Journal of Machine Learning Research 19 (2018) 1-21 Submitted 5/16; Revised 1/18; Published 8/18

Importance Sampling for Minibatches

Dominik Csiba CDOMINIK@GMAIL.COM
School of Mathematics
University of Edinburgh
Edinburgh, United Kingdom

Peter Richtárik[*] PETER.RICHTARIK@ED.AC.UK
School of Mathematics
University of Edinburgh
Edinburgh, United Kingdom

Editor: Benjamin Recht

Abstract

Minibatching is a very well studied and highly popular technique in supervised learning, used by practitioners due to its ability to accelerate training through better utilization of parallel processing power and reduction of stochastic variance. Another popular technique is importance sampling—a strategy for preferential sampling of more important examples also capable of accelerating the training process. However, despite considerable effort by the community in these areas, and due to the inherent technical difficulty of the problem, there is virtually no existing work combining the power of importance sampling with the strength of minibatching. In this paper we propose the first practical *importance sampling for minibatches* and give simple and rigorous complexity analysis of its performance. We illustrate on synthetic problems that for training data of certain properties, our sampling can lead to several orders of magnitude improvement in training time. We then test the new sampling on several popular data sets, and show that the improvement can reach an order of magnitude.
keywords: empirical risk minimization; importance sampling; minibatching; variance-reduced methods; convex optimization

1. Introduction

Supervised learning is a widely adopted learning paradigm with important applications such as regression, classification and prediction. The most popular approach to training supervised learning models is via empirical risk minimization (ERM). In ERM, the practitioner collects data composed of example-label pairs, and seeks to identify the best predictor by minimizing the empirical risk, i.e., the average risk associated with the predictor over the training data.

With ever increasing demand for accuracy of the predictors, largely due to successful industrial applications, and with ever more sophisticated models that need to be trained, such as deep neural networks Hinton (2007); Krizhevsky et al. (2012), or multiclass classifi-

[*]. This author would like to acknowledge support from the EPSRC Grant EP/K02325X/1, *Accelerated Coordinate Descent Methods for Big Data Optimization* and the EPSRC Fellowship EP/N005538/1, *Randomized Algorithms for Extreme Convex Optimization.*

cation Huang et al. (2012), increasing volumes of data are used in the training phase. This leads to huge and hence extremely computationally intensive ERM problems.

Batch algorithms—methods that need to look at all the data before taking a single step to update the predictor—have long been known to be prohibitively impractical to use. Typical examples of batch methods are gradient descent and classical quasi-Newton methods. One of the most popular algorithms for overcoming the deluge-of-data issue is stochastic gradient descent (SGD), which can be traced back to a seminal work of Robbins and Monro (1951). In SGD, a single random example is selected in each iteration, and the predictor is updated using the information obtained by computing the gradient of the loss function associated with this example. This leads to a much more fine-grained iterative process, but at the same time introduces considerable stochastic noise, which eventually—typically after one or a few passes over the data—effectively halts the progress of the method, rendering it unable to push the training error (empirical risk) to the realm of small values.

1.1 Strategies for dealing with stochastic noise

Several approaches have been proposed to deal with the issue of stochastic noise in the finite-data regime. The most important of these are i) decreasing stepsizes, ii) minibatching, iii) importance sampling and iv) variance reduction via "shift", listed here from historically first to the most modern.

The first strategy, *decreasing stepsizes*, takes care of the noise issue by a gradual and direct scale-down process, which ensures that SGD converges to the ERM optimum Zhang (2004). However, an unwelcome side effect of this is a considerable slowdown of the iterative process Bottou (2010). For instance, the convergence rate is sublinear even if the function to be minimized is strongly convex.

The second strategy, *minibatching*, deals with the noise by utilizing a random set of examples in the estimate of the gradient, which effectively decreases the variance of the estimate Shalev-Shwartz et al. (2011). However, this has the unwelcome side-effect of requiring more computation. On the other hand, if a parallel processing machine is available, the computation can be done concurrently, which ultimately leads to speedup. This strategy does not result in an improvement of the convergence rate (unless progressively larger minibatch sizes are used, at the cost of further computational burden Friedlander and Schmidt (2012)), but can lead to massive improvement of the leading constant, which ultimately means acceleration (almost linear speedup for sparse data) Takáč et al. (2013).

