OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Adaptive Learning of the Optimal Batch Size of SGD

Motasem Alfarra MOTASEM.ALFARRA@KAUST.EDU.SA
Slavomír Hanzely SLAVOMIR.HANZELY@KAUST.EDU.SA
Alyazeed Albasyoni ALYAZEED.ALBASYONI@KAUST.EDU.SA
Bernard Ghanem BERNARD.GHANEM@KAUST.EDU.SA
Peter Richtárik PETER.RICHTARIK@KAUST.EDU.SA

Abstract

Recent advances in the theoretical understanding of SGD [8] led to a formula for the optimal batch size minimizing the number of effective data passes, i.e., the number of iterations times the batch size. However, this formula is of no practical value as it depends on the knowledge of the variance of the stochastic gradients evaluated at the optimum. In this paper we design a practical SGD method capable of learning the optimal batch size adaptively throughout its iterations for strongly convex and smooth functions. Our method does this provably, and in our experiments with synthetic and real data robustly exhibits nearly optimal behaviour; that is, it works as if the optimal batch size was known a-priori. Further, we generalize our method to several new batch strategies not considered in the literature before, including a sampling suitable for distributed implementations.

1. Introduction

Stochastic Gradient Descent (SGD), in one disguise or another, is undoubtedly the backbone of modern systems for training supervised machine learning models [2, 7, 9]. The method earns its popularity due to its superior performance on very large datasets where more traditional methods such as gradient descent (GD), relying on a pass through the entire training dataset before adjusting the model parameters, are simply too slow to be useful. In contrast, SGD in each iteration uses a small portion of the training data only (a batch) to adjust the model parameters, and this process repeats until a model of suitable quality is found. In practice, batch SGD is virtually always applied to a finite-sum problem of the form $x^* = \arg\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$, where n is the number of training data and $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ represents the average loss, i.e. empirical risk, of model x on the training dataset. With this formalism in place, a generic batch SGD method performs the iteration $x^{k+1} = x^k - \gamma^k \sum_{i\in S^k} v_i^k \nabla f_i(x^k)$, where $S^k \subseteq \{1, 2, \ldots, n\}$ is the batch considered in iteration k and $v_1^k, \ldots, v_n^k$ are appropriately chosen scalars. Often in practice, and almost invariably in theory, the batch S^k is chosen at random according to some fixed probability law, and the scalars v_i^k are chosen to ensure that $g^k = \sum_{i\in S^k} v_i^k \nabla f_i(x^k)$ is an unbiased estimator of the gradient $\nabla f(x^k)$. One standard choice is to fix a batch size $\tau \in \{1, 2, \ldots, n\}$, and pick S^k uniformly from all subsets of size τ. Another option is to partition the training dataset into n/τ subsets of size τ, and then in each iteration let S^k to be one of these partitions, chosen with some probability, e.g., uniformly.

Contributions Effective online learning of the optimal batch size. We make a step towards the development of a practical variant of optimal batch SGD, aiming to learn the optimal batch size τ^* on the fly. To the best of our knowledge, our method (Algorithm 1) is the first variant of SGD able to learn the optimal batch. **Sampling strategies.** We do not limit our selves to the uniform sampling strategy we used for illustration purposes above and develop closed-form expressions for the optimal batch size for several other sampling techniques. Our adaptive method works well for all of them. **Convergence theory.** We prove that our adaptive method converges, and moreover learns the optimal batch size. **Practical robustness.** We show the algorithm's robustness by conducting extensive experiments using different sampling techniques and different machine learning models on both real and synthetic datasets.

Related work A stream of research attempts to boost the performance of SGD in practice is tuning its hyperparameters such as learning rate, and batch size while training. In this context, a lot of work has been done in proposing various learning rate schedulers [1, 6, 10, 11, 13, 16]. De et al. [4] showed that one can reduce the variance by increasing the batch size without decreasing step-size (to maintain the constant signal to noise ratio). Besides, Smith et al. [12] demonstrated the effect of increasing the batch size instead of decreasing the learning rate in training a deep neural network. However, most of these strategies are based on empirical results only. You et al. [14, 15] show empirically the advantage of training on large batch size, while Masters and Luschi [5] claim that it is preferable to train on smaller one.

SGD Overview. To study batch SGD for virtually all (stationary) subsampling rules, we adopt the stochastic reformulation paradigm for finite-sum problems proposed in [8]. The random vector $v \in \mathbb{R}^n$ is sampled from a distribution $\mathcal{D}$ and satisfies $\mathbb{E}_{\mathcal{D}}[v_i] = 1$. Typically, the vector v is defined by first choosing a random batch $\mathcal{S}^k \subseteq \{1, 2, \ldots, n\}$, then defining $v_i^k = 0$ for $i \notin \mathcal{S}^k$, and choosing v_i^k to an appropriate value for $i \notin \mathcal{S}^k$ in order to make sure the stochastic gradient $\nabla f_{v^k}(x^k)$ is unbiased. In this work, we consider two particular choices of the probability law governing the selection of $\mathcal{S}^k$.

$\tau-$**partition nice sampling** In this sampling, we divide the training set into partitions $\mathcal{C}_j$ (of possibly different sizes $n_{\mathcal{C}_j}$), and each of them has at least a cardinality of τ. At each iteration, one of the sets $\mathcal{C}_j$ is chosen with probability $q_{\mathcal{C}_j}$, and then $\tau-$nice sampling (without replacement) is applied on the chosen set. For each subset C cardinality τ of partition C_j cardinality n_{C_j}, $\mathbb{P}[v_C = \mathbb{1}_{C \in \mathcal{S}}] = q_j / \binom{n_{C_j}}{\tau}$. $\tau-$**partition independent sampling** Similar to $\tau-$partition nice sampling, we divide the training set into partitions $\mathcal{C}_j$, and each of them has at least a cardinality of τ. At each iteration one of the sets $\mathcal{C}_j$ is chosen with probability $q_{\mathcal{C}_j}$, and then $\tau-$independent sampling is applied on the chosen set. For each element i of partition C_j, we have $\mathbb{P}[v_i = \mathbb{1}_{C \in \mathcal{S}}] = q_j p_i$. The stochastic formulation naturally leads to the following concept of expected smoothness.

Assumption 1 *The function f is $\mathcal{L}-$smooth with respect to a datasets $\mathcal{D}$ if there exist $\mathcal{L} > 0$ with*

$$\mathbb{E}_{\mathcal{D}}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|^2\right] \leq 2\mathcal{L}(f(x) - f(x^*)). \tag{1}$$

Assumption 2 *The gradient noise $\sigma = \sigma(x^*)$, where $\sigma(x) := \mathbb{E}_{\mathcal{D}}[\|\nabla f_v(x)\|^2]$, is finite.*

Theorem 1 *Assume f is $\mu-$strongly convex and Assumptions 1, 2 are satisfied. For any $\epsilon > 0$, if the learning rate γ is set to be*

$$\gamma = \tfrac{1}{2}\min\left\{\tfrac{1}{\mathcal{L}}, \tfrac{\epsilon\mu}{2\sigma}\right\} \quad \textit{and} \quad k \geq \tfrac{2}{\mu}\max\left\{\mathcal{L}, \tfrac{2\sigma}{\epsilon\mu}\right\}\log\left(\tfrac{2\|x^0 - x^*\|^2}{\epsilon}\right) \;\; \textit{then}\; \mathbb{E}\left\|x^k - x^*\right\|^2 \leq \epsilon \tag{2}$$

2. Deriving Optimal Batch Size

After giving this thorough introduction to the stochastic reformulation of SGD, we can move on to study the effect of the batch size on the total iteration complexity. In fact, for each sampling technique, the batch size will affect both the expected smoothness $\mathcal{L}$ and the gradient noise σ. This effect reflects on the number of iterations required to reach to ϵ neighborhood around the optimum.

