor solution for the matrix completion problem when the workers are approaching different local minimizers, even though all local minimizers are globally optimal. This induces a pull on each node *away* from the minimizers, which makes it extremely difficult for EAGD to attain a solution of high accuracy. In comparison, LGD does not have this issue. Further details of this experiment, and other illustrative examples of the difference between EAGD and LGD, can be found in the Supplement.

2.2 Symmetry-breaking updates

Next we explain the basic update of the L(S)GD algorithm. Consider first the single-leader setting and the problem of minimizing loss function L in a parallel computing environment. The optimization problem is given as

$$\min_{x^1,x^2,\ldots,x^l} L(x^1, x^2, \ldots, x^l) \coloneqq \min_{x^1,x^2,\ldots,x^l} \sum_{i=1}^{l} \mathbb{E}[f(x^i;\xi^i)] + \frac{\lambda}{2}||x^i - \tilde{x}||^2, \tag{1}$$

where l is the number of workers, $x^1, x^2, \ldots, x^l$ are the parameters of the workers and $\tilde{x}$ are the parameters of the leader. The best performing worker, i.e. $\tilde{x} = \underset{x^1,x^2,\ldots,x^l}{\arg\min}\ \mathbb{E}[f(x^i;\xi^i)]$), and ξ^is are data samples drawn from some probability distribution $\mathcal{P}$. λ is the hyperparameter that denotes the strength of the force pulling the workers to the leader. In the theoretical section we will refer to $\mathbb{E}[f(x^i;\xi^i)]$ as simply $f(x^i)$. This formulation can be further extended to the multi-leader setting. The optimization problem is modified to the following form

$$\begin{aligned}&\min_{x^{1,1},x^{1,2},\ldots,x^{n,l}} L(x^{1,1}, x^{1,2}, \ldots, x^{n,l})\\ &\coloneqq \min_{x^{1,1},x^{1,2},\ldots,x^{n,l}} \sum_{j=1}^{n}\sum_{i=1}^{l} \mathbb{E}[f(x^{j,i};\xi^{j,i})] + \frac{\lambda}{2}||x^{j,i} - \tilde{x}^j||^2 + \frac{\lambda_G}{2}||x^{j,i} - \tilde{x}||^2,\end{aligned} \tag{2}$$

where n is the number of groups, l is the number of workers in each group, $\tilde{x}^j$ is the local leader of the j^{th} group (i.e. $\tilde{x}^j = \arg\min_{x^{j,1},x^{j,2},\ldots,x^{j,l}} \mathbb{E}[f(x^{j,i};\xi^{j,i})]$), $\tilde{x}$ is the global leader (the best worker among local leaders, i.e. $\tilde{x} = \underset{x^{1,1},x^{1,2},\ldots,x^{n,l}}{\arg\min}\ \mathbb{E}[f(x^{j,i};\xi^{j,i})]$), $x^{j,1}, x^{j,2}, \ldots, x^{j,l}$ are the parameters of the workers in the j^{th} group, and $\xi^{j,i}$s are the data samples drawn from $\mathcal{P}$. λ and λ_G are the hyperparameters that denote the strength of the forces pulling the workers to their local and global leader respectively.

The updates of the LSGD algorithm are captured below, where t denotes iteration. The first update shown in Equation 3 is obtained by taking the gradient descent step on the objective in Equation 2 with respect to variables $x^{j,i}$. The stochastic gradient of $\mathbb{E}[f(x^i;\xi^i)]$ with respect to $x^{j,i}$ is denoted as $g_t^{j,i}$ (in case of LGD the gradient is computed over all training examples) and η is the learning rate.

$$x_{t+1}^{j,i} = x_t^{j,i} - \eta g_t^{j,i}(x_t^{j,i}) - \lambda(x_t^{j,i} - \tilde{x}_t^j) - \lambda_G(x_t^{j,i} - \tilde{x}_t) \tag{3}$$

where $\tilde{x}_{t+1}^j$ and $\tilde{x}_{t+1}$ are the local and global leaders defined above.

Equation 3 describes the update of any given worker and is comprised of the regular gradient step and two corrective forces (in single-leader setting the third term disappears as $\lambda_G = 0$ then). These

Algorithm 1 LSGD Algorithm (Asynchronous)

Input: pulling coefficients λ, λ_G, learning rate η, local/global communication periods τ, τ_G
Initialize:
Randomly initialize $x^{1,1}, x^{1,2}, ..., x^{n,l}$
Set iteration counters $t^{j,i} = 0$
Set $\tilde{x}_0^j = \underset{x^{j,1},...,x^{j,l}}{\arg\min}\ \mathbb{E}[f(x^{j,i}; \xi_0^{j,i})], \tilde{x}_0 = \underset{x^{1,1},...,x^{n,l}}{\arg\min}\ \mathbb{E}[f(x^{j,i}; \xi_0^{j,i})]$;
repeat
  for all $j = 1, 2, \ldots, n$, $i = 1, 2, \ldots, l$ **do** ▷ Do in parallel for each worker
    Draw random sample $\xi_{t^{j,i}}^{j,i}$
    $x^{j,i} \leftarrow x^{j,i} - \eta g_t^{j,i}(x^{j,i})$
    $t^{j,i} = t^{j,i} + 1$;
    if $nl\tau$ divides $(\sum_{j=1}^{n}\sum_{i=1}^{l} t^{j,i})$ **then**
      $\tilde{x}^j = \arg\min_{x^{j,1},...,x^{j,l}} \mathbb{E}[f(x^{j,i}; \xi_{t^{j,i}}^{j,i})]$. ▷ Determine the local best workers
      $x^{j,i} \leftarrow x^{j,i} - \lambda(x^{j,i} - \tilde{x}^j)$ ▷ Pull to the local best workers
    end if
    if $nl\tau_G$ divides $(\sum_{j=1}^{n}\sum_{i=1}^{l} t^{j,i})$ **then**
      $\tilde{x} = \arg\min_{x^{1,1},...,x^{n,l}} \mathbb{E}[f(x^{j,i}; \xi_{t^{j,i}}^{j,i})]$. ▷ Determine the global best worker
      $x^{j,i} \leftarrow x^{j,i} - \lambda_G(x^{j,i} - \tilde{x})$ ▷ Pull to the global best worker
    end if
  end for
until termination

forces constitute the communication mechanism among the workers and pull all the workers towards the currently best local and global solution to ensure fast convergence. As opposed to EASGD, the updates performed by workers in LSGD break the curse of symmetry and avoid convergence decelerations that result from workers being pulled towards the average which is inherently influenced by poorly performing workers. In this paper, instead of pulling workers to their averaged parameters, we propose the mechanism of pulling the workers towards the leaders. The flavor of the update resembles a particle swarm optimization approach [29], which is not typically used in the context of stochastic gradient optimization for deep learning. Our method may therefore be viewed as a dedicated particle swarm optimization approach for training deep learning models in the stochastic setting and parallel computing environment.

Next we describe the LSGD algorithm in more detail. We rely on the collective communication scheme. In order to reduce the amount of communication between the workers, it is desired to pull them towards the leaders less often than every iteration. Also, in practice each worker can have a different speed. To prevent waiting for the slower workers and achieve communication efficiency, we implement the algorithm in the asynchronous operation mode. In this case, the communication period is determined based on the total number of iterations computed across all workers and the communication is performed every $nl\tau$ or $nl\tau_G$ iterations, where τ and τ_G denote local and global communication periods, respectively. In practice, we use $\tau_G > \tau$ since communication between workers lying in different groups is more expensive than between workers within one group, as explained above. When communication occurs, all workers are updated at the same time (i.e. pulled towards the leaders) in order to take advantage of the collective communication scheme. Between communications, workers run their own local SGD optimizers. The resulting LSGD method is very simple, and is depicted in Algorithm 1.

The next section provides a theoretical description of the single-leader batch (LGD) and stochastic (LSGD) variants of our approach.

3 Theoretical Analysis

We assume without loss of generality that there is a single leader. The objective function with multiple leaders is given by $f(x)+\frac{\lambda_1}{2}\|x-z_1\|^2+\ldots+\frac{\lambda_c}{2}\|x-z_c\|^2$, which is equivalent to $f(x)+\frac{\Lambda}{2}\|x-\widetilde{z}\|^2$ for $\Lambda=\sum_{i=1}^c\lambda_i$ and $\widetilde{z}=\frac{1}{\Lambda}\sum_{i=1}^c\lambda_i z_i$. Proofs for this section are deferred to the Supplement.

3.1 Convergence Rates for Stochastic Strongly Convex Optimization

We first show that LSGD obtains the same convergence rate as SGD for stochastic strongly convex problems [30]. In Section 3.3 we discuss how and when LGD can obtain *better* search directions than gradient descent. We discuss non-convex optimization in Section 3.2. Throughout Section 3.1, f will typically satisfy:

Assumption 1 f is M-Lipschitz-differentiable and m-strongly convex, which is to say, the gradient ∇f satisfies $\|\nabla f(x)-\nabla f(y)\|\leq M\|x-y\|$, and f satisfies $f(y)\geq f(x)+\nabla f(x)^T(y-x)+\frac{m}{2}\|y-x\|^2$. We write x^* for the unique minimizer of f, and $\kappa:=\frac{M}{m}$ for the condition number of f.

3.1.1 Convergence Rates

The key technical result is that LSGD satisfies a similar one-step descent in expectation as SGD, with an additional term corresponding to the pull of the leader. To provide a unified analysis of 'pure' LSGD as well as more practical variants where the leader is updated infrequently or with errors, we consider a general iteration $x_+=x-\eta(\widetilde{g}(x)+\lambda(x-z))$, where z is an arbitrary guiding point; that is, z may not be the minimizer of $x^1,\ldots,x^p$, nor even satisfy $f(z)\leq f(x^i)$. Since the nodes operate independently except when updating z, we may analyze LSGD steps for each node individually, and we write $x=x^i$ for brevity.

Theorem 1. *Let f satisfy Assumption 1. Let $\widetilde{g}(x)$ be an unbiased estimator for $\nabla f(x)$ with $\mathrm{Var}(\widetilde{g}(x))\leq\sigma^2+\nu\|\nabla f(x)\|^2$, and let z be any point. Suppose that η,λ satisfy $\eta\leq(2M(\nu+1))^{-1}$ and $\eta\lambda\leq(2\kappa)^{-1}, \eta\sqrt{\lambda}\leq(\kappa\sqrt{2m})^{-1}$. Then the LSGD step satisfies*

$$\mathbb{E}f(x_+)-f(x^*)\leq(1-m\eta)(f(x)-f(x^*))-\eta\lambda(f(x)-f(z))+\frac{\eta^2M}{2}\sigma^2. \tag{4}$$

Note the presence of the new term $-\eta\lambda(f(x)-f(z))$ which speeds up convergence when $f(z)\leq f(x)$, i.e the leader is better than x. If the leader z_k is always chosen so that $f(z_k)\leq f(x_k)$ at every step k, then $\limsup_{k\to\infty}\mathbb{E}f(x_k)-f(x^*)\leq\frac{1}{2}\eta\kappa\sigma^2$. If η decreases at the rate $\eta_k=\Theta(\frac{1}{k})$, then $\mathbb{E}f(x_k)-f(x^*)\leq O(\frac{1}{k})$.

The $O(\frac{1}{k})$ rate of LSGD matches that of comparable distributed methods. Both Hogwild [31] and EASGD achieve a rate of $O(\frac{1}{k})$ on strongly convex objective functions. We note that published convergence rates are not available for many distributed algorithms (including DOWNPOUR [9]).