The third strategy, *importance sampling*, operates by a careful data-driven design of the probabilities of selecting examples in the iterative process, leading to a reduction of the variance of the stochastic gradient thus selected. Typically, the overhead associated with computing the sampling probabilities and with sampling from the resulting distribution is negligible, and hence the net effect is speedup. In terms of theory, for standard SGD this improves a non-dominant term in the complexity. On the other hand, when SGD is combined with variance reduction, then this strategy leads to the improvement of the leading constant in the complexity estimate, typically via replacing the maximum of certain data-dependent quantities by their average Richtárik and Takáč (2016b); Konečný et al. (2017); Zhao and Zhang (2015); Qu et al. (2015); Needell et al. (2014); Csiba and Richtárik (2015); Csiba et al. (2015).

Finally, and most recently, there has been a considerable amount of research activity due to the ground-breaking realization that one can gain the benefits of SGD (cheap iterations) without having to pay through the side effects mentioned above (e.g., halt in convergence due to decreasing stepsizes or increase of workload due to the use of minibatches) in the finite data regime. The result, in theory, is that for strongly convex losses (for example), one does not have to suffer sublinear convergence any more, but instead a fast linear rate "kicks in". In practice, these methods dramatically surpass all previous existing approaches.

The main algorithmic idea is to change the search direction itself, via a properly designed and cheaply maintainable *"variance-reducing shift"* (control variate). Methods in this category are of two types: those operating in the primal space (i.e., directly on ERM) and those operating in a dual space (i.e., with the dual of the ERM problem). Methods of the primal variety include SAG Schmidt et al. (2013), SVRG Johnson and Zhang (2013), S2GD Konečný and Richtárik (2017), proxSVRG Xiao and Zhang (2014), SAGA Defazio et al. (2014), mS2GD Konečný et al. (2016) and MISO Mairal (2015). Methods of the dual variety work by updating randomly selected dual variables, which correspond to examples. These methods include SCD Shalev-Shwartz and Tewari (2011), RCDM Nesterov (2012); Richtárik and Takáč (2014), SDCA Shalev-Shwartz and Zhang (2013b), Hydra Richtárik and Takáč (2016a); Fercoq et al. (2014), mSDCA Takáč et al. (2013), APCG Lin et al. (2015), AsySPDC Liu and Wright (2015), RCD Necoara and Patrascu (2014), APPROX Fercoq and Richtárik (2015), SPDC Zhang and Xiao (2015), ProxSDCA Shalev-Shwartz and Zhang (2012), ASDCA Shalev-Shwartz and Zhang (2013a), IProx-SDCA Zhao and Zhang (2015), and QUARTZ Qu et al. (2015).

1.2 Combining strategies

We wish to stress that the key strategies, mini-batching, importance sampling and variance-reducing shift, should be seen as orthogonal tricks, and as such they can be combined, achieving an amplification effect. For instance, the first primal variance-reduced method allowing for mini-batching was Konečný et al. (2016); while dual-based methods in this category include Shalev-Shwartz and Zhang (2013a); Qu et al. (2015); Csiba and Richtárik (2015). Variance-reduced methods with importance sampling include Nesterov (2012); Richtárik and Takáč (2014); Richtárik and Takáč (2016b); Qu and Richtárik (2016) for general convex minimization problems, and Zhao and Zhang (2015); Qu et al. (2015); Needell et al. (2014); Csiba and Richtárik (2015) for ERM.

2. Contributions

Despite considerable effort of the machine learning and optimization research communities, virtually no importance sampling for minibatches was previously proposed, nor analyzed.[1]. The reason for this lies in the underlying theoretical and computational difficulties associated with the design and successful implementation of such a sampling. One needs to come up with a way to focus on a reasonable set of subsets (minibatches) of the examples to be used in each iteration (issue: there are many subsets; which ones to choose?), as-

1. A brief note in Richtárik and Takáč (2016b) is an exception, but the sampling is different from ours, was not implemented nor tested, leads to the necessity to solve a linear program and hence is impractical. Another exception is Harikandeh et al. (2015).

sign meaningful data-dependent non-uniform probabilities to them (issue: how?), and then be able to sample these subsets according to the chosen distribution (issue: this could be computationally expensive).