Formulas for $\mathcal{L}$ and σ Before proceeding, we establish some terminologies. In addition of having f to be $L-$smooth, we also assume each f_i to be L_i-smooth. In $\tau-$partition samplings (both nice and independent), let $n_{\mathcal{C}_j}$ be number of data-points in the partition $\mathcal{C}_j$, where $n_{\mathcal{C}_j} \geq \tau$. Let $L_{\mathcal{C}_j}$ be the smoothness constants of the function $f_{\mathcal{C}_j} = \frac{1}{n_{\mathcal{C}_j}} \sum_{i \in \mathcal{C}_j} f_i$. Also, let $\overline{L}_{\mathcal{C}_j} = \frac{1}{n_{\mathcal{C}_j}} \sum_{i \in \mathcal{C}_j} L_i$ be the average of the Lipschitz smoothness constants of the functions in partition $\mathcal{C}_j$. In addition, let $h_{\mathcal{C}_j}(x) = \left\|\nabla f_{\mathcal{C}_j}(x)\right\|^2$ be the norm of the gradient of $f_{\mathcal{C}_j}$ at x. Finally, let $\overline{h}_{\mathcal{C}_j}(x) = \frac{1}{n_{\mathcal{C}_j}} \sum_{i \in \mathcal{C}_j} h_i(x)$. For ease of notation, we will drop x from all of the expression since it is understood from the context $(h_i = h_i(x))$. Also, superscripts with $(*, k)$ refer to evaluating the function at x^* and x^k respectively $(e.g.\ h_i^* = h_i(x^*))$. Now we introduce our key lemma, which gives an estimate of the expected smoothness for different sampling techniques.

Lemma 2 *For the considered samplings, the expected smoothness constants $\mathcal{L}$ can be upper bounded by $\mathcal{L}(\tau)$ (i.e. $\mathcal{L} \leq \mathcal{L}(\tau)$), where $\mathcal{L}(\tau)$ is expressed as follows*

(i) For τ-partition nice sampling,$\mathcal{L}(\tau) = \frac{1}{n\tau} \max_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}}{q_{\mathcal{C}_j}(n_{\mathcal{C}_j}-1)} \left[(\tau-1) L_{\mathcal{C}_j} n_{\mathcal{C}_j} + (n_{\mathcal{C}_j}-\tau) \max_{i \in \mathcal{C}_j} L_i\right]$.

(ii) For τ-partition independent sampling, we have: $\mathcal{L}(\tau) = \frac{1}{n} \max_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j} L_{\mathcal{C}_j}}{q_{\mathcal{C}_j}} + \max_{i \in C_j} \frac{L_i(1-p_i)}{q_{C_j} p_i}$.

For the considered samplings, the gradient noise is given by $\sigma(x^*, \tau)$, where

(i) For τ-partition nice sampling,$\sigma(x, \tau) = \frac{1}{n^2\tau} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2}{q_{\mathcal{C}_j}(n_{\mathcal{C}_j}-1)} \left[(\tau-1) h_{\mathcal{C}_j} n_{\mathcal{C}_j} + (n_{\mathcal{C}_j}-\tau)\overline{h}_{\mathcal{C}_j}\right]$.

(ii) For $\tau-$partition independent sampling, we have $\sigma(x, \tau) = \frac{1}{n^2} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2 h_{\mathcal{C}_j} + \sum_{i \in C_j} \frac{1-p_i}{p_i} h_i}{q_{C_j}}$

Optimal Batch Size Our goal is to estimate total iteration complexity as a function of τ. In each iteration, we work with τ gradients, thus we can lower bound on the total iteration complexity by multiplying lower bound on iteration complexity (2) by τ. We can apply similar analysis as in [8]. Since we have estimates on both the expected smoothness constant and the gradient noise in terms of the batch size τ, we can lower bound total iteration complexity (2) as $T(\tau) = \frac{2}{\mu} \max\left\{\tau\mathcal{L}(\tau), \frac{2}{\epsilon\mu}\tau\sigma(x^*, \tau)\right\} \log\left(\frac{2\|x^0 - x^*\|^2}{\epsilon}\right)$. Note that if we are interested in minimizer of $T(\tau)$, we can drop all constant terms in τ. Therefore, optimal batch size τ^* minimizes $\max\left\{\tau\mathcal{L}(\tau), \frac{2}{\epsilon\mu}\tau\sigma(x^*, \tau)\right\}$. It turns out that all $\tau\mathcal{L}(\tau)$, and $\tau\sigma(x^*, \tau)$ from Lemma 2, are piecewise linear functions in τ, which is crucial in helping us find the optimal τ^* that minimizes $T(\tau)$ which can be accomplished through the following theorem.

Theorem 3 *For τ-partition nice sampling and $\tau-$partition independent sampling with $p_i = \frac{\tau}{n_{\mathcal{C}_j}}$, the optimal batch size is $\tau(x^*)$, where $\tau(x)$ is given by*

$$\min_{\mathcal{C}_r} \frac{\frac{n n_{\mathcal{C}_r}^2}{e_{\mathcal{C}_r}}(L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r}) + \frac{2}{\epsilon\mu} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^3}{e_{\mathcal{C}_j}}\left(\overline{h}_{\mathcal{C}_j} - h_{\mathcal{C}_j}\right)}{\frac{n n_{\mathcal{C}_r}}{e_{\mathcal{C}_r}}(n_{\mathcal{C}_r} L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r}) + \frac{2}{\epsilon\mu} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2}{e_{\mathcal{C}_j}}(\overline{h}_{\mathcal{C}_j} - n_{\mathcal{C}_j} h_{\mathcal{C}_j})}, \quad \min_{\mathcal{C}_r} \frac{\frac{2}{\epsilon\mu} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2}{q_{\mathcal{C}_j}} \overline{h}_{\mathcal{C}_j} - \frac{n}{q_{\mathcal{C}_r}} L_{\max}^{\mathcal{C}_r}}{\frac{2}{\epsilon\mu} \sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}}{q_{\mathcal{C}_j}}(\overline{h}_{\mathcal{C}_j} - n_{\mathcal{C}_j} h_{\mathcal{C}_j}) + \frac{n}{q_{\mathcal{C}_r}}(n_{\mathcal{C}_r} L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r})},$$

respectively, if $\sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2}{e_j}(h_{\mathcal{C}_j}^* n_{\mathcal{C}_j} - \overline{h}_{\mathcal{C}_j}^*) \leq 0$ *for τ-partition nice sampling, and* $\sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}}{q_{\mathcal{C}_j}}(n_{\mathcal{C}_j} h_{\mathcal{C}_j}^* - \overline{h}_{\mathcal{C}_j}^*) \leq 0$ *for $\tau-$partition independent sampling, where $e_{\mathcal{C}_k} = q_{\mathcal{C}_k}(n_{\mathcal{C}_k}-1)$ and $L_{\max}^{\mathcal{C}_r} = \max_{i \in \mathcal{C}_r} L_i$. Otherwise: $\tau(x^*) = 1$.*