3.1.2 Communication Periods

In practice, communication between distributed machines is costly. The LSGD algorithm has a *communication period* τ for which the leader is only updated every τ iterations, so each node can run independently during that period. This τ is allowed to differ between nodes, and over time, which captures the asynchronous and multi-leader variants of LSGD. We write $x_{k,j}$ for the j-th step during the k-th period. It may occur that $f(z)>f(x_{k,j})$ for some k,j, that is, the current solution $x_{k,j}$ is now better than the last selected leader. In this case, the leader term $\lambda(x-z)$ may no longer be beneficial, and instead simply pulls x toward z. There is no general way to determine how many steps are taken before this event. However, we can show that if $f(z)\geq f(x)$, then

$$\mathbb{E}f(x_+)\leq f(z)+\frac{1}{2}\eta^2M\sigma^2, \tag{5}$$

so the solution will not become *worse* than a stale leader (up to gradient noise). As τ goes to infinity, LSGD converges to the minimizer of $\psi(x)=f(x)+\frac{\lambda}{2}\|x-z\|^2$, which is quantifiably better than z as captured in Theorem 2. Together, these facts show that LSGD is safe to use with long communication periods as long as the original leader is good.

Theorem 2. *Let f be m-strongly convex, and let x^* be the minimizer of f. For fixed λ, z, define $\psi(x) = f(x) + \frac{\lambda}{2}\|x - z\|^2$. The minimizer w of ψ satisfies $f(w) - f(x^*) \leq \frac{\lambda}{m+\lambda}(f(z) - f(x^*))$.*

The theoretical results here and in Section 3.1.1 address two fundamental instances of the LSGD algorithm: the 'synchronous' case where communication occurs each round, and the 'infinitely asynchronous' case where communication periods are arbitrarily long. For unknown periods $\tau > 1$, it is difficult to demonstrate general quantifiable improvements beyond (5), but we note that (4), Theorem 2, and the results on stochastic leader selection (Sections 3.1.3 and 7.6) can be combined to analyze specific instances of the asynchronous LSGD.

In our experiments, we employ another method to avoid the issue of stale leaders. To ensure that the leader is good, we perform an LSGD step only on the first step after a leader update, and then take standard SGD steps for the remainder of the communication period.

3.1.3 Stochastic Leader Selection

Next, we consider the impact of selecting the leader with errors. In practice, it is often costly to evaluate $f(x)$, as in deep learning. Instead, we estimate the values $f(x^i)$, and then select z as the variable having the smallest estimate. Formally, suppose that we have an unbiased estimator $\widetilde{f}(x)$ of $f(x)$, with uniformly bounded variance. At each step, a single sample $y_1, \ldots, y_p$ is drawn from each estimator $\widetilde{f}(x^1), \ldots, \widetilde{f}(x^p)$, and then $z = \{x^i : y_i = \min\{y_1, \ldots, y_p\}\}$. We refer to this as *stochastic leader selection*. The stochastic leader satisfies $\mathbb{E}f(z) \leq f(z_{true}) + 4\sqrt{p}\sigma_f$, where z_{true} is the true leader (see supplementary materials). Thus, the error introduced by the stochastic leader contributes an additive error of at most $4\eta\lambda\sqrt{p}\sigma_f$. Since this is of order η rather than η^2, we cannot guarantee convergence with $\eta_k = \Theta(\frac{1}{k})$[1] unless λ_k is also decreasing. We have the following result:

Theorem 3. *Let f satisfy Assumption 1, and let $\widetilde{g}(x)$ be as in Theorem 1. Suppose we use stochastic leader selection with $\widetilde{f}(x)$ having $\mathrm{Var}(\widetilde{f}(x)) \leq \sigma_f^2$. If η, λ are fixed so that $\eta \leq (2M(\nu+1))^{-1}$ and $\eta\lambda \leq (2\kappa)^{-1}, \eta\sqrt{\lambda} \leq (\kappa\sqrt{2m})^{-1}$, then $\limsup_{k\to\infty} \mathbb{E}f(x_k) - f(x^*) \leq \frac{1}{2}\eta\kappa\sigma^2 + \frac{4}{m}\lambda\sqrt{p}\sigma_f$. If η, λ decrease at the rate $\eta_k = \Theta(\frac{1}{k}), \lambda_k = \Theta(\frac{1}{k})$, then $\mathbb{E}f(x_k) - f(x^*) \leq O(\frac{1}{k})$.*

The communication period and the accuracy of stochastic leader selection are both methods of reducing the cost of updating the leader, and can be substitutes. When the communication period is long, it may be effective to estimate $f(x^i)$ to higher accuracy, since this can be done independently.

3.2 Non-convex Optimization: Stationary Points

As mentioned above, EASGD has the flaw that the EASGD objective function can have stationary points such that none of $x^1, \ldots, x^p, \widetilde{x}$ is a stationary point of the underlying function f. LSGD does not have this issue.

Theorem 4. *Let Ω_i be the points $(x^1, \ldots, x^p)$ where x^i is the unique minimizer among $(x^1, \ldots, x^p)$. If $x^* = (w^1, \ldots, w^p) \in \Omega_i$ is a stationary point of the LSGD objective function, then $\nabla f^i(w^i) = 0$.*

Moreover, it can be shown that for the deterministic algorithm LGD with *any choice of communication periods*, there will always be some variable x^i such that $\liminf \|\nabla f(x_k^i)\| = 0$.

Theorem 5. *Assume that f is bounded below and M-Lipschitz-differentiable, and that the LGD step sizes are selected so that $\eta_i < \frac{2}{M}$. Then for any choice of communication periods, it holds that for every i such that x^i is the leader infinitely often, $\liminf_k \|\nabla f(x_k^i)\| = 0$.*

3.3 Search Direction Improvement from Leader Selection

In this section, we discuss how LGD can obtain better search directions than gradient descent. In general, it is difficult to determine when the LGD step will satisfy $f(x - \eta(\nabla f(x) + \lambda(x - z))) \leq f(x - \eta\nabla f(x))$, since this depends on the precise combination of f, x, z, η, λ, and moreover, the maximum allowable value of η is different for LGD and gradient descent. Instead, we measure the goodness of a search direction by the angle it forms with the Newton direction $d_N(x) = -(\nabla^2 f(x))^{-1}\nabla f(x)$. The Newton method is locally quadratically convergent around local minimizers with non-singular

[1] For intuition, note that $\sum_{n=1}^{\infty} \frac{1}{n}$ is divergent.

Hessian, and converges in a single step for quadratic functions if $\eta = 1$. Hence, we consider it desirable to have search directions that are close to d_N. Let $\theta(u, v)$ denote the angle between u, v. Let $d_z = -(\nabla f(x) + \lambda(x - z))$ be the LGD direction with leader z, and $d_G(x) = -\nabla f(x)$. The *angle improvement set* is the set of leaders $I_\theta(x, \lambda) = \{z : f(z) \leq f(x), \theta(d_z, d_N(x)) \leq \theta(d_G(x), d_N(x))\}$. The set of candidate leaders is $E = \{z : f(z) \leq f(x)\}$. We aim to show that a large subset of leaders in E belong to $I_\theta(x, \lambda)$.

In this section, we consider the positive definite quadratic $f(x) = \frac{1}{2}x^T Ax$ with condition number κ and $d_G(x) = -Ax, d_N(x) = -x$. The first result shows that as λ becomes sufficiently small, at least half of E improves the angle. We use the n-dimensional volume $\mathrm{Vol}(\cdot)$ to measure the relative size of sets: an ellipsoid E given by $E = \{x : x^T Ax \leq 1\}$ has volume $\mathrm{Vol}(E) = \det(A)^{-1/2} \mathrm{Vol}(S_n)$, where S_n is the unit ball.

Theorem 6. *Let x be any point such that $\theta_x = \theta(d_G(x), d_N(x)) > 0$, and let $E = \{z : f(z) \leq f(x)\}$. Then* $\lim_{\lambda \to 0} \mathrm{Vol}(I_\theta(x, \lambda)) \geq \frac{1}{2} \mathrm{Vol}(E)$[2].

Next, we consider when λ is large. We show that points with large angle between $d_G(x), d_N(x)$ exist, which are most suitable for improvement by LGD. For $r \geq 2$, define $S_r = \{x : \cos(\theta(d_G(x), d_N(x))) = \frac{r}{\sqrt{\kappa}}\}$. It can be shown that S_r is nonempty for all $r \geq 2$. We show that for $x \in S_r$ for a certain range of r, $I_\theta(x, \lambda)$ is at least half of E *for any choice of* λ.

Theorem 7. *Let $R_\kappa = \{r : \frac{r}{\sqrt{\kappa}} + \frac{r^{3/2}}{\kappa^{1/4}} \leq 1\}$. If $x \in S_r$ for $r \in R_\kappa$, then for any $\lambda \geq 0$,* $\mathrm{Vol}(I_\theta(x, \lambda)) \geq \frac{1}{2} \mathrm{Vol}(E)$.

Note that Theorems 6 and 7 apply only to *convex* functions, or in the neighborhoods of local minimizers where the objective function is locally convex. In nonconvex landscapes, the Newton direction may point towards saddle points [32], which is undesirable; however, since Theorems 6 and 7 do not apply in this situation, these results do not imply that LSGD has harmful behavior. For nonconvex problems, our intuition is that many candidate leaders lie in directions of *negative curvature*, which would actually lead away from saddle points, but this is significantly harder to analyze since the set of candidates is unbounded a priori.

4 Experimental Results

4.1 Experimental setup

In this section we compare the performance of LSGD with state-of-the-art methods for parallel training of deep networks, such as EASGD and DOWNPOUR (their pseudo-codes can be found in [1]), as well as sequential technique SGD. The codes for LSGD can be found at `https://github.com/yunfei-teng/LSGD`. We use communication period equal to 1 for DOWNPOUR in all our experiments as this is the typical setting used for this method ensuring stable convergence. The experiments were performed using the CIFAR-10 data set [33] on three benchmark architectures: 7-layer CNN used in the original EASGD paper (see Section 5.1. in [1]) that we refer to as CNN7, VGG16 [34], and ResNet20 [35]; and ImageNet (ILSVRC 2012) data set [36] on ResNet50.

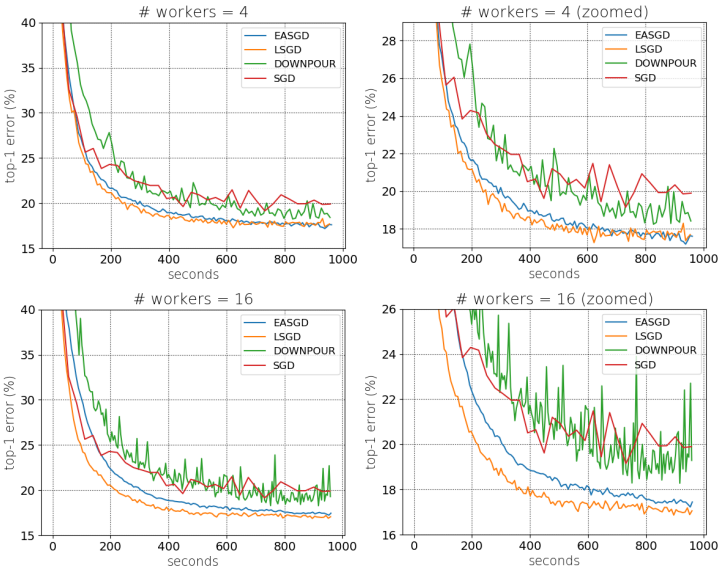

Figure 2: CNN7 on CIFAR-10. Test error for the center variable versus wall-clock time (original plot on the left and zoomed on the right). Test loss is reported in Figure 10 in the Supplement.