The tools that would enable one to consider these questions did not exist until recently. However, due to a recent line of work on analyzing variance-reduced methods utilizing what is known as *arbitrary sampling* Richtárik and Takáč (2016b); Qu et al. (2015); Qu and Richtárik (2016); Qu and Richtárik (2016); Csiba and Richtárik (2015), we are able to ask these questions and provide answers. In this work we design a novel family of samplings—*bucket samplings*—and a particular member of this family—*importance sampling for minibatches*. We illustrate the power of this sampling in combination with the reduced-variance dfSDCA method for ERM. This method is a primal variant of SDCA, first analyzed by Shalev-Shwartz (2015), and extended by Csiba and Richtárik (2015) to the arbitrary sampling setting. However, our sampling can be combined with any stochastic method for ERM, such as SGD or S2GD, and extends beyond the realm of ERM, to convex optimization problems in general. However, for simplicity, we do not discuss these extensions in this work.

We analyze the performance of the new sampling theoretically, and by inspecting the results we are able to comment on when can one expect to be able to benefit from it. We illustrate on synthetic data sets with varying distributions of example sizes that our approach can lead to *dramatic speedups* when compared against standard (uniform) minibatching, of *one or more degrees of magnitude.* We then test our method on real data sets and confirm that the use of importance minibatching leads to up to an order of magnitude speedup. Based on our experiments and theory, we predict that for real data with particular shapes and distributions of example sizes, importance sampling for minibatches will operate in a favourable regime, and can lead to speedup higher than one order of magnitude.

2.1 Related work

The idea of using non-uniform sampling in the parallel regime is by no means new. In the following we highlight several recent approaches in a chronological order and we describe their main differences to our method.

The first attempt for a potential speed-up using a non-uniform parallel sampling was proposed in Richtárik and Takáč (2016b). However, to compute the optimal probability vector one has to solve a linear programming problem, which can easily be more complex than the original problem. The authors do not propose a practical version, which would overcome this issue.

The approach described in Zhao and Zhang (2014) uses the idea of a stratified sampling, which is a well-known strategy in statistics. The authors use clustering to group the examples into several partitions and sample an example from each of the partitions uniformly. This approach is similar to ours, with two main differences: i) we do not need clustering for our approach (it can be computationally very expensive) ii) we allow non-uniform sampling inside each of the partitions, which leads to the main speed-up in our work.

Instead of directly improving the convergence rate of the methods, the authors in Csiba and Richtárik (2015) propose a strategy to improve the synchronized parallel implementation of a method by a load-balancing scheme. The method divides the examples into groups, which have similar sum of the amount of nonzero entries. When each core processes a single group, it should take the same time to finish as all the other groups, which leads to shorter waiting time in synchronization. Although this is a non-uniform parallel sampling, this approach takes a completely different direction than our method. The only speedup of the method proposed in the above work is achieved due to a shorter waiting time during the synchronization between parallel processing units, while the method proposed in this work directly decreases the iteration complexity.

Lastly, in Harikandeh et al. (2015) the authors actually propose a scheme for importance sampling with minibatches. In the paper they assume, that they can sample a minibatch with a fixed size (without repetition), such that the probabilities of sampling individual examples will be proportional to some given values. However, this is easier said than done—until our work there was no sampling scheme, which would allow for such minibatches. Therefore, the authors theoretically described an idea, which can be used in practice using our scheme.

3. The Problem

Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ be a data matrix in which features are represented in rows and examples in columns, and let $y \in \mathbb{R}^n$ be a vector of labels corresponding to the examples. Our goal is to find a linear predictor $w \in \mathbb{R}^d$ such that $x_i^\top w \sim y_i$, where the pair $x_i, y_i \in \mathbb{R}^d \times \mathbb{R}$ is sampled from the underlying distribution over data-label pairs. In the L2-regularized Empirical Risk Minimization problem, we find w by solving the optimization problem

$$\min_{w\in\mathbb{R}^d} \left[P(w) := \frac{1}{n}\sum_{i=1}^n \phi_i(\mathbf{X}_{:i}^\top w) + \frac{\lambda}{2}\|w\|_2^2 \right], \tag{1}$$