Algorithm 1: SGD with Adaptive Batch size

```
Input: Smoothness constants $L$, $L_i$, strong convexity constant $\mu$, target neighborhood $\epsilon$,
Sampling Strategy $S$, initial point $x^0$, variance cap $C \geq 0$. Initialize: Set $k = 0$
while not converged do
    Set $\tau^k \leftarrow \tau(x^k)$,   $\mathcal{L}^k \leftarrow \mathcal{L}(\tau^k)$,   $\sigma^k \leftarrow \sigma(x^k, \tau^k)$,   $\gamma^k \leftarrow \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \frac{\epsilon\mu}{\min(C, 2\sigma^k)}\right\}$
    Sample $v_k$ from $S$ and Do SGD step: $x^{k+1} \leftarrow x^k - \gamma^k \nabla f_{v_k}(x^k)$
end while.     Output: $x^k$
```

3. Proposed Algorithm

The theoretical analysis gives us the optimal batch size for each of the proposed sampling techniques. However, we are unable to use these formulas directly since all of the expressions of optimal batch size depend on the knowledge of x^* through the values of $h_i^* \forall i \in [n]$. Our algorithm overcomes this problem by estimating the values of h_i^* at every iteration by h_i^k. Although this approach seems to be mathematically sound, it is costly because it requires passing through the whole training set every iteration. Alternatively, a more practical approach is to store $h_i^0 = h_i(x^0)\ \forall i \in [n]$, then set $h_i^k = \left\|\nabla f_i(x^k)\right\|^2\ for\ i \in \mathcal{S}_k$ and $h_i^k = h_i^{k-1}\ for\ i \notin \mathcal{S}_k$, where $\mathcal{S}_k$ is the set of indices considered in the k^{th} iteration. In addition to storing an extra n dimensional vector, this approach costs only computing the norms of the stochastic gradients that we already used in the SGD step. Both options lead to convergence in a similar number of epochs, so we let our proposed algorithm adopt the second (more practical) option of estimating h_i^*.

In our algorithm, for a given sampling technique, we use the current estimate of the model x^k to estimate the sub-optimal batch size $\tau^k := \tau(x^k)$ at the k^{th} iteration. Based on this estimate, we use Lemma 2 in calculating an estimate for both the expected smoothness $\mathcal{L}(\tau^k)$ and the noise gradient $\sigma(x^k, \tau^k)$ at that iteration. After that, we compute the step-size γ^k and finally conduct a SGD step. The summary can be found in Algorithm 1. For theoretical convergence purposes, we cap σ^k by a positive constant C, and we set the learning rate at each iteration to $\gamma^k \leftarrow \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \frac{\epsilon\mu}{\min\{C, 2\sigma^k\}}\right\}$. This way, learning rates generated by Algorithm 1 are bounded by positive constants $\gamma_{\max} = \frac{1}{2}\max_{\tau \in [n]}\left\{\frac{1}{\mathcal{L}(\tau)}\right\}$ and $\gamma_{\min} = \frac{1}{2}\min\left\{\min_{\tau \in [n]}\left\{\frac{1}{\mathcal{L}(\tau)}\right\}, \frac{\epsilon\mu}{C}\right\}$.

Theorem 4 *Assume f is $\mu-$strongly convex and assumptions 1, and 2 hold. Then the iterates of Algorithm 1 satisfy:* $\mathbb{E}\left\|x^k - x^*\right\|^2 \leq (1 - \gamma_{\min}\mu)^k \left\|x^0 - x^*\right\|^2 + R,$

where $R = \frac{2\gamma_{\max}^2 \sigma^*}{\gamma_{\min}\mu}$. Theorem 4 guarantees the convergence of the proposed algorithm. Although there is no significant theoretical improvement here compared to previous SGD results in the fixed batch and learning rate regimes, we measure the improvement to be significant in practice.

Convergence of τ^k to τ^*. The motivation behind the proposed algorithm is to learn the optimal batch size in an online fashion so that we get to $\epsilon-$neighborhood of the optimal model with the minimum number of epochs. For simplicity, let's assume that $\sigma^* = 0$. As $x^k \rightarrow x^*$, then $h_i^k = \nabla f_i(x^k) \rightarrow \nabla f_i(x^*) = h_i^*$, and thus $\tau^k \rightarrow \tau^*$. In Theorem 4, we showed the convergence of x^k to a neighborhood around x^*. Hence the theory predicts that our estimate of the optimal batch τ^k will converge to a neighborhood of the optimal batch size τ^*.

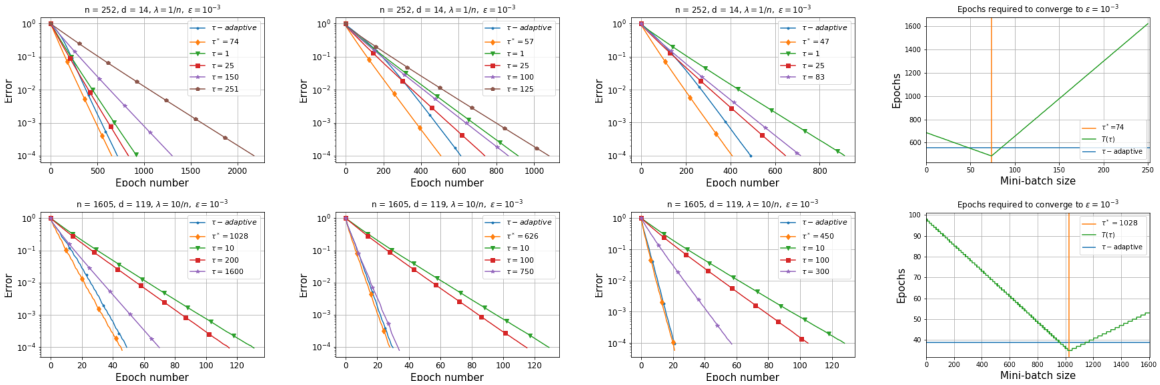

Figure 1: **Convergence of ridge and logistic regression** using $\tau-$partition nice sampling on *bodyfat* dataset (first row) and $\tau-$partition independent sampling on *a1a* dataset (second row).