[2]Note that $I_\theta(x, \lambda_1) \supseteq I_\theta(x, \lambda_2)$ for $\lambda_1 \leq \lambda_2$, so the limit is well-defined.

During training, we select the leader for the LSGD method based on the average of the training loss computed over the last 10 (CIFAR-10) and 64 (ImageNet) data batches. At testing, we report the performance of the center variable for EASGD and LSGD, where for LSGD the center variable is computed as the average of the parameters of all workers. [*Remark*: Note that we use the leader's parameter to pull to at training and we report the averaged parameters at testing deliberately. It is demonstrated in our paper (e.g.: Figure 1) that pulling workers to the averaged parameters at training may slow down convergence and we address this problem. Note that after training, the parameters that workers obtained after convergence will likely lie in the same valley of the landscape (see [37]) and thus their average is expected to have better generalization ability (e.g. [27, 38]), which is why we report the results for averaged parameters at testing.] Finally, for all methods we use weight decay with decay coefficient set to 10^{-4}. In our experiments we use either 4 workers (single-leader LSGD setting) or 16 workers (multi-leader LSGD setting with 4 groups of workers). For all methods, we report the learning rate leading to the smallest achievable test error under similar convergence rates (we rejected small learning rates which led to unreasonably slow convergence).

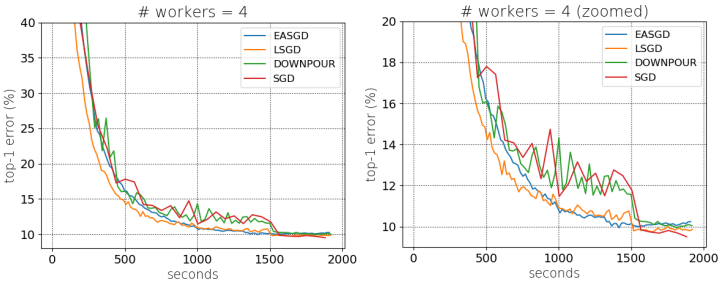

Figure 3: VGG16 on CIFAR-10. Test error for the center variable versus wall-clock time (original plot on the left and zoomed on the right). Test loss is reported in Figure 12 in the Supplement.

We use GPU nodes interconnected with Ethernet. Each GPU node has four GTX 1080 GPU processors where each local worker corresponds to one GPU processor. We use CUDA Toolkit 10.0[3] and NCCL 2[4]. We have developed a software package based on PyTorch for distributed training, which will be released (details are elaborated in Section 9.4).

Data processing and prefetching are discussed in the Supplement. The summary of the hyperparameters explored for each method are also provided in the Supplement. We use constant learning rate for CNN7 and learning rate drop (we divide the learning rate by 10 when we observe saturation of the optimizer) for VGG16, ResNet20, and ResNet50.

4.2 Experimental Results

In Figure 2 we report results obtained with CNN7 on CIFAR-10. We run EASGD and LSGD with communication period $\tau = 64$. We used $\tau_G = 128$ for the multi-leader LSGD case. The number of workers was set to $l = \{4, 16\}$. Our method consistently outperforms the competitors in terms of convergence speed (it is roughly 1.5 times faster than EASGD for 16 workers) and for 16 workers it obtains smaller error.

In Figure 3 we demonstrate results for VGG16 and CIFAR-10 with communication period 64 and number of workers equal to 4. LSGD converges marginally faster than EASGD and recovers

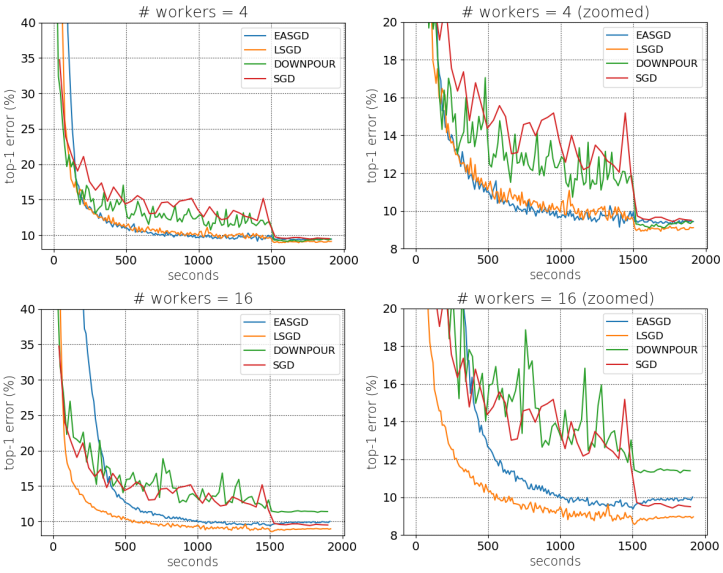

Figure 4: ResNet20 on CIFAR-10. Test error for the center variable versus wall-clock time (original plot on the left and zoomed on the right). Test loss is reported in Figure 11 in the Supplement.

[3]https://developer.nvidia.com/cuda-zone
[4]https://developer.nvidia.com/nccl

the same error. At the same time it outperforms significantly DOWNPOUR in terms of convergence speed and obtains a slightly better solution.

The experimental results obtained using ResNet20 and CIFAR-10 for the same setting of communication period and number of workers as in case of CNN7 are shown in Figure 4. On 4 workers we converge comparably fast to EASGD but recover better test error. For this experiment in Figure 5 we show the switching pattern between the leaders indicating that LSGD indeed takes advantage of all workers when exploring the landscape. On 16 workers we converge roughly 2 times faster than EASGD and obtain significantly smaller error. In this and CNN7 experiment LSGD (as well as EASGD) are consistently better than DONWPOUR and SGD, as expected.

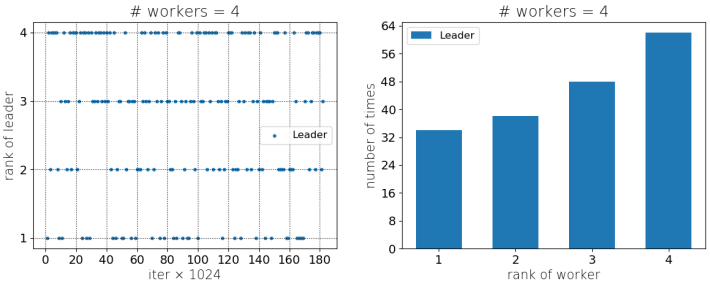

Figure 5: ResNet20 on CIFAR-10. The identity of the worker that is recognized as the leader (i.e. rank) versus iterations (on the left) and the number of times each worker was the leader (on the right).

Remark 1. *We believe that these two facts together — (1) the schedule of leader switching recorded in the experiments shows frequent switching, and (2) the leader point itself is not pulled away from minima — suggest that the 'pulling away' in LSGD is beneficial: non-leader workers that were pulled away from local minima later became the leader, and thus likely obtained an even better solution than they originally would have.*

Finally, in Figure 6 we report the empirical results for ResNet50 run on ImageNet. The number of workers was set to 4 and the communication period τ was set to 64. In this experiment our algorithm behaves comparably to EASGD but converges much faster than DOWNPOUR. Also note that for ResNet50 on ImageNet, SGD is consistently worse than all reported methods (training on ImageNet with SGD on a single GTX1080 GPU until convergence usually takes about a week and gives slightly worse final performance), which is why the SGD curve was deliberately omitted (other methods converge in around two days).

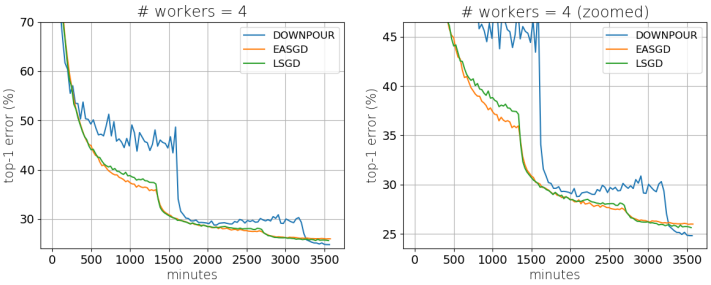

Figure 6: ResNet50 on ImageNet. Test error for the center variable versus wall-clock time (original plot on the left and zoomed on the right). Test loss is reported in Figure 13 in the Supplement.

5 Conclusion

In this paper we propose a new algorithm called LSGD for distributed optimization in non-convex settings. Our approach relies on pulling workers to the current best performer among them, rather than their average, at each iteration. We justify replacing the average by the leader both theoretically and through empirical demonstrations. We provide a thorough theoretical analysis, including proof of convergence, of our algorithm. Finally, we apply our approach to the matrix completion problem and training deep learning models and demonstrate that it is well-suited to these learning settings.

Acknowledgements

WG and DG were supported in part by NSF Grant CCF-1838061. AW acknowledges support from the David MacKay Newton research fellowship at Darwin College, The Alan Turing Institute under EPSRC grant EP/N510129/1 & TU/B/000074, and the Leverhulme Trust via the CFI.

References

[1] S. Zhang, A. Choromanska, and Y. LeCun. Deep learning with elastic averaging SGD. In *NIPS*, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *ICASSP*, 2012.

[4] J. Weston, S. Chopra, and K. Adams. #tagspace: Semantic embeddings from hashtags. In *EMNLP*, 2014.

[5] U. Wickramasinghe and A. Lumsdaine. A survey of methods for collective communication optimization and tuning. *CoRR*, abs/1611.06334, 2016.

[6] T. Ben-Nun and T. Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *CoRR*, abs/1802.09941, 2018.

[7] A. Gholami, A. Azad, P. Jin, K. Keutzer, and A. Buluc. Integrated model, batch, and domain parallelism in training neural networks. *Proceedings of the 30th Syposium on Parallelism in Algorithms and Architectures*, pages 77–86, 2018.

[8] L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *NIPS*, 2012.

[10] X. Lian, W. Zhang, C. Zhang, and J. Liu. Asynchronous decentralized parallel stochastic gradient descent. In *ICML*, 2018.

[11] A. Sergeev and M. Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *CoRR*, abs/1802.05799, 2018.

[12] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[13] S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Finding flatter minima with sgd. In *ICLR Workshop Track*, 2018.

[14] S. L. Smith and Q. V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *ICLR*, 2018.

[15] S. Ma, R. Bassily, and M. Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *ICML*, 2018.

[16] Y. You, I. Gitman, and B. Ginsburg. Scaling SGD batch size to 32k for imagenet training. In *ICLR*, 2018.

[17] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[18] X. Li, J. Lu, R. Arora, J. Haupt, H. Liu, Z. Wang, and T. Zhao. Symmetry, saddle points, and global optimization landscape of nonconvex matrix factorization. *IEEE Transactions on Information Theory*, PP:1–1, 03 2019.

[19] R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *ICML*, 2017.

[20] J. Sun, Q. Qu, and J. Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[21] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere I: overview and the geometric picture. *IEEE Trans. Information Theory*, 63(2):853–884, 2017.

[22] R. Ge, J. D. Lee, and T. Ma. Matrix completion has no spurious local minimum. In *NIPS*, 2016.

[23] V. Badrinarayanan, B. Mishra, and R. Cipolla. Understanding symmetries in deep networks. *CoRR*, abs/1511.01029, 2015.