where $\phi_i : \mathbb{R} \to \mathbb{R}$ is a loss function associated with example-label pair $(\mathbf{X}_{:i}, y_i)$, and $\lambda > 0$. For instance, the square loss function is given by $\phi_i(t) = 0.5(t - y_i)^2$. Our results are not limited to L2-regularized problems though: an arbitrary strongly convex regularizer can be used instead Qu et al. (2015). We shall assume throughout that the loss functions are convex and $1/\gamma$-smooth, where $\gamma > 0$. The latter means that for all $x, y \in \mathbb{R}$ and all $i \in [n] := \{1, 2, \dots, n\}$, we have

$$|\phi_i'(x) - \phi_i'(y)| \le \frac{1}{\gamma}|x - y|.$$

This setup includes ridge and logistic regression, smoothed hinge loss, and many other problems as special cases Shalev-Shwartz and Zhang (2013b). Again, our sampling can be adapted to settings with non-smooth losses, such as the hinge loss.

4. The Algorithm

In this paper we illustrate the power of our new sampling in tandem with Algorithm 1 (dfSDCA) for solving (1).

The method has two parameters. A “sampling” $\hat{S}$, which is a random set-valued mapping Richtárik and Takáč (2016) with values being subsets of $[n]$, the set of examples. No

Algorithm 1 dfSDCA Csiba and Richtárik (2015)

```
Parameters: Sampling $\hat{S}$, stepsize $\theta > 0$
Initialization: Choose $\alpha^{(0)} \in \mathbb{R}^n$,
set $w^{(0)} = \frac{1}{\lambda n}\sum_{i=1}^n \mathbf{X}_{:i}\alpha_i^{(0)},\ p_i = \mathbf{Prob}(i \in \hat{S})$
for $t \geq 1$ do
    Sample a fresh random set $S_t$ according to $\hat{S}$
    for $i \in S_t$ do
        $\Delta_i = \phi_i'(\mathbf{X}_{:i}^\top w^{(t-1)}) + \alpha_i^{(t-1)}$
        $\alpha_i^{(t)} = \alpha_i^{(t-1)} - \theta p_i^{-1}\Delta_i$
    end for
    $w^{(t)} = w^{(t-1)} - \sum_{i\in S_t}\theta(n\lambda p_i)^{-1}\Delta_i\mathbf{X}_{:i}$
end for
```

assumptions are made on the distribution of $\hat{S}$ apart from requiring that p_i is positive for each i, which simply means that each example has to have a chance of being picked. The second parameter is a stepsize θ, which should be as large as possible, but not larger than a certain theoretically allowable maximum depending on P and $\hat{S}$, beyond which the method could diverge.

Algorithm 1 maintains n "dual" variables, $\alpha_1^{(t)}, \dots, \alpha_n^{(t)} \in \mathbb{R}$, which act as variance-reduction shifts. This is most easily seen in the case when we assume that $S_t = \{i\}$ (no minibatching). Indeed, in that case we have

$$w^{(t)} = w^{(t-1)} - \frac{\theta}{n\lambda p_i}(g_i^{(t-1)} + \mathbf{X}_{:i}\alpha_i^{(t-1)}),$$

where $g_i^{(t-1)} := \mathbf{X}_{:i}\Delta_i$ is the stochastic gradient. If θ is set to a proper value, as we shall see next, then it turns out that for all $i \in [n]$, α_i is converging $\alpha_i^* := -\phi_i'(\mathbf{X}_{:i}^\top w^*)$, where w^* is the solution to (1), which means that the shifted stochastic gradient converges to zero. This means that its variance is progressively vanishing, and hence no additional strategies, such as decreasing stepsizes or minibatching are necessary to reduce the variance and stabilize the process. In general, dfSDCA in each step picks a random subset of the examples, denoted as S_t, updates variables $\alpha_i^{(t)}$ for $i \in S_t$, and then uses these to update the predictor w.