4. Experiments

In this section, we compare our algorithm to fixed batch size SGD in terms of the number of epochs needed to reach a pre-specified neighborhood $\epsilon/10$. In the following results, we capture the convergence rate by recording the relative error $\left(\|x^k - x^*\|^2 / \|x^0 - x^*\|^2\right)$ where x^0 is drawn from a standard normal distribution $\mathcal{N}(0, \mathbf{I})$. We also report the number of training examples n, the dimension of the machine learning model d, regularization factor λ, and the target neighborhood ϵ above each figure. We consider the problems of regularized ridge and logistic regression where each f_i is strongly convex and L-smooth, and x^* can be known a-priori.Specifically, we want to $\min_{x \in \mathbb{R}^d} f(x)$ where

$$f_{\text{ridge}}(x) = \frac{1}{2n} \sum_{i=1}^{n} \left\| a_i^T x - b_i \right\|_2^2 + \frac{\lambda}{2} \|x\|_2^2, \;\; f_{\text{logistic}}(x) = \frac{1}{2n} \sum_{i=1}^{n} \log\left(1 + \exp\left(b_i a_i^T x\right)\right) + \frac{\lambda}{2} \|x\|_2^2$$

where $(a_i, b_i) \sim \mathcal{D}$ are pairs of data examples from the training set. For each of the considered problems, we performed experiments on real datasets from LIBSVM [3]. We tested our algorithm on ridge and logistic regression on *bodyfat* and *a1a* datasets in Figure 1. For these datasets, we considered $\tau-$partition independent and $\tau-$partition nice sampling with distributing the training set into one, two, and three partitions. Moreover, we take the previous experiments one step further by running a comparison of various fixed batch size SGD, as well as our adaptive method with a single partition (last column of 1). We plot the total iteration complexity for each batch size, and highlight optimal batch size obtained from our theoretical analysis, and how many epochs our adaptive algorithm needs to converge. This plot can be viewed as a summary of grid-search for optimal batch size (throughout all possible fixed batch sizes). Despite the fact that the optimal batch size is nontrivial and varies significantly with the model, dataset, sampling strategy, and number of partitions, our algorithm demonstrated consistent performance overall. In some cases, it was even able to cut down the number of epochs needed to reach the desired error to a factor of six.

The produced figures of our grid-search perfectly capture the tightness of our theoretical analysis. In particular, the total iteration complexity decreases linearly up to a neighborhood of τ^* and then increases linearly. In addition, Theorem 3 always captures the empirical minimum of $T(\tau)$ up to a negligible error. Moreover, these figures show how close T_{adaptive} is to the total iteration complexity using optimal batch size $T(\tau^*)$. Finally, in terms of running time, our algorithm requires 0.2322 ms per epoch, while running SGD with the optimal batch size requires 0.2298 ms.

5. Acknowledgement

This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research. The work of Motasem Alfarra and Bernard Ghanem was supported by Award No. OSR-CRG2019-4033.

References

[1] Jonathan Barzilai and Jonathan M Borwein. Two-point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[2] Léon Bottou. Large-scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[4] Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated Inference with Adaptive Batches. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1504–1513, 2017.

[5] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks. *arXiv preprint arXiv:1804.07612*, 2018.

[6] V John Mathews and Zhenhua Xie. A Stochastic Gradient Adaptive Filter with Gradient Adaptive Step Size. *IEEE transactions on Signal Processing*, 41(6):2075–2087, 1993.

[7] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust Stochastic Approximation Approach to Stochastic Programming. *SIAM Journal on Optimization*, 19 (4):1574–1609, 2009.

[8] Xun Qian, Peter Richtárik, Robert M. Gower, Alibek Sailanbayev, Nicolas Loizou, and Egor Shulgin. SGD with arbitrary sampling: General analysis and improved rates. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15*, 2019.

[9] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[10] MA Schumer and Kenneth Steiglitz. Adaptive Step Size Random Search. *IEEE Transactions on Automatic Control*, 13(3):270–276, 1968.

[11] Sungho Shin, Yoonho Boo, and Wonyong Sung. Fixed-Point Optimization of Deep Neural Networks with Adaptive Step Size Retraining. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1203–1207. IEEE, 2017.

[12] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2019.

[13] Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-Borwein Step Size for Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 685–693, 2016.

[14] Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. *arXiv preprint arXiv:1708.03888*, 2017.

[15] Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD Batch Size to 32k for Imagenet Training. *arXiv preprint arXiv:1708.03888*, 6, 2017.

[16] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix A. Proof of Lemma 2

For the considered partition sampling, the indices $1, \ldots, n$ are distributed into the sets $\mathcal{C}_1, \ldots, \mathcal{C}_K$ with each having a minimum cardinality of τ. We choose each set $\mathcal{C}_j$ with probability $q_{\mathcal{C}_j}$ where $\sum\limits_j q_{\mathcal{C}_j} = 1$. Note that

$$\mathbf{P}_{ij} = \begin{cases} 0 & \text{if } i \in C_k, j \in C_l, k \neq l \\ q_k \frac{\tau(\tau-1)}{n_k(n_k-1)} & \text{if } i \neq j, i, j \in C_k, |C_k| = \tau_k \\ q_k \frac{\tau}{n_k} & \text{if i=j} \end{cases} .$$

Therefore

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(y)\|^2\right] &= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i,j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_i p_j}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_i p_j}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_i(x) - \nabla f_i(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{n_{\mathcal{C}_k}}{q_{\mathcal{C}_k}\tau}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_i(x) - \nabla f_i(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}\sum_{i\neq j\in\mathcal{C}_k}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}}{q_{\mathcal{C}_k}\tau}\sum_{i\in\mathcal{C}_k}\|\nabla f_i(x) - \nabla f_i(y)\|^2 \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}\left\|\sum_{i\in\mathcal{C}_k}\nabla f_i(x) - \nabla f_i(y)\right\|^2 \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}\sum_{i\in\mathcal{C}_k}\|\nabla f_i(x) - \nabla f_i(y)\|^2 \\
&\leq \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^3_{\mathcal{C}_k}(\tau-1)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}2L_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}\sum_{i\in\mathcal{C}_k}2L_i D_{f_i}(x,y) \\
&\leq \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^3_{\mathcal{C}_k}(\tau-1)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}2L_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^2_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}2\max_{i\in\mathcal{C}_k}L_i D_{f_{\mathcal{C}_k}}(x,y) \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}2\frac{n^2_{\mathcal{C}_k}(\tau-1)L_{\mathcal{C}_k} + n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)\max_{i\in\mathcal{C}_k}L_i}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}n_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\leq 2\frac{1}{n}(\max_{\mathcal{C}_k}\frac{n^2_{\mathcal{C}_k}(\tau-1)L_{\mathcal{C}_k} + n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)\max_{i\in\mathcal{C}_k}L_i}{q_{\mathcal{C}_k}\tau(n_{\mathcal{C}_k}-1)}n_{\mathcal{C}_k})D_f(x,y),
\end{aligned}$$

where $D_f(x,y) = f(x) - f(y) - \langle \nabla f(y), x-y \rangle$. Setting $y \leftarrow x^*$, leads to the desired upper bound of the expected smoothness which is given by

$$\mathcal{L}(\tau) = \frac{1}{n\tau}\left(\max_{\mathcal{C}_k} \frac{n_{\mathcal{C}_k}}{q_{\mathcal{C}_k}\left(n_{\mathcal{C}_k}-1\right)}\left(n_{\mathcal{C}_k}^2(\tau-1)L_{\mathcal{C}_k} + n_{\mathcal{C}_k}\left(n_{\mathcal{C}_k}-\tau\right)\max_{i\in\mathcal{C}_k} L_i\right)\right).$$

Next, we derive a similar bound for $\tau-$independent partition sampling.