[24] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

[25] S. Liang, R. Sun, Y. Li, and R. Srikant. Understanding the loss surface of neural networks for binary classification. In *ICML*, 2018.

[26] K. Kawaguchi. Deep learning without poor local minima. In *NIPS*, 2016.

[27] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. T. Chayes, L. Sagun, and R. Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. In *ICLR*, 2017.

[28] P. Chaudhari, C. Baldassi, R. Zecchina, S. Soatto, and A. Talwalkar. Parle: parallelizing stochastic gradient descent. In *SysML*, 2018.

[29] J. Kennedy and R. Eberhart. Particle swarm optimization. In *ICNN*, 1995.

[30] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[31] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

[32] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc., 2014.

[33] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[37] Baldassi, C. et al. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. In *PNAS*, 2016.

[38] Izmailov, P. et al. Averaging weights leads to wider optima and better generalization. *arXiv:1803.05407*, 2018.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Leader Stochastic Gradient Descent for Distributed Training of Deep Learning Models (Supplementary Material)

Abstract

This Supplement presents additional details in support of the full article. These include the proofs of the theoretical statements from the main body of the paper and additional theoretical results. We also provide a toy illustrative example of the difference between LSGD and EASGD. Finally, the Supplement contains detailed description of the experimental setup and additional experiments and figures to provide further empirical support for the proposed methodology.

6 LGD versus EAGD: Illustrative Example

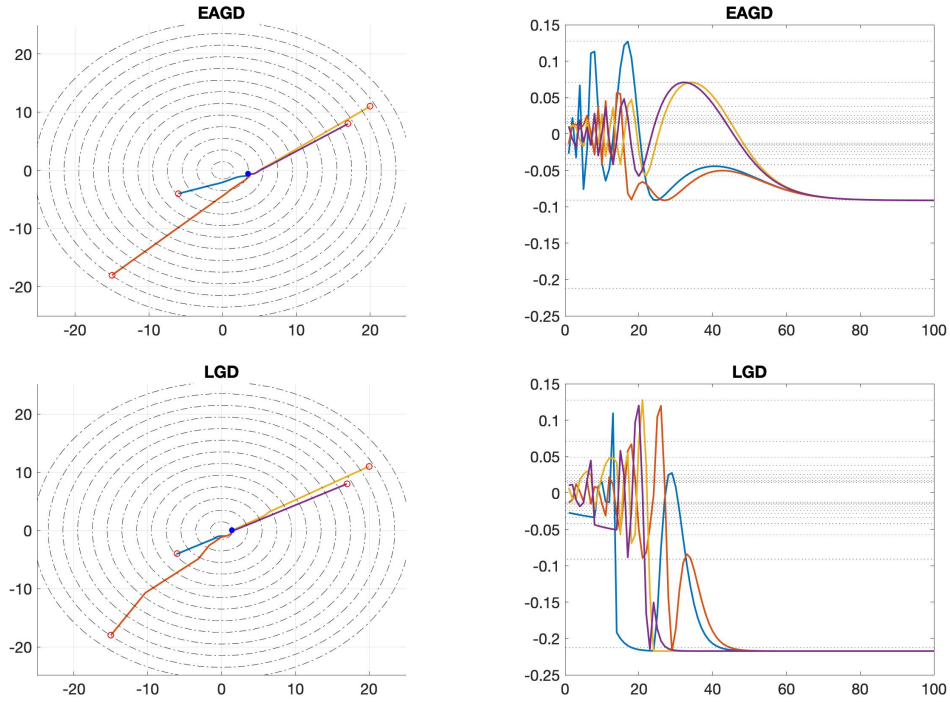

Figure 7: **Left**: Trajectories of variables $(\mathbf{x}, \mathbf{y})$ during optimization. The dashed lines represent the local minima. The red and blue circles are the start and end points of each trajectory, respectively. **Right**: The value of the objective function $L(x, y)$ for each worker during training.

We consider the following non-convex optimization problem:

$$\min_{x,y} L(x, y), \quad \text{where} \quad L(x, y) = \frac{\sin(\sqrt{x^2 + y^2} \cdot \pi)}{\sqrt{x^2 + y^2} \cdot \pi}.$$

Both methods use 4 workers with initial points $(-6,-4)$, $(-15,-18)$, $(20,11)$ and $(17,8)$. The communication period is set to 1. The learning rate for both EAGD and LGD equals 0.1. Furthermore, EAGD uses $\beta = 0.43$ and LGD uses $\lambda = 0.1$.

Table 1 captures optima obtained by different methods.

Optimizer	$L(x,y)$
EAGD	-0.0912
LGD	**-0.2172**

Table 1: Optimum $L(x^*, y^*)$ recovered by EAGD and LGD.

Figure 7 captures the optimization trajectories of EAGD and LGD algorithms. Clearly, EAGD suffers from the averaging policy, whereas LGD is able to recover a solution close to the *global optimum*.

7 Proofs of Theoretical Results

We provide omitted proofs from the main text.

7.1 Definitions and Notation

Recall that the objective function of Leader (Stochastic) Gradient Descent (L(S)GD) is defined as

$$\min_{x^1,\dots,x^p} L(x^1,\dots,x^p) := \sum_{i=1}^{p} f(x^i) + \frac{\lambda}{2}\|x^i - \widetilde{x}\|^2 \tag{6}$$

where $\widetilde{x} = \arg\min\{f(x^1),\dots,f(x^p)\}$. An L(S)GD step is a (stochastic) gradient step applied to L. Writing $z = \widetilde{x}$ at a particular $(x^1,\dots,x^n)$, the update in the variable x^i is

$$x^i_+ = x^i - \eta(\nabla f(x^i) + \lambda(x^i - z))$$

Observe that this reduces to a (S)GD step for the variable which is the leader.

Practical variants of the algorithm do not communicate the updated leader at every iteration. Thus, in our analysis, we will generally take z to be an arbitrary guiding point, which is not necessarily the minimizer of $x^1,\dots,x^p$, nor even satisfy $f(z) \leq f(x^i)$ for all i. The required properties of z will be specified on a result-by-result basis.

When discussing the optimization landscape of LSGD, the term 'LSGD objective function' will refer to (6) with $\widetilde{x}$ defined as the argmin.

Communication periods are sequences of steps where the leader is not updated. We introduce the notation $x_{k,j}$ for the j-th step in the k-th period, where the leader z is updated only at the beginning of each period. We write $b_i(k)$ for the number of steps that x^i takes during the k-th period. The standard LSGD defined above has $b_i(k) = 1$ for all i,k, in which case $x^i_{k,1} = x^i_k$. In addition, let $\widetilde{x}_k = \mathrm{argmin}\{f(x^1_{k,1}),\dots,f(x^p_{k,1})\}$, the leader for the k-th period.

7.2 Stationary Points of EASGD

The EASGD [1] objective function is defined as

$$\min_{x^1,\dots,x^p,\widetilde{x}} L(x^1,\dots,x^p,\widetilde{x}) := \sum_{i=1}^{p} f(x^i) + \frac{\lambda}{2}\|x^i - \widetilde{x}\|^2. \tag{7}$$

Observe that unlike LSGD, $\widetilde{x}$ is a decision variable of EASGD. A stationary point of EASGD is a point such that $\nabla L(x^1,\dots,x^p,\widetilde{x}) = 0$.

Proposition 8. *There exists a Lipschitz differentiable function $f : \mathbb{R} \to \mathbb{R}$ such that for every $0 < \lambda \leq 1$, there exists a point $(x_\lambda, y_\lambda, 0)$ which is a stationary point of EASGD with parameter λ, but none of $\{x_\lambda, y_\lambda, 0\}$ is a stationary point of f.*

Proof. Define $f(x)$ by

$$f(x) = \begin{cases} e^{x+1} & \text{if } x < -1 \\ p(x) & \text{if } -1 \le x \le 1 \\ e^{-x+1} & \text{if } x > 1 \end{cases}$$

where $p(x) = a_6x^6 + \ldots + a_1x + a_0$ is a sixth-degree polynomial. For f to be Lipschitz differentiable, we will select $p(x)$ to make f twice continuously differentiable, with bounded second derivative. To make f twice continuously differentiable, we must have $p(1) = 1, p'(1) = -1, p''(1) = 1$ and $p(-1) = -1, p'(-1) = 1, p''(-1) = -1$. Since we aim to have $f'(0) \neq 0$, we also will require $f'(0) = p'(0) = 1$. The existence of p is equivalent to the solvability of a linear system, which is easily checked to be invertible. Thus, we deduce that such a function f exists.

It remains to show that for any $0 < \lambda \le 1$, there exists a stationary point $(x, y, 0)$ of EASGD. Set $x = -y$. The first-order condition yields $f'(x) + \lambda x = 0$. Since $\lambda \le 1$, we have $\lambda(1) + f'(1) \le 0$. For $x \ge 1$, $f'(x) = -e^{-x+1}$ is an increasing function, so $f'(x) + \lambda x$ is increasing, and we deduce that there exists a solution $y_\lambda \ge 1$ with $\lambda y_\lambda + f'(y_\lambda) = 0$. By symmetry, $-y_\lambda \le -1$ satisfies $f'(-y_\lambda) + \lambda(-y_\lambda) = 0$, since $f'(x) = e^{x+1}$ for $x \le -1$. Hence, $(-y_\lambda, y_\lambda, 0)$ is a stationary point of EASGD, but none of $\{-y_\lambda, y_\lambda, 0\}$ are stationary points of f. □

7.3 Technical Preliminaries

Recall the statement of Assumption 1:

Assumption 1 f is M-Lipschitz-differentiable and m-strongly convex, which is to say, the gradient ∇f satisfies $\|\nabla f(x) - \nabla f(y)\| \le M\|x - y\|$, and f satisfies

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

We write x^* for the unique minimizer of f, and $\kappa := \frac{M}{m}$ for the condition number of f.

We will frequently use the following standard result.