4.1 Complexity of dfSDCA

In order to state the theoretical properties of the method, we define

$$E^{(t)} := \frac{\lambda}{2}\|w^{(t)} - w^*\|_2^2 + \frac{\gamma}{2n}\|\alpha^{(t)} - \alpha^*\|_2^2.$$

Most crucially to this paper, we assume the knowledge of parameters $v_1, \dots, v_n > 0$ for which the following ESO[2] inequality holds

$$\mathbf{E}\left[\left\|\sum_{i\in S_t} h_i\mathbf{X}_{:i}\right\|^2\right] \leq \sum_{i=1}^n p_i v_i h_i^2 \tag{2}$$

2. ESO = Expected Separable Overapproximation Richtárik and Takáč (2016); Qu and Richtárik (2016).

holds for all $h \in \mathbb{R}^n$. Tight and easily computable formulas for such parameters can be found in Qu and Richtárik (2016). For instance, whenever $\mathbf{Prob}(|S_t| \leq \tau) = 1$, inequality (2) holds with $v_i = \tau \|\mathbf{X}_{:i}\|^2$. However, this is a conservative choice of the parameters. Convergence of dfSDCA is described in the next theorem.

Theorem 1 (Csiba and Richtárik (2015)) *Assume that all loss functions $\{\phi_i\}$ are convex and $1/\gamma$ smooth. If we run Algorithm 1 with parameter θ satisfying the inequality*

$$\theta \leq \min_i \frac{p_i n \lambda \gamma}{v_i + n\lambda\gamma}, \tag{3}$$

where $\{v_i\}$ satisfy (2), then the potential $E^{(t)}$ decays exponentially to zero as

$$\mathbf{E}\left[E^{(t)}\right] \leq e^{-\theta t} E^{(0)}.$$

Moreover, if we set θ equal to the upper bound in (3) so that

$$\frac{1}{\theta} = \max_i \left(\frac{1}{p_i} + \frac{v_i}{p_i n \lambda \gamma} \right) \tag{4}$$

then

$$t \geq \frac{1}{\theta} \log \left(\frac{(1+\lambda\gamma)E^{(0)}}{\lambda\gamma\epsilon} \right) \qquad \Rightarrow \qquad \mathbf{E}[P(w^{(t)}) - P(w^*)] \leq \epsilon.$$

5. Bucket Sampling

We shall first explain the concept of "standard" importance sampling.

5.1 Standard importance sampling

Assume that $\hat{S}$ always picks a single example only. In this case, (2) holds for $v_i = \|\mathbf{X}_{:i}\|^2$, independently of $p := (p_1, \ldots, p_n)$ Qu and Richtárik (2016). This allows us to choose the sampling probabilities as $p_i \sim v_i + n\lambda\gamma$, which ensures that (4) is minimized. This is *importance sampling.* The number of iterations of dfSDCA is in this case proportional to

$$\frac{1}{\theta^{(\text{imp})}} := n + \frac{\sum_{i=1}^n v_i}{n\lambda\gamma}.$$

If uniform probabilities are used, the average in the above formula gets replaced by the maximum:

$$\frac{1}{\theta^{(\text{unif})}} := n + \frac{\max_i v_i}{\lambda\gamma}.$$

Hence, one should expect the following *speedup* when comparing the importance and uniform samplings:

$$\sigma := \frac{\max_i \|\mathbf{X}_{:i}\|^2}{\frac{1}{n}\sum_{i=1}^n \|\mathbf{X}_{:i}\|^2}. \tag{5}$$

If $\sigma = 10$ for instance, then dfSDCA with importance sampling is $10\times$ faster than dfSDCA with uniform sampling.

5.2 Uniform minibatch sampling

In machine learning, the term "minibatch" is virtually synonymous with a special sampling, which we shall here refer to by the name τ-nice sampling Richtárik and Takáč (2016). Sampling $\hat{S}$ is τ-nice if it picks uniformly at random from the collection of all subsets of $[n]$ of cardinality τ. Clearly, $p_i = \tau/n$ and, moreover, it was show by Qu and Richtárik (2016) that (2) holds with $\{v_i\}$ defined by

$$v_i^{(\tau\text{-nice})} = \sum_{j=1}^{d} \left(1 + \frac{(|J_j|-1)(\tau-1)}{n-1}\right) \mathbf{X}_{ji}^2, \tag{6}$$

where $J_j := \{i \in [n] : \mathbf{X}_{ji} \neq 0\}$. In the case of τ-nice sampling we have the stepsize and complexity given by

$$\theta^{(\tau\text{-nice})} = \min_i \frac{\tau\lambda\gamma}{v_i^{(\tau\text{-nice})} + n\lambda\gamma}, \tag{7}$$

$$\frac{1}{\theta^{(\tau\text{-nice})}} = \frac{n}{\tau} + \frac{\max_i v_i^{(\tau\text{-nice})}}{\tau\lambda\gamma}. \tag{8}$$