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(y)\|^2\right] &= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i,j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_i p_j}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_i p_j}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_i(x) - \nabla f_i(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k} p_i}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_i(x) - \nabla f_i(y)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\neq j\in\mathcal{C}_k}\Big\langle \nabla f_i(x) - \nabla f_i(y), \nabla f_j(x) - \nabla f_j(y)\Big\rangle \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}\|\nabla f_i(x) - \nabla f_i(y)\|^2 \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\left\|\sum_{i\in\mathcal{C}_k}\nabla f_i(x) - \nabla f_i(y)\right\|^2 \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}\frac{1-p_i}{p_i}\|\nabla f_i(x) - \nabla f_i(y)\|^2 \\
&\leq \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}^2}{q_{\mathcal{C}_k}}2L_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}\frac{1-p_i}{p_i}2L_i D_{f_i}(x,y) \\
&\leq \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}^2}{q_{\mathcal{C}_k}}2L_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\quad + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}}{q_{\mathcal{C}_k}}2\max_{i\in\mathcal{C}_k}\frac{1-p_i}{p_i}L_i D_{f_{\mathcal{C}_k}}(x,y) \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}2\Big(\frac{n_{\mathcal{C}_k}L_{\mathcal{C}_k}}{q_{\mathcal{C}_k}} + \max_{i\in\mathcal{C}_k}\frac{(1-p_i)L_i}{q_{\mathcal{C}_k}p_i}\Big)n_{\mathcal{C}_k}D_{f_{\mathcal{C}_k}}(x,y) \\
&\leq 2\frac{1}{n}\max_{i\in\mathcal{C}_k}\Big(\frac{n_{\mathcal{C}_k}L_{\mathcal{C}_k}}{q_{\mathcal{C}_k}} + \max_{i\in\mathcal{C}_k}\frac{(1-p_i)L_i}{q_{\mathcal{C}_k}p_i}\Big)D_f(x,y).
\end{aligned}$$

This gives the desired upper bound for the expected smoothness

$$\mathcal{L}(\tau) = \frac{1}{n}\max_{i\in\mathcal{C}_k}\left(\frac{n_{\mathcal{C}_k}L_{\mathcal{C}_k}}{q_{\mathcal{C}_k}} + \max_{i\in\mathcal{C}_k}\frac{(1-p_i)L_i}{q_{\mathcal{C}_k}p_i}\right).$$

Following the same notation, we move on to compute σ for each sampling. First, for $\tau-$nice partition sampling we have

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_v(x^*)\|^2\right] &= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i,j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_ip_j}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_ip_j}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}\Big\langle\nabla f_i(x^*),\nabla f_i(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{n_{\mathcal{C}_k}}{\tau q_{\mathcal{C}_k}}\Big\langle\nabla f_i(x^*),\nabla f_i(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}\sum_{i\neq j\in\mathcal{C}_k}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}}{\tau q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}h_i \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(\tau-1)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}\left\|\sum_{i\in\mathcal{C}_k}\nabla f_i(x^*)\right\|^2 + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}h_i \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^3_{\mathcal{C}_k}(\tau-1)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}h_{\mathcal{C}_k} + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^2_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)}{\tau(n_{\mathcal{C}_k}-1)q_{\mathcal{C}_k}}\bar{h}_{\mathcal{C}_k}.
\end{aligned}$$

Where its left to rearrange the terms to get the first result of the lemma. Next, we compute σ for $\tau-$independent partition:

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_v(x^*)\|^2\right] &= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i,j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_ip_j}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{\mathbf{P}_{ij}}{p_ip_j}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}\Big\langle\nabla f_i(x^*),\nabla f_i(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\neq j\in\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}p_i}\Big\langle\nabla f_i(x^*),\nabla f_i(x^*)\Big\rangle \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\neq j\in\mathcal{C}_k}\Big\langle\nabla f_i(x^*),\nabla f_j(x^*)\Big\rangle + \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\sum_{i\in\mathcal{C}_k}\frac{1}{p_i}h_i \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{1}{q_{\mathcal{C}_k}}\left\|\sum_{i\in\mathcal{C}_k}\nabla f_i(x^*)\right\|^2 + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{(1-p_i)h_i}{q_{\mathcal{C}_k}p_i} \\
&= \frac{1}{n^2}\sum_{\mathcal{C}_k}\frac{n^2_{\mathcal{C}_k}}{q_{\mathcal{C}_k}}h_{\mathcal{C}_k} + \frac{1}{n^2}\sum_{\mathcal{C}_k}\sum_{i\in\mathcal{C}_k}\frac{(1-p_i)h_i}{q_{\mathcal{C}_k}p_i}.
\end{aligned}$$

Appendix B. Proof of Theorem 3

Recall that the optimal batch size $\tau(x^*)$ is chosen such that the quantity $\max\left\{\tau\mathcal{L}(\tau), \frac{2}{\epsilon\mu}\tau\sigma(x^*,\tau)\right\}$ is minimized. Note that in both $\tau-$nice partition, and $\tau-$ independent partition with ($p_i = \frac{\tau}{n_{\mathcal{C}_j}}$), $\tau\sigma(x^*,\tau)$ is a linear function of τ while $\tau\mathcal{L}(\tau)$ is a max across linearly increasing functions of τ. To find the minimized in such a case, we leverage the following lemma.

Lemma 5 *Suppose that $l_1(x), l_2(x), ..., l_k(x)$ are increasing linear functions of x, and $r(x)$ is linear decreasing function of x, then the minimizer of $\max(l_1(x), l_2(x), ..., l_k(x), r(x))$ is $x^* = \min_i(x_i)$ where x_i is the unique solution for $l_i(x) = r(x)$*

Proof: Let x^* be defined as above, and let x be an arbitrary number. If $x \leq x^*$, then $r(x) \geq r(x^*) \geq r(x_i) = l_i(x_i) \geq l_i(x^*)$ for each i which means $r(x) \geq \max(l_1(x^*), ..., l_k(x^*), r(x^*))$. On the other hand, if $x \geq x^*$, then let i be the index s.t. $x_i = x^*$. We have $l_i(x) \geq l_i(x^*) = r(x^*) \geq r(x_j) = l_j(x_j) \geq l_j(x^*)$, hence $l_i(x) \geq \max(l_1(x^*), ..., l_k(x^*), r(x^*))$. This means that $x^* = \min_i(x_i)$ is indeed the minimizer of $\max(l_1(x), l_2(x), ..., l_k(x), r(x))$.