Lemma 9. *If f is M-Lipschitz-differentiable, then*

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

Proof. See [30, eq. (4.3)]. □

Lemma 10. *Let f be m-strongly convex, and let x^* be the minimizer of f. Then*

$$f(w) - f(x^*) \le \frac{1}{2m}\|\nabla f(w)\|^2 \tag{8}$$

and

$$f(w) - f(x^*) \ge \frac{m}{2}\|w - x^*\|^2 \tag{9}$$

Proof. Equation (8) is the well-known Polyak-Łojasiewicz inequality. Equation (9) follows from the definition of strong convexity, and $\nabla f(x^*) = 0$. □

Lemma 11. *Let f be M-Lipschitz-differentiable. If the gradient descent step size $\eta < \frac{2}{M}$, then $\|\nabla f(x)\|^2 \le \alpha(f(x) - f(x^+))$, where $\alpha = \frac{2}{\eta(2-\eta M)}$.*

Proof. By Theorem 9,

$$\begin{aligned} f(x_+) &\le f(x) - \eta\|\nabla f(x)\|^2 + \frac{\eta^2}{2}M\|\nabla f(x)\|^2 \\ &= f(x) - \frac{\eta}{2}(2 - \eta M)\|\nabla f(x)\|^2 \end{aligned}$$

Rearranging yields the desired result. □

7.4 Proofs from Section 3.1.1

Lemma 12 (One-Step Descent)**.** *Let f satisfy Assumption 1. Let $\widetilde{g}(x)$ be an unbiased estimator for $\nabla f(x)$ with $\mathrm{Var}(\widetilde{g}(x)) \leq \sigma^2 + \nu\|\nabla f(x)\|^2$. Let x be the current iterate, and let z be another point, with $\delta := x - z$. The LSGD step $x_+ = x - \eta(\widetilde{g}(x) + \lambda(x - z))$ satisfies:*

$$\mathbb{E}f(x_+) \leq f(x) - \frac{\eta}{2}(1 - \eta M(\nu+1))\|\nabla f(x)\|^2 - \frac{\eta}{4}\lambda(m - 2\eta M\lambda)\|\delta\|^2 \tag{10}$$
$$- \frac{\eta\sqrt{\lambda}}{\sqrt{2}}(\sqrt{m} - \eta M\sqrt{2\lambda})\|\nabla f(x)\|\|\delta\| - \eta\lambda(f(x) - f(z)) + \frac{\eta^2}{2}M\sigma^2$$

where the expectation is with respect to $\widetilde{g}(x)$, and conditioned on the current point x. Hence, for sufficiently small η, λ with $\eta \leq (2M(\nu+1))^{-1}$ and $\eta\lambda \leq (2\kappa)^{-1}, \eta\sqrt{\lambda} \leq (\kappa\sqrt{2m})^{-1}$,

$$\mathbb{E}f(x_+) - f(x^*) \leq (1 - m\eta)(f(x) - f(x^*)) - \eta\lambda(f(x) - f(z)) + \frac{\eta^2 M}{2}\sigma^2 \tag{11}$$

Proof. The proof is similar to the convergence analysis of SGD. We apply Theorem 9 to obtain

$$f(x_+) \leq f(x) - \eta\nabla f(x)^T(\widetilde{g}(x) + \lambda\delta) + \frac{\eta^2}{2}M\|\widetilde{g}(x) + \lambda\delta\|^2.$$

Taking the expectation and using $\mathbb{E}\widetilde{g}(x) = \nabla f(x)$,

$$\mathbb{E}f(x_+) \leq f(x) - \eta\|\nabla f(x)\|^2 - \eta\lambda\nabla f(x)^T\delta + \frac{\eta^2\lambda^2}{2}M\|\delta\|^2 + \eta^2\lambda M\nabla f(x)^T\delta + \frac{\eta^2}{2}M\mathbb{E}[\widetilde{g}(x)^T\widetilde{g}(x)]$$

Using the definition of m-strong convexity, we have $f(z) \geq f(x) - \nabla f(x)^T\delta + \frac{m}{2}\|\delta\|^2$, from which we deduce that $-\nabla f(x)^T\delta \leq -(f(x) - f(z) + \frac{m}{2}\|\delta\|^2)$. Substituting this above, and splitting both the terms $\eta\|\nabla f(x)\|^2, \frac{\eta}{2}m\lambda\|\delta\|^2$ in half, we obtain

$$\begin{aligned}\mathbb{E}f(x_+) = f(x) &- \frac{\eta}{2}\|\nabla f(x)\|^2 + \frac{\eta^2}{2}M\mathbb{E}[\widetilde{g}(x)^T\widetilde{g}(x)] \\ &- \frac{\eta}{4}m\lambda\|\delta\|^2 + \frac{\eta^2}{2}\lambda^2 M\|\delta\|^2 \\ &- \frac{\eta}{2}\|\nabla f(x)\|^2 - \frac{\eta}{4}m\lambda\|\delta\|^2 + \eta^2\lambda M\nabla f(x)^T\delta \\ &- \eta\lambda(f(x) - f(z))\end{aligned}$$

We proceed to bound each line. For the first line, the standard bias-variance decomposition yields

$$\mathbb{E}[\widetilde{g}(x)^T\widetilde{g}(x)] \leq (\nu+1)\|\nabla f(x)\|^2 + \sigma^2$$

and so we have

$$-\frac{\eta}{2}\|\nabla f(x)\|^2 + \frac{\eta^2}{2}M\mathbb{E}[\widetilde{g}(x)^T\widetilde{g}(x)] \leq -\frac{\eta}{2}(1 - \eta M(\nu+1))\|\nabla f(x)\|^2 + \frac{\eta^2}{2}M\sigma^2.$$

For the second line, we obtain

$$-\frac{\eta}{4}m\lambda\|\delta\|^2 + \frac{\eta^2}{2}\lambda^2 M\|\delta\|^2 \leq -\frac{\eta}{4}\lambda(m - 2\eta M\lambda)\|\delta\|^2.$$

For the third line, we apply the inequality $a^2 + b^2 \geq 2ab$ to obtain

$$\frac{\eta}{2}\|\nabla f(x)\|^2 + \frac{\eta}{4}m\lambda\|\delta\|^2 \geq \frac{\eta}{\sqrt{2}}\sqrt{m\lambda}\|\nabla f(x)\|\|\delta\|.$$

Using the Cauchy-Schwarz inequality, we then obtain

$$-\frac{\eta}{2}\|\nabla f(x)\|^2 - \frac{\eta}{4}m\lambda\|\delta\|^2 + \eta^2\lambda\nabla M f(x)^T\delta \leq -\frac{\eta\sqrt{\lambda}}{\sqrt{2}}(\sqrt{m} - \eta M\sqrt{2\lambda})\|\nabla f(x)\|\|\delta\|.$$

Combining these inequalities yields the desired result. □

Theorem 13. *Let f satisfy Assumption 1. Suppose that the leader z_k is always chosen so that $f(z_k) \leq f(x_k)$. If η, λ are fixed so that $\eta \leq (2M(\nu+1))^{-1}$ and $\eta\lambda \leq (2\kappa)^{-1}, \eta\sqrt{\lambda} \leq (\kappa\sqrt{2m})^{-1}$, then $\limsup_{k\to\infty} \mathbb{E}f(x_k) - f(x^*) \leq \frac{1}{2}\eta\kappa\sigma^2$. If η decreases at the rate $\eta_k = \Theta(\frac{1}{k})$, then $\mathbb{E}f(x_k) - f(x^*) = O(\frac{1}{k})$.*

Proof. This result follows (11) and Theorems 4.6 and 4.7 of [30]. □

7.5 Proofs from Section 3.1.2

Theorem 14. *Let f satisfy Assumption 1. Suppose that η, λ are small enough that $\eta\lambda \leq 1$ and $\eta \leq (2M(\nu+1))^{-1}, \eta\lambda \leq (2\kappa)^{-1}, \eta\sqrt{\lambda} \leq (\kappa\sqrt{2m})^{-1}$. If $f(x) \leq f(z)$, then $\mathbb{E}f(x_+) \leq f(z) + \frac{1}{2}\eta^2 M\sigma^2$.*

Proof. This follows from (13), by combining $f(x) - \eta\lambda(f(x) - f(z))$, and using $f(z) \geq f(x)$. □

Theorem 15. *Let f be m-strongly convex, and let x^* be the minimizer of f. Fix a constant λ and any point z, and define the function $\psi(x) = f(x) + \frac{\lambda}{2}\|x - z\|^2$. Since ψ is strongly convex, it has a unique minimizer w. The minimizer w satisfies*

$$f(w) - f(x^*) \leq \frac{\lambda}{m+\lambda}(f(z) - f(x^*)) \tag{12}$$

and[5]

$$\|w - x^*\|^2 \leq \frac{\lambda^2}{m(m+\lambda)}\|z - x^*\|^2 \tag{13}$$

Proof. The first-order condition for w implies that $\nabla f(w) + \lambda(w - z) = 0$, so $\lambda^2\|w - z\|^2 = \|\nabla f(w)\|^2$. Combining this with the Polyak-Łojasiewicz inequality, we obtain

$$\frac{\lambda}{2}\|w - z\|^2 = \frac{1}{2\lambda}\|\nabla f(w)\|^2 \geq \frac{m}{\lambda}(f(w) - f(x^*))$$

We have $\psi(w) \leq \psi(z) = f(z)$, so $f(w) - f(x^*) \leq f(z) - f(x^*) - \frac{\lambda}{2}\|w - z\|^2$. Substituting, $f(w) - f(x^*) \leq f(z) - f(x^*) - \frac{m}{\lambda}(f(w) - f(x^*))$, which yields the first inequality.

We also have $\psi(w) = f(w) + \frac{\lambda}{2}\|w - z\|^2 \leq \psi(x^*) = f(x^*) + \frac{\lambda}{2}\|x^* - z\|^2$, whence $f(w) - f(x^*) \leq \frac{\lambda}{2}(\|x^* - z\|^2 - \|w - z\|^2)$. Hence, we have

$$\begin{aligned} f(w) - f(x^*) &\leq \frac{\lambda}{2}(\|x^* - z\|^2 - \|w - z\|^2) \\ &\leq \frac{\lambda}{2}\|z - x^*\|^2 - \frac{m}{\lambda}(f(w) - f(x^*)) \end{aligned}$$

so $f(w) - f(x^*) \leq \frac{\lambda^2}{2(m+\lambda)}\|z - x^*\|^2$. Finally, by Theorem 10, $f(w) - f(x^*) \geq \frac{m}{2}\|w - x^*\|^2$, which yields the result. □

7.6 Proofs from Section 3.1.3

We first present two lemmas which consider the problem of selecting the minimizer from a collection, based on a single estimate of the value of each item.

Lemma 16. *Let $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_p$. Suppose that $Y_1, \ldots, Y_p$ is a collection of random variables with $\mathbb{E}Y_i = \mu_i$ and $\mathrm{Var}(Y_i) \leq \sigma^2$. Let $\widetilde{\mu} = \mu_m$ where $m = \mathrm{argmin}\{Y_1, \ldots, Y_p\}$. Then*

$$\Pr(\widetilde{\mu} \geq \mu_k) \leq 4\sigma^2 \sum_{i=k}^{p} \frac{1}{(\mu_i - \mu_1)^2}$$

Therefore, for any $a \geq 0$,

$$\Pr(\widetilde{\mu} \geq \mu_1 + a) \leq 4\sigma^2 \frac{p}{a^2}.$$

[5] If we also assume that f is Lipschitz-differentiable (that is, $\nabla^2 f(x) \preceq MI$), then we can obtain a similar inequality to the second directly from the first, but this is generally weaker than the bound given here.

Proof. In order for $\mu_m \geq \mu_k$, we must have $Y_j \leq Y_1$ for some $j \geq k$. Thus, $\{\widetilde{\mu} \geq \mu_k\}$ is a subset of the event $\{Y_1 \geq \min\{Y_k, \ldots, Y_p\}\}$. Taking the union bound,

$$\Pr(Y_1 \geq \min\{Y_k, \ldots, Y_p\}) \leq \sum_{i=k}^{p} \Pr(Y_1 \geq Y_i)$$

Applying Chebyshev's inequality to $Y_1 - Y_i$, and noting that $\mathrm{Var}(Y_1 - Y_i) \leq 4\sigma^2$ (if Y_1, Y_i are independent, then this can be tightened to $2\sigma^2$), we have

$$\Pr(Y_1 - Y_i \geq 0) \leq \Pr(|Y_1 - Y_i - (\mu_i - \mu_1)| \geq \mu_i - \mu_1) \leq \frac{4\sigma^2}{(\mu_i - \mu_1)^2}.$$

□

Lemma 17. *Let $\widetilde{\mu}$ be defined as in Theorem 16. Then*

$$\mathbb{E}\widetilde{\mu} - \mu_1 \leq 4\sqrt{p}\sigma$$

Proof. Recall that the expected value of a non-negative random variable Z can be expressed as $\mathbb{E}Z = \int_0^\infty \Pr(Z \geq t)dt$. We apply this to the variable $\widetilde{\mu} - \mu_1$. Using Theorem 16, we obtain, for any $a > 0$,

$$\begin{aligned}
\mathbb{E}\widetilde{\mu} - \mu_1 = \int_0^\infty \Pr(\widetilde{\mu} - \mu_1 \geq t)dt &= \int_0^a \Pr(\widetilde{\mu} - \mu_1 \geq t)dt + \int_a^\infty \Pr(\mu^* - \mu_1 \geq t)dt \\
&\leq a + \int_a^\infty \Pr(\widetilde{\mu} - \mu_1 \geq t)dt \\
&\leq a + \int_a^\infty 4\sigma^2 \frac{p}{t^2} dt = a + 4\sigma^2 \frac{p}{a}
\end{aligned}$$

The AM-GM inequality implies that $a + 4\sigma^2 \frac{p}{a} \geq 4\sqrt{p}\sigma$, with equality when $a = 2\sqrt{p}\sigma$. □

We now apply this to stochastic leader selection in LSGD, where μ_i corresponds to the true value $f(x^i)$, and Y_i is a function estimator.