Learning from the difference between the uniform and importance sampling of single example (Section 5.1), one would ideally wish the importance minibatch sampling, which we are yet to define, to lead to complexity of the type (8), where the maximum is replaced by an average.

5.3 Bucket sampling: definition

We now propose a family of samplings, which we call *bucket samplings*. Let $B_1, \dots, B_\tau$ be a partition of $[n] = \{1, 2, \dots, n\}$ into τ nonempty sets ("buckets").

Definition 2 (Bucket sampling) *We say that $\hat{S}$ is a bucket sampling if for all $i \in [\tau]$, $|\hat{S} \cap B_i| = 1$ with probability 1.*

Informally, a bucket sampling picks one example from each of the τ buckets, forming a minibatch. Hence, $|\hat{S}| = \tau$ and $\sum_{i \in B_l} p_i = 1$ for each $l = 1, 2 \dots, \tau$, where, as before, $p_i := \mathbf{Prob}(i \in \hat{S})$. Notice that given the partition, the vector $p = (p_1, \dots, p_n)$ *uniquely determines* a bucket sampling. Hence, we have a family of samplings indexed by a single n-dimensional vector. Let $\mathcal{P}_B$ be the set of all vectors $p \in \mathbb{R}^n$ describing bucket samplings associated with partition $B = \{B_1, \dots, B_\tau\}$. Clearly,

$$\mathcal{P}_B = \left\{ p \in \mathbb{R}^n : \sum_{i \in B_l} p_i = 1 \text{ for all } l \;\&\; p_i \geq 0 \text{ for all } i \right\}.$$

Note, that the sampling inside each bucket B_i can be performed in $\mathcal{O}(\log |B_i|)$ time using a binary tree, with an initial overhead and memory of $\mathcal{O}(|B_i| \log |B_i|)$, as explained in Nesterov (2012).

5.4 Optimal bucket sampling

The optimal bucket sampling is that for which (4) is minimized, which leads to a complicated optimization problem:

$$\min_{p \in \mathcal{P}_B} \max_i \frac{1}{p_i} + \frac{v_i}{p_i n \lambda \gamma} \qquad \text{subject to } \{v_i\} \text{ satisfy (2).}$$

A particular difficulty here is the fact that the parameters $\{v_i\}$ depend on the vector p in a complicated way. In order to resolve this issue, we prove the following result.

Theorem 3 *Let $\hat{S}$ be a bucket sampling described by partition $B = \{B_1, \ldots, B_\tau\}$ and vector p. Then the ESO inequality (2) holds for parameters $\{v_i\}$ set to*

$$v_i = \sum_{j=1}^{d} \left(1 + \left(1 - \tfrac{1}{\omega_j'}\right) \delta_j\right) \mathbf{X}_{ji}^2, \tag{9}$$

where $J_j := \{i \in [n] \ : \ \mathbf{X}_{ji} \neq 0\}$, $\delta_j := \sum_{i \in J_j} p_i$ and $\omega_j' := |\{l \ : \ J_j \cap B_l \neq \emptyset\}|$.

Observe that J_j is the set of examples which express feature j, and ω_j' is the number of buckets intersecting with J_j. Clearly, that $1 \leq \omega_j' \leq \tau$ (if $\omega_j' = 0$, we simply discard this feature from our data as it is not needed). Note that the effect of the quantities $\{\omega_j'\}$ on the value of v_i is small. Indeed, unless we are in the extreme situation when $\omega_j' = 1$, which has the effect of neutralizing δ_j, the quantity $1 - 1/\omega_j'$ is between $1 - 1/2$ and $1 - 1/\tau$. Hence, for simplicity, we could instead use the slightly more conservative parameters:

$$v_i = \sum_{j=1}^{d} \left(1 + \left(1 - \frac{1}{\tau}\right) \delta_j\right) \mathbf{X}_{ji}^2.$$