Now we can estimate optimal batch sizes for proposed samplings.

$\tau-$**nice partition:** if $\sum_{\mathcal{C}_j} \frac{n_{\mathcal{C}_j}^2}{e_j}(h_{\mathcal{C}_j}^* n_{\mathcal{C}_j} - \overline{h}_{\mathcal{C}_j}^*) \leq 0$ then $\tau\sigma(\tau)$ is a decreasing linear function of τ, and $\tau\mathcal{L}(\tau)$ is the max of increasing linear functions. Therefore, we can leverage the previous lemma with $r(\tau) = \frac{2}{\epsilon\mu}\tau\sigma(x^*,\tau)$ and $l_{\mathcal{C}_k}(\tau) = \frac{n_{\mathcal{C}_k}^2(\tau-1)L_{\mathcal{C}_k} + n_{\mathcal{C}_k}(n_{\mathcal{C}_k}-\tau)\max_{i\in\mathcal{C}_k} L_i}{q_{\mathcal{C}_k}(n_{\mathcal{C}_k}-1)} n_{\mathcal{C}_k}$ to find the optimal batch size as $\tau^* = \min_{\mathcal{C}_k}(\tau_{\mathcal{C}_k}^*)$, where

$$\tau_{\mathcal{C}_k}^* = \frac{\frac{nn_{\mathcal{C}_r}^2}{e_{\mathcal{C}_r}}(L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r}) + \frac{2}{\epsilon\mu}\sum_{\mathcal{C}_j}\frac{n_{\mathcal{C}_j}^3}{e_{\mathcal{C}_j}}\left(\overline{h}_{\mathcal{C}_j} - h_{\mathcal{C}_j}\right)}{\frac{nn_{\mathcal{C}_r}}{e_{\mathcal{C}_r}}(n_{\mathcal{C}_r}L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r}) + \frac{2}{\epsilon\mu}\sum_{\mathcal{C}_j}\frac{n_{\mathcal{C}_j}^2}{e_{\mathcal{C}_j}}(\overline{h}_{\mathcal{C}_j} - n_{\mathcal{C}_j}h_{\mathcal{C}_j})}.$$

$\tau-$**independent partition:** Similar to $\tau-$nice partition, we have $\tau\sigma(\tau)$ is a decreasing linear function of τ if $\sum_{\mathcal{C}_j}\frac{n_{\mathcal{C}_j}}{q_{\mathcal{C}_j}}(n_{\mathcal{C}_j}h_{\mathcal{C}_j}^* - \overline{h}_{\mathcal{C}_j}^*) \leq 0$, and $\tau\mathcal{L}(\tau)$ is the max of increasing linear functions of τ. Hence, we can leverage the previous lemma with $r(\tau) = \frac{2}{\epsilon\mu}\tau\sigma(x^*,\tau)$ and $l_{\mathcal{C}_k}(\tau) = \frac{n_{\mathcal{C}_k}L_{\mathcal{C}_k}\tau}{q_{\mathcal{C}_k}} + (n_{\mathcal{C}_k} - \tau)\max_{i\in\mathcal{C}_k}\frac{L_i}{q_{\mathcal{C}_k}}$ to find the optimal batch size as $\tau^* = \min_{\mathcal{C}_k}(\tau_{\mathcal{C}_k}^*)$, where

$$\tau_{\mathcal{C}_k}^* = \frac{\frac{2}{\epsilon\mu}\sum_{\mathcal{C}_j}\frac{n_{\mathcal{C}_j}^2}{q_{\mathcal{C}_j}}\overline{h}_{\mathcal{C}_j} - \frac{n}{q_{\mathcal{C}_r}}L_{\max}^{\mathcal{C}_r}}{\frac{2}{\epsilon\mu}\sum_{\mathcal{C}_j}\frac{n_{\mathcal{C}_j}}{q_{\mathcal{C}_j}}(\overline{h}_{\mathcal{C}_j} - n_{\mathcal{C}_j}h_{\mathcal{C}_j}) + \frac{n}{q_{\mathcal{C}_r}}(n_{\mathcal{C}_r}L_{\mathcal{C}_r} - L_{\max}^{\mathcal{C}_r})}.$$

Appendix C. Proof of bounds on step sizes

For our choice of the learning rate we have

$$\gamma^k = \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \frac{\epsilon\mu}{\min(C, 2\sigma^k)}\right\} = \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \max\left\{\frac{\epsilon\mu}{C}, \frac{\epsilon\mu}{2\sigma^k}\right\}\right\} \geq \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \frac{\epsilon\mu}{C}\right\}.$$

Since $\mathcal{L}^k$ is a linear combination between the smoothness constants of the functions f_i, then it is bounded. Therefore, both $\mathcal{L}^k$ and C are upper bounded and lower bounded as well as $\frac{1}{\mathcal{L}^k}$ and $\frac{\epsilon\mu}{C}$, thus also γ^k is bounded by positive constants $\gamma_{\max} = \frac{1}{2}\max_{\tau\in[n]}\left\{\frac{1}{\mathcal{L}(\tau)}\right\}$ and $\gamma_{\min} = \frac{1}{2}\min\left\{\min_{\tau\in[n]}\left\{\frac{1}{\mathcal{L}(\tau)}\right\}, \frac{\epsilon\mu}{C}\right\}$.

Appendix D. Proof of Theorem 4

Let $r^k = \left\|x^k - x^*\right\|^2$, then

$$\begin{aligned}
\mathbb{E}\left[r^{k+1}|x^k\right] &= \mathbb{E}\left[\left\|x^k - \gamma^k\nabla f_{v_k}(x^k) - x^*\right\|^2|x^k\right] \\
&= r^k + (\gamma^k)^2\mathbb{E}\left[\left\|\nabla f_{v_k}(x^k)\right\|^2|x^k\right] - 2\gamma^k\langle\mathbb{E}\left[\nabla f_{v_k}(x^k)|x^k\right], r^k\rangle \\
&= r^k + (\gamma^k)^2\mathbb{E}\left[\left\|\nabla f_{v_k}(x^k)\right\|^2|x^k\right] - 2\gamma^k\left(f(x^k) - f(x^*) + \frac{\mu}{2}r^k\right) \\
&= (1-\gamma^k\mu)r^k + (\gamma^k)^2\mathbb{E}\left[\left\|\nabla f_{v_k}(x^k)\right\|^2|x^k\right] - 2\gamma^k(f(x^k) - f(x^*)) \\
&\leq (1-\gamma^k\mu)r^k + (\gamma^k)^2(4\mathcal{L}^k(f(x^k) - f(x^*)) + 2\sigma) - 2\gamma^k(f(x^k) - f(x^*)) \\
&= (1-\gamma^k\mu)r^k - 2\gamma^k((1-2\gamma^k\mathcal{L}^k)(f(x^k) - f(x^*))) + 2(\gamma^k)^2\sigma \\
&\leq (1-\gamma^k\mu)r^k + 2(\gamma^k)^2\sigma \quad \text{for } \gamma_k \leq \frac{1}{2\mathcal{L}^k}.
\end{aligned}$$