Lemma 18. *Let f satisfy Assumption 1. Suppose that LSGD has a gradient estimator with $\mathrm{Var}(\widetilde{g}(x)) \leq \sigma^2 + \nu\|\nabla f(x)\|^2$ and selects the stochastic leader with a function estimator $\widetilde{f}(x)$ with $\mathrm{Var}(\widetilde{f}(x)) \leq \sigma_f^2$. Then, taking the expectation with respect to the gradient estimator and the stochastic leader z, we have*

$$\begin{aligned}
\mathbb{E}f(x_+) \leq f(x) &+ 4\eta\lambda\sqrt{p}\sigma_f + \frac{\eta^2}{2}M\sigma^2 \\
&- \frac{\eta}{2}(1 - \eta M(\nu + 1))\|\nabla f(x)\|^2 - \frac{\eta}{4}\lambda(m - 2\eta M\lambda)\|\delta\|^2 - \frac{\eta\sqrt{\lambda}}{\sqrt{2}}(\sqrt{m} - \eta M\sqrt{2\lambda})\|\nabla f(x)\|\|\delta\|
\end{aligned}$$

Proof. From Theorem 12, we obtain

$$\begin{aligned}
\mathbb{E}f(x_+) \leq f(x) &- \frac{\eta}{2}(1 - \eta M(\nu + 1))\|\nabla f(x)\|^2 \\
&- \frac{\eta}{4}\lambda(m - 2\eta M\lambda)\|\delta\|^2 \\
&- \frac{\eta\sqrt{\lambda}}{\sqrt{2}}(\sqrt{m} - \eta M\sqrt{2\lambda})\|\nabla f(x)\|\|\delta\| \\
&- \eta\lambda(f(x) - \mathbb{E}f(z)) + \frac{\eta^2}{2}M\sigma^2
\end{aligned}$$

Note that in the last line, we have $\mathbb{E}f(z)$ because z is now stochastic. Applying Theorem 17 to the stochastic leader, we obtain $\mathbb{E}f(z) \leq f(z_{true}) + 4\sqrt{p}\sigma_f$. The true leader satisfies $f(z_{true}) \leq f(x)$ by definition. Hence $f(x) - \mathbb{E}f(z) \geq f(x) - f(z_{true}) - 4\sqrt{p}\sigma_f \geq -4\sqrt{p}\sigma_f$, and so $-\eta\lambda(f(x) - \mathbb{E}f(z)) \leq 4\eta\lambda\sqrt{p}\sigma_f$. □

Theorem 19. *Let f satisfy Assumption 1. If η, λ are fixed so that $\eta \leq (2M(\nu+1))^{-1}$ and $\eta\lambda \leq (2\kappa)^{-1}, \eta\sqrt{\lambda} \leq (\kappa\sqrt{2m})^{-1}$, then $\limsup_{k\to\infty} \mathbb{E}f(x_k) - f(x^*) \leq \frac{1}{2}\eta\kappa\sigma^2 + \frac{4}{m}\lambda\sqrt{p}\sigma_f$. If η, λ decrease at the rate $\eta_k = \Theta(\frac{1}{k}), \lambda_k = \Theta(\frac{1}{k})$, then $\mathbb{E}f(x_k) - f(x^*) = O(\frac{1}{k})$.*

Proof. Interpret the term $4\eta\lambda\sqrt{p}\sigma_f$ as additive noise. Note that if $\eta_k, \lambda_k = \Theta(\frac{1}{k})$, then $\eta\lambda = \Theta(\frac{1}{k^2})$. The proof is then similar to Theorem 13 and follows from Theorems 4.6 and 4.7 of [30]. □

7.7 Proofs from Section 3.2

Theorem 20. *Let Ω_i be the set of points $(x^1, \ldots, x^p)$ where x^i is the unique minimizer among $(x^1, \ldots, x^p)$[6]. Let $x^* = (w^1, \ldots, w^p) \in \Omega_i$ be a stationary point of the LGD objective function (6). Then $\nabla f^i(w^i) = 0$.*

Proof. This follows from the fact that on Ω_i, $\frac{\partial L}{\partial x^i} = \nabla f^i(x^i)$. □

Lemma 21. *Let f be M-Lipschitz-differentiable. Let $\widetilde{x}_k$ denote the leader at the end of the k-th period. If the LGD step size is chosen so that $\eta^i < \frac{2}{M}$, then $f(\widetilde{x}_k) \leq f(\widetilde{x}_{k-1})$.*

Proof. Assume that $\widetilde{x}_{k-1} = x^1_{k-1}$. Since x^1 is the leader during the k-th period, the LGD steps for x^1 are gradient descent steps. By Theorem 11, η^1 has been chosen so that gradient descent on f is monotonically decreasing, so we know that $f(x^1_k) \leq f(x^1_{k-1})$. Hence $f(\widetilde{x}_k) \leq f(x^1_k) \leq f(x^1_{k-1}) = f(\widetilde{x}_{k-1})$. □

Theorem 22. *Assume that f is bounded below and M-Lipschitz-differentiable, and that the LGD step sizes are selected so that $\eta^i < \frac{2}{M}$. Then for any choice of communication periods, it holds that for every i such that x^i is the leader infinitely often, $\liminf_k \|\nabla f(x^i_k)\| = 0$.*

Note that there necessarily exists an index i such that x^i is the leader infinitely often.

Proof. Without loss of generality, we assume it to be x^1. Let $\tau(1), \tau(2), \ldots$ denote the periods where x^1 is the leader, with $b(k)$ steps in the period $\tau(k)$. By Theorem 21, $f(x^1_{\tau(k+1)}) \leq f(x^1_{\tau(k)})$, since the objective value of the leaders is monotonically decreasing. Now, by Theorem 11, we have $\sum_{i=0}^{b(k)-1} \|\nabla f(x^1_{\tau(k),i})\|^2 \leq \alpha(f(x^1_{\tau(k),0}) - f(x^1_{\tau(k),b(k)})) = \alpha(f(x^1_{\tau(k)}) - f(x^1_{\tau(k+1)}))$. Since f is bounded below, and the sequence $\{f(x^1_{\tau(k)})\}$ is monotonically decreasing, we must have $f(x^1_{\tau(k)}) - f(x^1_{\tau(k+1)}) \to 0$. Therefore, we must have $\|\nabla f(x^1_{\tau(k),i})\| \to 0$. □

7.8 Proofs from Section 3.3

The *cone* with center d and angle θ_c is defined to be

$$\text{cone}(d, \theta_c) = \{x : x^T d \geq 0, \theta(x, d) \leq \theta_c\}.$$

We record the following facts about cones which will be useful.

Proposition 23. *Let $C \subseteq \text{cone}(d, \theta_c)$. If y is a point such that $sy \in C$ for some $s \geq 0$, then $y \in \text{cone}(d, \theta_c)$.*

Proof. This follows immediately from the fact that $\theta(y, d) = \theta(sy, d)$ for all $s \geq 0$. □

Proposition 24. *Let $C = \text{cone}(d, \theta_c)$ with $\theta_c > 0$. The outward normal vector at the point $x \in \partial C$ is given by $N_x = x - \frac{\|x\|}{\cos(\theta_c)\|d\|}d$. Moreover, if v satisfies $N_x^T v < 0$, then for sufficiently small positive λ, $x + \lambda v \in \text{cone}(d, \theta_c)$.*

[6]The uniqueness of the minimizer on Ω_i is only to avoid ambiguities in $\arg\min$.

Proof. The first statement follows from the second, by the supporting hyperplane theorem.

Write $\gamma = \cos(\theta_c)$. Let $N_x = x - \frac{\|x\|}{\gamma\|d\|}d$, and let v be a unit vector with $N_x^T v = x^T v - \frac{\|x\|}{\gamma\|d\|}d^T v < 0$. The angle satisfies

$$\cos(\theta(x+\lambda v, d)) = \frac{d^T(x+\lambda v)}{\|d\|\|x+\lambda v\|} = \frac{d^T x + \lambda d^T v}{\|d\|\sqrt{\|x\|^2 + \lambda^2\|v\|^2 + 2\lambda x^T v}}$$

Differentiating, the numerator $g(\lambda)$ of $\frac{\partial}{\partial\lambda}\cos(\theta(x+\lambda v, d))$ is given by

$$g(\lambda) = \|x\|^2 v^T d - x^T v x^T d + \lambda \cdot (2v^T d x^T d + \|v\|^2(\lambda v - x)^T d - \lambda\|v\|^2 v^T d - x^T v v^T d)$$

Evaluating at $\lambda = 0$ and using $x^T v - \frac{\|x\|}{\gamma\|d\|}d^T v < 0$, we obtain

$$\begin{aligned} g(0) = \|x\|^2 v^T d - x^T v x^T d &= \|x\|^2 v^T d - x^T v(\gamma\|x\|\|d\|) \\ &= \|x\|(\|x\|v^T d - \gamma\|d\|x^T v) > 0. \end{aligned}$$

Therefore, for small positive λ, we have $\cos(\theta(x+\lambda v, d)) > \cos(\theta(x, d)) = \gamma$, so $x + \theta v \in \mathrm{cone}(d, \theta_c)$. □

Proposition 25. *Let x be any point such that $\theta_x = \theta(d_G(x), d_N(x)) > 0$, and let $E = \{z : f(z) \le f(x)\}$. Let $C = \mathrm{cone}(-x, \theta_x)$, and let N_x be the outward normal $-\nabla f(x) + \frac{\|\nabla f(x)\|}{\cos(\theta_x)\|x\|}x$ of the cone C at the point $-\nabla f(x)$. Then*

$$\bigcup_{\lambda>0} I_\theta(x,\lambda) \supseteq E \cap \{z : N_x^T z < N_x^T x\} \tag{14}$$

and consequently, $\lim_{\lambda\to 0}\mathrm{Vol}(I_\theta(x,\lambda)) \ge \frac{1}{2}\mathrm{Vol}(E)$.

Proof. First, note that if $\lambda_2 \le \lambda_1$, then for all z with $-\nabla f(x) + \lambda_1 z \in C$, we also have $-\nabla f(x) + \lambda_2 z \in C$ by the convexity of C. Therefore $I_\theta(x,\lambda_2) \supseteq I_\theta(x,\lambda_1)$, so $\lim_{\lambda\to 0}\mathrm{Vol}(I_\theta(x,\lambda))$ exists. We first prove the second statement. For any normal vector h and $\beta > 0$, $\mathrm{Vol}(E \cap \{z : h^T z < \beta\}) \ge \frac{1}{2}\mathrm{Vol}(E)$, since the center $0 \in \{z : h^T z < \beta\}$. The result follows because $N_x^T x > 0$.