From Eariler bounds, there exist upper and lower bounds for step-sizes, $\gamma_{\min} \leq \gamma^k \leq \gamma_{\max}$, thus

$$\mathbb{E}\left[r^{k+1}|x^k\right] \leq (1-\gamma^k\mu)r^k + 2(\gamma^k)^2\sigma \leq (1-\gamma_{\min}\mu)r^k + 2\gamma_{\max}^2\sigma.$$

Therefore, unrolling the above recursion gives

$$\begin{aligned}
\mathbb{E}\left[r^{k+1}|x^k\right] &\leq (1-\gamma_{\min}\mu)^k r^0 + 2\gamma_{\max}^2\sigma\sum_{i=0}^{k}(1-\gamma_{\min}\mu)^k \\
&\leq (1-\gamma_{\min}\mu)^k r^0 + \frac{2\gamma_{\max}^2\sigma}{\gamma_{\min}\mu}.
\end{aligned}$$

Appendix E. Proof of convergence to linear neighborhood in ϵ

In this section, we prove that our algorithm converges to a neighborhood around the optimal solution with size upper bounded by an expression linear in ϵ. First of all, we prove that $\sigma(x)$ is lower bounded by a multiple of the variance in the optimum σ^* (in Lemma 6). Then, by showing an alternative upper bound on the step-size, we obtain an upper bound for neighborhood size R as expression of ϵ.

Lemma 6 *Suppose f is $\mu-$strongly convex, $L-$smooth and with expected smoothness constant $\mathcal{L}$. Let v be as in the SGD overview, i.e., $\mathbb{E}[v_i = 1]$ for all i. Fix any $c > 0$. The function $\sigma(x) =$*

$\mathbb{E}\left[\|\nabla f_v(x)\|^2\right]$ *can be lower bounded as follows:*

$$\sigma(x) \geq \left\{\mu^2 c, 1 - 2\sqrt{\mathcal{L}Lc}\right\} \sigma(x^*), \qquad \forall x \in \mathbb{R}^d.$$

The constant c maximizing this bound is $c = \left(\frac{\sqrt{\mathcal{L}L+\mu^2}-\sqrt{\mathcal{L}L}}{\mu^2}\right)^2$*, giving the bound*

$$\sigma(x) \geq \frac{\left(\sqrt{\mathcal{L}L+\mu^2}-\sqrt{\mathcal{L}L}\right)^2}{\mu^2}\sigma(x^*), \qquad \forall x \in \mathbb{R}^d.$$

Proof: Choose $c > 0$. If $\|x - x^*\| \geq \sqrt{c\sigma(x^*)}$, then using Jensen's inequality and strong convexity of f, we get

$$\sigma(x) \geq \|\mathbb{E}\left[\nabla f_v(x)\right]\|^2 = \|\nabla f(x)\|^2 = \|\nabla f(x) - \nabla f(x^*)\|^2 \geq \mu^2 \|x - x^*\|^2 \geq \mu^2 c\sigma(x^*). \tag{3}$$

If $\|x - x^*\| \leq \sqrt{c\sigma(x^*)}$, then using expected smoothness and L-smoothness, we get

$$\mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|^2\right] \overset{(1)}{\leq} 2\mathcal{L}(f(x) - f(x^*)) \leq \mathcal{L}L\|x - x^*\|^2 \leq \mathcal{L}Lc\sigma(x^*). \tag{4}$$

Further, we can write

$$\begin{aligned}
\sigma(x^*) - \sigma(x) &= \mathbb{E}\left[\|\nabla f_v(x^*)\|^2\right] - \mathbb{E}\left[\|\nabla f_v(x)\|^2\right] \\
&= -2\mathbb{E}\left[\langle \nabla f_v(x) - \nabla f_v(x^*), \nabla f_v(x^*)\rangle\right] - \mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|^2\right] \\
&\leq -2\mathbb{E}\left[\langle \nabla f_v(x) - \nabla f_v(x^*), \nabla f_v(x^*)\rangle\right] \\
&\leq 2\mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\| \|\nabla f_v(x^*)\|\right] \\
&\leq 2\sqrt{\mathbb{E}\left[\|\nabla f_v(x) - \nabla f_v(x^*)\|^2\right]}\sqrt{\mathbb{E}\left[\|\nabla f_v(x^*)\|^2\right]} \\
&\overset{(4)}{\leq} 2\sqrt{\mathcal{L}Lc}\sqrt{\sigma(x^*)}\sqrt{\sigma(x^*)} \\
&= 2\sqrt{\mathcal{L}Lc}\sigma(x^*),
\end{aligned}$$

where the first inequality follows by neglecting a negative term, the second by Cauchy-Schwarz and the third by Hölder inequality for bounding the expectation of the product of two random variables. The last inequality implies that

$$\sigma(x) \geq \left(1 - 2\sqrt{\mathcal{L}Lc}\right)\sigma(x^*). \tag{5}$$

By combining the bounds (3) and (5), we get

$$\sigma(x) \geq \min\left\{\mu^2 c, 1 - 2\sqrt{\mathcal{L}Lc}\right\}\sigma(x^*).$$

Using Lemma 6, we can upper bound step-sizes γ^k. Assume that $\sigma^* = \sigma(x^*) > 0$. Let $\gamma_{\max}' = \frac{\epsilon\mu}{2}\max\left\{\frac{1}{C}, \frac{1}{2\eta\sigma^*}\right\}$, where $\eta = \frac{\left(\sqrt{\mathcal{L}L+\mu^2}-\sqrt{\mathcal{L}L}\right)^2}{\mu^2}$.

We have

$$\gamma^k = \frac{1}{2}\min\left\{\frac{1}{\mathcal{L}^k}, \frac{\epsilon\mu}{\min(C, 2\sigma^k)}\right\} \leq \frac{\epsilon\mu}{2}\max\left\{\frac{1}{C}, \frac{1}{2\sigma^k}\right\} \leq \frac{\epsilon\mu}{2}\max\left\{\frac{1}{C}, \frac{1}{2\eta\sigma^*}\right\} = \gamma^{'}_{\max}.$$

Now, we use this alternative step-sizes upper bound to obtain alternative expression for residual term $R = \frac{2\gamma_{\max}^2\sigma^*}{\gamma_{\min}\mu}$ in Theorem 4 (let's denote it $R^{'}$). Analogically to proof of Theorem 4 (with upper bound of step-sizes $\gamma^{'}_{\max}$), we have $R^{'} = \frac{2\gamma^{'2}_{\max}\sigma^*}{\gamma_{\min}\mu}$.

Finally, expanding expression for residual term $R^{'}$ yields the result:

$$R^{'} = \frac{2\gamma^{'2}_{\max}\sigma^*}{\gamma_{\min}\mu} = \frac{2\left(\frac{\epsilon\mu}{2}\max\left\{\frac{1}{C}, \frac{1}{2\eta\sigma^*}\right\}\right)^2\sigma^*}{\frac{1}{2}\min\left\{\min_{\tau\in[n]}\left\{\frac{1}{\mathcal{L}(\tau)}\right\}, \frac{\epsilon\mu}{C}\right\}\mu} = \epsilon^2\mu\left(\max\left\{\frac{1}{C}, \frac{1}{2\eta\sigma^*}\right\}\right)^2 \max\left\{\max_{\tau\in[n]}\left\{\mathcal{L}(\tau)\right\}, \frac{C}{\mu\epsilon}\right\}\sigma^*$$

$$= \epsilon\mu\left(\max\left\{\frac{1}{C}, \frac{1}{2\eta\sigma^*}\right\}\right)^2 \max\left\{\epsilon\max_{\tau\in[n]}\left\{\mathcal{L}(\tau)\right\}, \frac{C}{\mu}\right\}\sigma^*.$$

As conclusion, if we consider $R^{'}$ to be function of ϵ, then it is at least linear in ϵ, $R^{'} \in \mathcal{O}(\epsilon)$.

Appendix F. Additional Experimental Results

In this section, we present additional experimental results. Here we test each dataset on the sampling that was not tested on. Similar to the earlier result, the proposed algorithm outperforms most of the fixed batch size SGD. Moreover, it can be seen as a first glance, that the optimal batch is non-trivial, and it is changing as we partition the dataset through different number of partitions. For example, in *bodyfat* dataset that is located in one partition, the optimal batch size was $\tau^* = 74$. Although one can still sample $\tau = 74$ when the data is divided into two partitions, the optimal has changed to $\tau^* = 57$. This is clearly shown in Figure 4 where it shows that the optimal batch size varies between different partitioning, and the predicted optimal from our theoretical analysis matches the actual optimal.

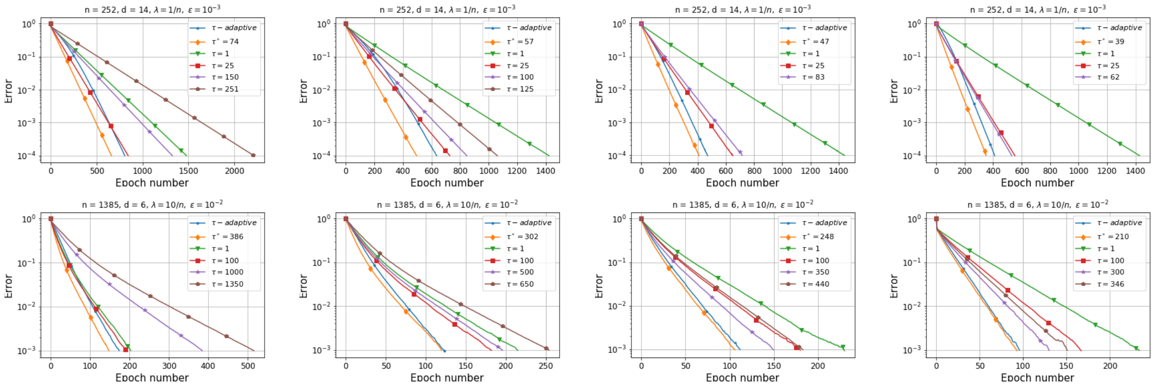

Figure 2: **Convergence of Ridge regression** using $\tau-$partition independent sampling on *Bodyfat* dataset (first row) and $\tau-$partition nice sampling on *mg* dataset (second row). In first three columns, training set is distributed among 1, 2, 3 and 4 partitions, respectively.

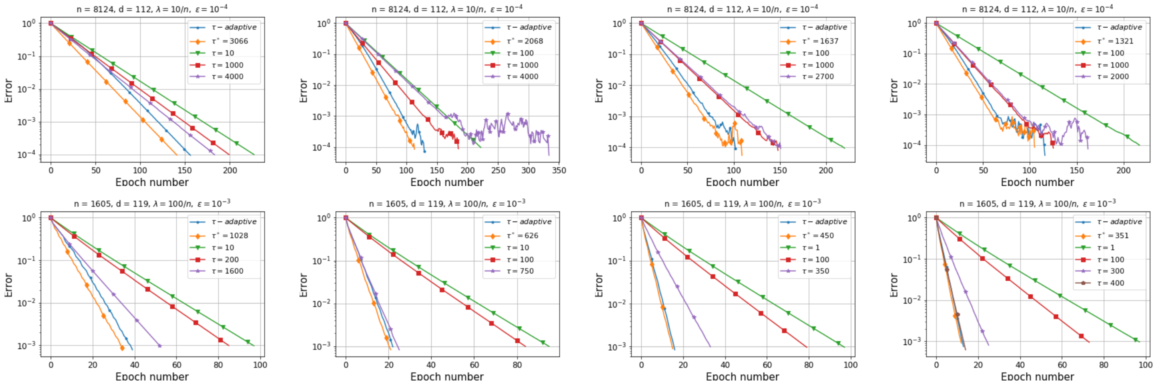

Figure 3: **Convergence of Logistic regression** using $\tau-$partition independent sampling on *mushrooms* dataset (first row) and $\tau-$partition nice sampling on *a1a* dataset (second row). In first three columns, training set is distributed among 1, 2, 3 and 4 partitions, respectively.

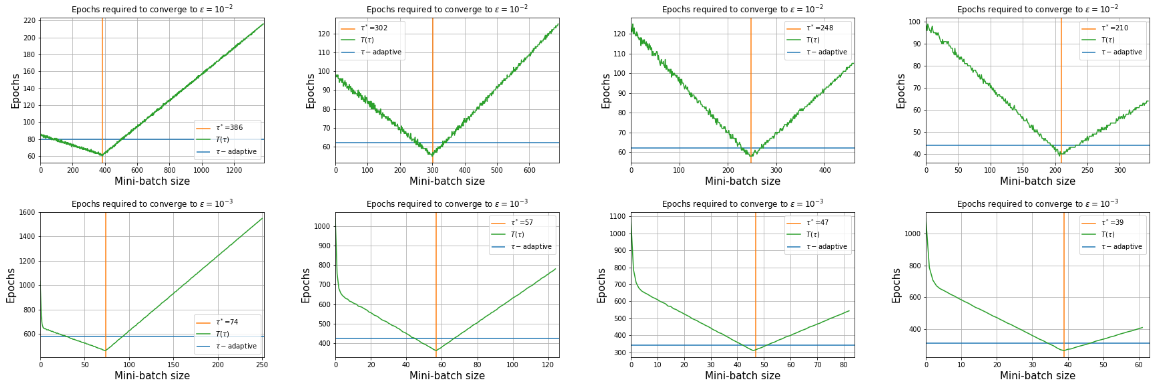

Figure 4: **Effect of batch size on the total iteration complexity**. First row: *mg* dataset sampled using $\tau-$nice partition sampling. Second row: *bodyfat* dataset sampled using $\tau-$independent partition sampling. From left to right: dataset is distributed across 1, 2, 3, and 4 partitions. This figure reflects the tightness of the theoretical result in relating the total iteration complexity with the batch size, and the optimal batch size. Moreover, This figure shows how close the proposed algorithm is to the optimal batch size in terms of the performance.