To prove (14), observe that $z \in I_\theta(x,\lambda)$ if equivalent to $-\nabla f(x) + \lambda(z-x) \in \mathrm{cone}(-x, \theta_c)$. By Theorem 24, there exists $\lambda > 0$ with $-\nabla f(x) + \lambda(z-x) \in \mathrm{cone}(-x,\theta_c)$ if $N_x^T(z-x) < 0$. Hence, it follows that every point in $E \cap \{z : N^T z < N^T x\}$ is contained in $I_\theta(x,\lambda)$ for some $\lambda > 0$. □

Lemma 26. *There exists a direction x such that $\cos(\theta(d_G(x), d_N(x))) = 2(\sqrt{\kappa} + \sqrt{\kappa^{-1}})^{-1}$. Thus, for all $r \ge 2$, there exists a direction x with $\cos(\theta(d_G(x), d_N(x))) \le \frac{r}{\sqrt{\kappa}}$.*

Proof. Take $x = \sqrt{\frac{\alpha_n}{\alpha_1+\alpha_n}}e_1 + \sqrt{\frac{\alpha_1}{\alpha_1+\alpha_n}}e_n$. It is easy to verify that $\cos(\theta(d_G, d_N)) = 2(\sqrt{\kappa} + \sqrt{\kappa^{-1}})^{-1}$. □

Proposition 27. *For any x, let $\theta_x = \theta(d_G(x), d_N(x))$. We have*

$$\max\{\|z\|^2 : f(z) \le f(x), z^T x = 0\} \le \kappa\cos(\theta_x)\|x\|^2$$

Proof. Form the maximization problem

$$\begin{cases} \max\limits_{z} & z^T z \\ & z^T A z \le x^T A x \\ & z^T x = 0 \end{cases}$$

The KKT conditions for this problem imply that the solution satisfies $z - \mu_1 A z - \mu_2 x = 0$, for Lagrange multipliers $\mu_1 \ge 0, \mu_2$. Since $z^T x = 0$, we obtain $z^T z = \mu_1 z^T A z$, and thus $\frac{1}{M} \le \mu_1 \le \frac{1}{m}$. Since $f(z) \le f(x)$, we find that $z^T z \le \frac{1}{m}x^T A x$. Using $\cos(\theta_x) = \frac{x^T Ax}{\|x\|\|Ax\|}$, we obtain

$$z^T z \le \frac{1}{m}\cos(\theta_x)\|x\|\|Ax\| \le \kappa\cos(\theta_x)\|x\|^2.$$

□

Theorem 28. *Let* $R_\kappa = \{r : \frac{r}{\sqrt{\kappa}} + \frac{r^{3/2}}{\kappa^{1/4}} \leq 1\}$. *Let* $x \in S_r$ *for* $r \in R_\kappa$, *and let* $E = \{y : f(y) \leq f(x)\}$, $E_2 = \{z \in E : z^T x \leq 0\}$, $\theta_x = \theta(d_G(x), d_N(x))$. *Then for all* $z \in E_2$ *and any* $\lambda \geq 0$, *the LGD direction* $d_z = -(\nabla f(x) + \lambda(x - z))$ *satisfies* $\theta(d_z, d_N(x)) \leq \theta_x$. *Thus,* $E_2 \subseteq I_\theta(x, \lambda)$, *and therefore* $\mathrm{Vol}(I_\theta(x, \lambda)) \geq \mathrm{Vol}(E_2) = \frac{1}{2}\mathrm{Vol}(E)$.

Proof. Define $D_2 = \{z - x : z \in E_2\}$[7]. The set of possible LGD directions with $z \in E_2$ is given by $D_3 = \{-\nabla f(x) + \lambda\delta : \delta \in D_2, \lambda \geq 0\}$. Since $d_N(x) = -x$, our desired result is equivalent to $D_3 \subseteq \mathrm{cone}(-x, \theta_x)$.

Define the subset $D_2' = \{z - x : z \in E_2, x^T z = 0\}$. We claim that it suffices to prove that $D_2' \subseteq \mathrm{cone}(-x, \theta_x)$. To see this, consider any $\lambda\delta$ for $\lambda \geq 0$ and $\delta \in D_2$. We have $x^T(\lambda\delta) = \lambda x^T(z - x) \leq -\lambda x^T x < 0$, so there exists a scalar s with $x^T(s\lambda\delta) = -x^T x$, whence $s\lambda\delta \in D_2' \subseteq \mathrm{cone}(-x, \theta_x)$. By Theorem 23, $\lambda\delta \in \mathrm{cone}(-x, \theta_x)$. Since $-\nabla f(x) \in \mathrm{cone}(-x, \theta_x)$, convexity implies that $-\nabla f(x) + \lambda\delta \in \mathrm{cone}(-x, \theta_x)$. Thus, $D_2' \subseteq \mathrm{cone}(-x, \theta_x)$ implies that $D_3 \subseteq \mathrm{cone}(-x, \theta_x)$.

To complete the proof, let $\delta = z - x \in D_2'$ and observe that $\cos(\theta(\delta, d_N(x))) = \frac{x^T(x-z)}{\|x\|\|x-z\|}$. By Theorem 27 and the definition of S_r,

$$\max\{\|z\| : z \in E_2, z^T x = 0\} \leq \sqrt{\kappa}\sqrt{\cos(\theta_x)}\|x\| = \sqrt{r}\kappa^{1/4}\|x\|$$

We compute that

$$\begin{aligned} x^T(x - z) - \frac{r}{\sqrt{\kappa}}\|x\|\|x - z\| &\geq \|x\|^2 - \frac{r}{\sqrt{\kappa}}(\|x\|^2 + \|x\|\|z\|) \\ &\geq \|x\|^2 - \frac{r}{\sqrt{\kappa}}\|x\|^2 - \frac{r}{\sqrt{\kappa}}\|x\|(\sqrt{r}\kappa^{1/4}\|x\|) \\ &\geq \left(1 - \frac{r}{\sqrt{\kappa}} - \frac{r^{3/2}}{\kappa^{1/4}}\right)\|x\|^2 \geq 0 \end{aligned}$$

By the definition of R_κ, this is non-negative, and thus $\theta(\delta, d_N(x)) \leq \theta_x$. This completes the proof. □

8 Low-Rank Matrix Completion Experiments

Low-rank matrix completion problem is an example of a non-convex learning problem whose landscape exhibits numerous symmetries. We consider the positive semi-definite case, where the objective is to find a low-rank matrix minimizing

$$\min_X \left\{ F(X) = \frac{1}{4}\|M - XX^T\|_F^2 : X \in \mathbb{R}^{d \times r} \right\}$$

It is routine to calculate that $\nabla F(X) = (XX^T - M)X$. The EAGD and LGD updates for X can be expressed as

$$X_+ = (1 - \eta\lambda)X + \eta\lambda Z - \eta\nabla F(X).$$

For EAGD, $Z = \widetilde{X}$, and $\widetilde{X}$ is updated by

$$\widetilde{X}_+ = (1 - p\eta\lambda)\widetilde{X} + p\eta\lambda\left(\frac{1}{p}\sum_{i=1}^{p} X^i\right).$$

For LGD, $Z = \arg\min\{F(X^1), \ldots, F(X^p)\}$, and is updated at the beginning of every communication period τ.

The parameters were set to:

$$\eta = \texttt{5e-4}, \lambda = \frac{1}{5}, p = 8, \tau = 1$$

The learning rate $\eta = \texttt{5e-4}$ was selected from a set $\{\texttt{1e-1}, \texttt{5e-2}, \texttt{1e-3}, \ldots\}$ by evaluating on a sample problem until a value was found for which both methods exhibited monotonic decrease.

[7] Note the sign change from $x - z$ to $z - x$ here.

The dimension was $d = 1000$, and the ranks $r \in \{1, 10, 50, 100\}$ were tested. For each rank, there were 10 random trials performed. In each trial, M and starting points $\{X_0^i\}$ are sampled. M is generated by sampling $U \in \mathbb{R}^{d \times r}$ with i.i.d entries from $N(0, 1)$, and taking $M = UU^T$. Initial points for each worker node X^i were also sampled from $N(0, 1)$. The same starting points were used for EAGD and LGD.

Code for this experiment is available at `https://github.com/wgao-res/lsgd_matrix_completion`.

9 Experimental Setup

9.1 Data preprocessing

For CIFAR-10 experiments we use the original images of size $3 \times 32 \times 32$. We then normalize each image by mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2023, 0.1994, 0.2010)$. We also augment the training data by horizontal flips with a probability of 0.5.

For CNN7 and ResNet20, we extract random crops of size $3 \times 28 \times 28$ and present these to the network in batches of size 128. The test loss and test error are only computed from the center patch $(3 \times 28 \times 28)$ of test images.

For VGG16 we pad the images to $3 \times 40 \times 40$, extract random crops of size $3 \times 32 \times 32$ and present these to the network in batches of size 128. The test loss and test error are computed from the test images.

For ImageNet experiments we normalize each image by mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$. We sample the training data in the same way as [39]. For each image, a crop of random size (chosen from 8% to 100% evenly) of the original size and a random aspect ratio (chosen from $3/4$ to $4/3$ evenly) of the original aspect ratio is made. Then we resize the crop to $3 \times 224 \times 224$. We also augment the training data by horizontal flips with a probability of 0.5. Finally we present these to the network in the batches of size 32. The test images are resized so that the smaller edge of each image is 256. The test loss and test error are only computed from the center patch $(3 \times 224 \times 224)$ of test images.

9.2 Data prefetching

We use the dataloader and distributed data sampler[8] from PyTorch. Each worker loads a subset of the original data set that is exclusive to that worker for every epoch. If the size of data set is not divisible by the batch size, the last incomplete batch will be dropped.

9.3 Hyperparameters

In Table 2 we summarize the learning rates and other hyperparameters explored for each method in the CNN7 experiment on CIFAR-10. The setting of β for EASGD was obtained from the original paper (its authors use this setting for all their experiments).

Table 2: Hyperparameters: CNN7 experiment on CIFAR-10

Name	Learning Rates	
SGD	$\{0.1, 0.05, 0.01, 0.005, 0.001\}$	
DOWNPOUR	$\{0.05, 0.01, 0.005, 0.001, 0.0005\}$	
EASGD	$\{0.1, 0.05, 0.01, 0.005, 0.001\}$	$\beta = 0.43$
LSGD	$\{0.1, 0.05, 0.01, 0.005, 0.001\}$	$\lambda = \{0.5, 0.2, 0.1, 0.05, 0.025\}$, $\lambda_G = \lambda$

In Table 3 we summarize the initial learning rates and other hyperparameters explored for each method in the ResNet20 experiment on CIFAR-10. We do learning rate drop at 1500 seconds by a factor of 0.1 for all the methods.

[8] https://pytorch.org/docs/stable/data.html

Table 3: Hyperparameters: ResNet20 experiment on CIFAR-10

Name	Learning Rates	
SGD	$\{0.2, 0.1, 0.05\}$	
DOWNPOUR	$\{0.2, 0.1, 0.05, 0.01\}$	
EASGD	$\{0.2, 0.1, 0.05\}$	$\beta = 0.43$
LSGD	$\{0.2, 0.1, 0.05\}$	$\lambda = \{0.5, 0.2, 0.1, 0.05, 0.025\}$, $\lambda_G = \lambda$

In Table 4 we summarize the learning rates and other hyperparameters explored for each method in the VGG16 experiment on CIFAR-10. We do learning rate drop at 1500 seconds by a factor of 0.1 for all the methods.

Table 4: Hyperparameters: VGG16 experiment on CIFAR-10

Name	Learning Rates	
SGD	$\{0.2, 0.1, 0.05\}$	
DOWNPOUR	$\{0.2, 0.1, 0.05, 0.01\}$	
EASGD	$\{0.2, 0.1, 0.05\}$	$\beta = 0.43$
LSGD	$\{0.2, 0.1, 0.05\}$	$\lambda = \{0.2, 0.1\}$

In Table 5 we summarize the initial learning rates and other hyperparameters explored for each method in the ResNet50 experiment on ImageNet. We do learning rate drop for every 30 epochs by a factor of 0.1 for all the methods.

Table 5: Hyperparameters: ResNet50 experiment on ImageNet

Name	Learning Rate	
DOWNPOUR	0.1	
EASGD	0.2	$\beta = 0.43$
LSGD	0.2	$\lambda = 0.1$

9.4 Implementation Details

To take advantage of both the efficiency of collective communication and the flexibility of peer-to-peer communication, we incorporate two backends, namely NCCL and GLOO[9], for GPU processors and CPU processors, respectively.

The global and local servers (running on CPU processors) control the training process and the workers (running on GPU processors) perform the actual computations. For each iteration each worker has only one of the following two choices:

1. Local Training: Each worker is trained with one batch of the training data;

2. Distributed Training: Each worker communicates with other workers and updates its parameters based on the pre-defined distributed training method.

To minimize the cost of communication over Ethernet, the global server is running on the first GPU node instead of a separate machine. Also, for a fair comparison, the center variable is being maintained and updated by the first GPU node as well[10].

[9] https://github.com/facebookincubator/gloo

[10] In the original implementation of [1] and [9], an individual parameter server is used for updating the center variable based on the peer-to-peer communication scheme. However, there is no need to use an individual parameter server under collective communication scheme as it will only induce extra communication cost.

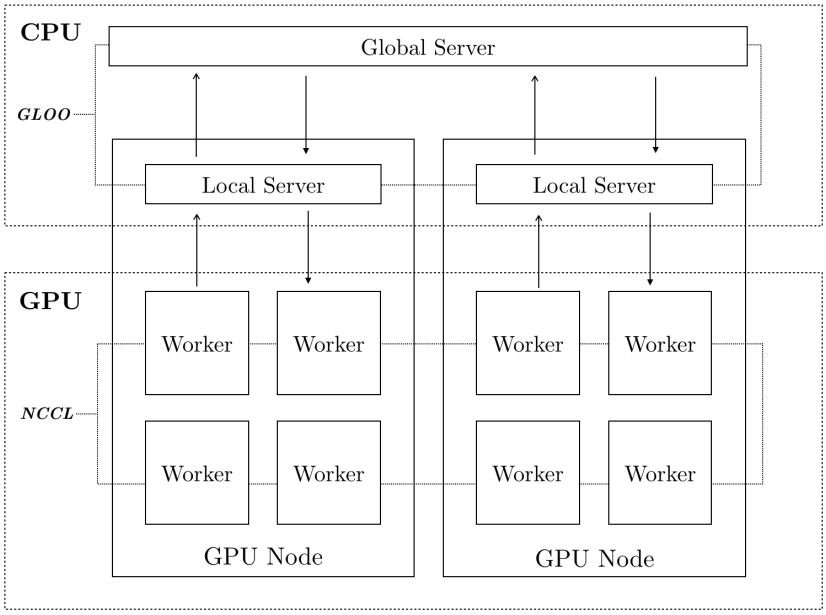

Figure 8: At the beginning of each iteration, the local worker sends out a request to its local server and then the local server passes on the worker's request to the global server. The global server checks the current status and replies to the local server. The local server passes on the global server's message to the worker. Finally, depending on the message from the global server, the worker will choose to follow the local training or distributed training scheme.

10 Additional Experimental Results

10.1 Word-level Language Model

We train an LSTM model on Wikitext-2 for word-level text prediction. Our network consists of two layers with 200 hidden units and we set the sequence length to 35. The implementation is adapted from PyTorch example[11]. In Figure 9 we show that LSGD outperforms other comparators.

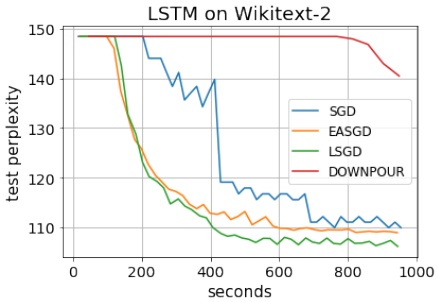

Figure 9: LSTM on Wikitext-2. Test perplexity for the center variable versus wall-clock time. The number of workers is set to 4.

10.2 More results from Section 4.2

[11] `https://github.com/pytorch/examples/tree/master/word_language_model`

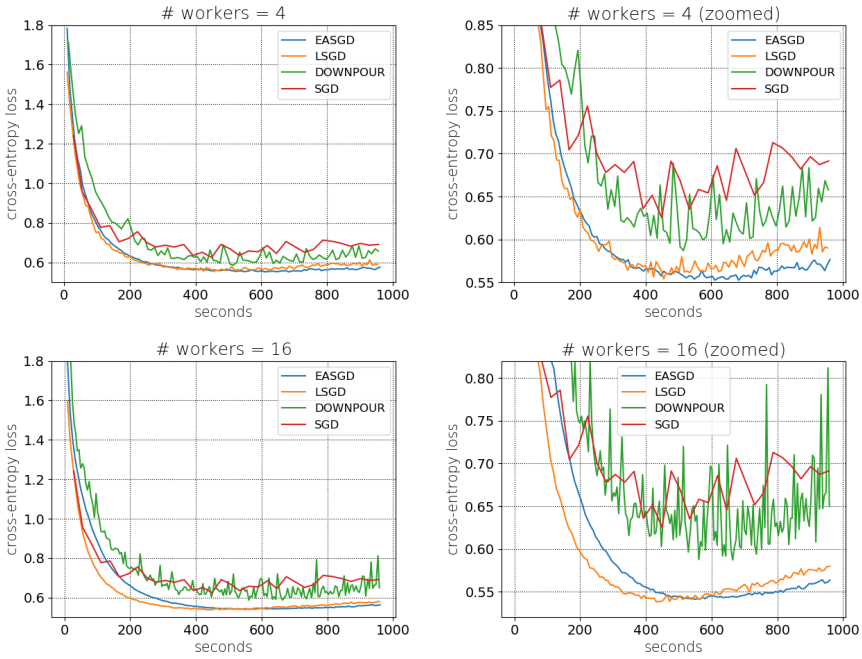

Figure 10: CNN7 on CIFAR-10. Test loss for the center variable versus wall-clock time (original plot on the left and zoomed on the right).

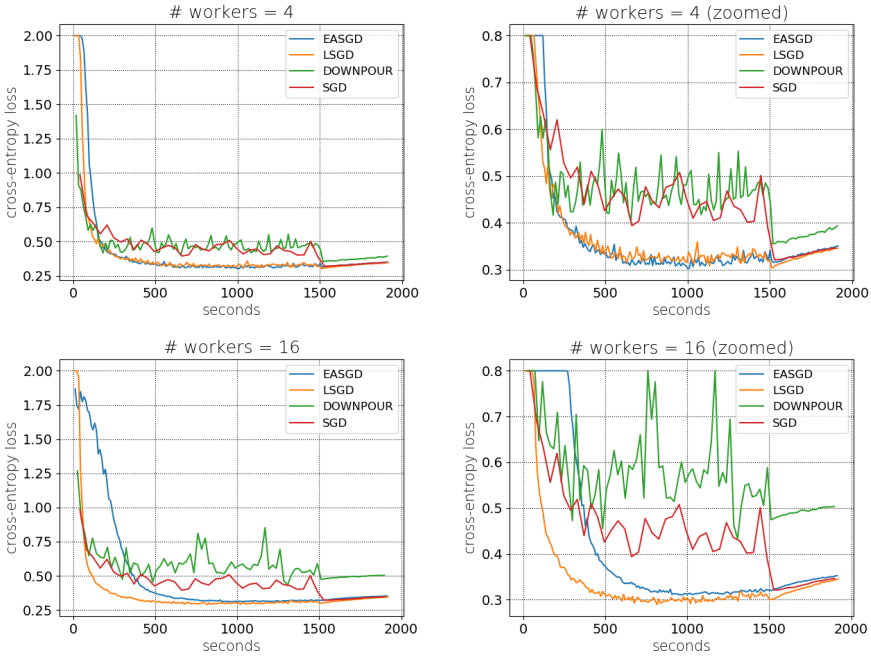

Figure 11: ResNet20 on CIFAR-10. Test loss for the center variable versus wall-clock time (original plot on the left and zoomed on the right).

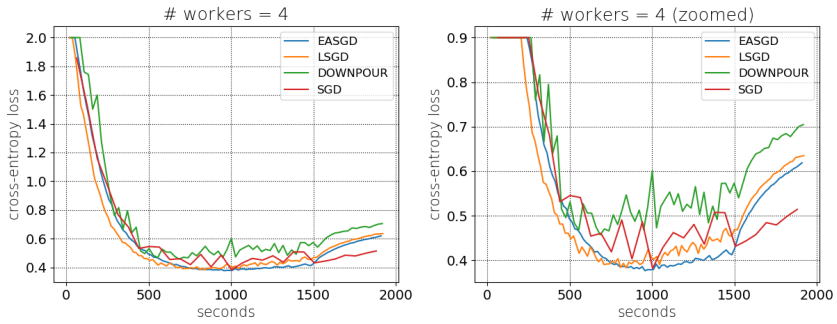

Figure 12: VGG16 on CIFAR-10. Test loss for the center variable versus wall-clock time (original plot on the left and zoomed on the right).

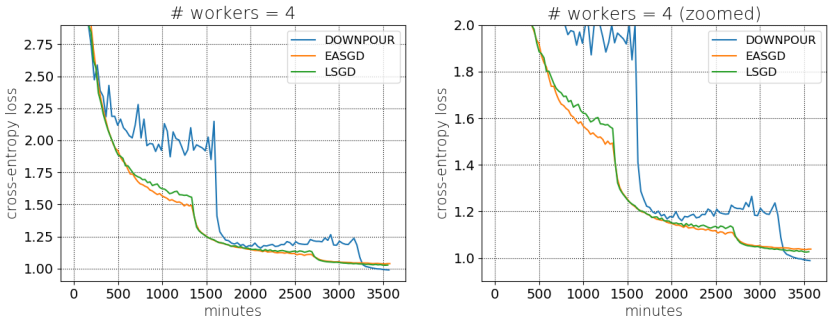

Figure 13: ResNet50 on ImageNet. Test loss for the center variable versus wall-clock time (original plot on the left and zoomed on the right).

11 Communication Efficiency

We report the proportion of communication costs with respect to the total time in Table 6. LSGD is roughly twice more communication-efficient than EASGD. Note that EASGD and DOWNPOUR require more time for data transmission and computation during communication as parameter updates involve an additional center variable.

Table 6: Proportion of communication costs with repect to the total time. Communication cost includes both data transmission and computation.

	LSGD	EASGD	DOWNPOUR
CNN7: 4/16 workers	1%/2%	2%/4%	20%/57%
ResNet20: 4/16 workers	1%/2%	2%/4%	21%/50%
VGG16	2%	3%	34%
ResNet50	1%	2%	17%