5.5 Uniform bucket sampling

Assume all buckets are of the same size: $|B_l| = n/\tau$ for all l. Further, assume that $p_i = 1/|B_l| = \tau/n$ for all i. Then $\delta_j = \tau|J_j|/n$, and hence Theorem 3 says that

$$v_i^{(\text{unif})} = \sum_{j=1}^{d} \left(1 + \left(1 - \frac{1}{\omega_j'}\right) \frac{\tau|J_j|}{n}\right) \mathbf{X}_{ji}^2, \tag{10}$$

and in view of (4), the complexity of dfSDCA with this sampling becomes

$$\frac{1}{\theta^{(\text{unif})}} = \frac{n}{\tau} + \frac{\max_i v_i^{(\text{unif})}}{\tau \lambda \gamma}. \tag{11}$$

Formula (6) is very similar to the one for τ-nice sampling (10), despite the fact that the sets/minibatches generated by the uniform bucket sampling have a special structure with respect to the buckets. Indeed, it is easily seen that the difference between between $1 + \frac{\tau|J_j|}{n}$ and $1 + \frac{(\tau-1)(|J_j|-1)}{(n-1)}$ is negligible. Moreover, if either $\tau = 1$ or $|J_j| = 1$ for all j, then $\omega_j' = 1$ for all j and hence $v_i = \|\mathbf{X}_{:i}\|^2$. This is also what we get for the τ-nice sampling.

quantity $\backslash$ iteration	1	2	3	4	5	6
$\max_i(\lvert p_i^{\text{new}} - p_i^{\text{old}}\rvert)$	$7 \cdot 10^{-5}$	$7 \cdot 10^{-6}$	$7 \cdot 10^{-7}$	$8 \cdot 10^{-8}$	$8 \cdot 10^{-9}$	$9 \cdot 10^{-10}$
$\lVert p^{\text{new}} - p^{\text{old}}\rVert_2$	$1 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-7}$	$2 \cdot 10^{-8}$

Table 1: Example of the convergence speed of the alternating optimization scheme for $w8a$ data set (see Table 5) with $\tau = 8$. The table demonstrates the difference in probabilities for two successive iterations (p^{old} and p^{new}). We observed a similar behaviour for all data sets and all choices of τ.

6. Importance Minibatch Sampling

In the light of Theorem 3, we can formulate the problem of searching for the optimal bucket sampling as

$$\min_{p \in \mathcal{P}_B} \max_i \frac{1}{p_i} + \frac{v_i}{p_i n \lambda \gamma} \quad \text{subject to } \{v_i\} \text{ satisfy (9).} \tag{12}$$

Still, this is not an easy problem. *Importance minibatch sampling* arises as an approximate solution of (12). Note that the uniform minibatch sampling is a feasible solution of the above problem, and hence we should be able to improve upon its performance.

6.1 Approach 1: alternating optimization

Given a probability distribution $p \in \mathcal{P}_B$, we can easily find v using Theorem 3. On the other hand, for any fixed v, we can minimize (12) over $p \in \mathcal{P}_B$ by choosing the probabilities in each group B_l and for each $i \in B_l$ via

$$p_i = \frac{n\lambda\gamma + v_i}{\sum_{j \in B_l} n\lambda\gamma + v_j}. \tag{13}$$

This leads to a natural alternating optimization strategy. An example of the standard convergence behaviour of this scheme is showed in Table 6.1. Empirically, this strategy converges to a pair (p^*, v^*) for which (13) holds. Therefore, the resulting complexity will be

$$\frac{1}{\theta^{(\tau\text{-imp})}} = \frac{n}{\tau} + \max_{l \in [\tau]} \frac{\frac{\tau}{n} \sum_{i \in B_l} v_i^*}{\tau \lambda \gamma}. \tag{14}$$

We can compare this result against the complexity of τ-nice in (8). We can observe that the terms are very similar, up to two differences. First, the importance minibatch sampling has a maximum over group averages instead of a maximum over everything, which leads to speedup, other things equal. On the other hand, $v^{(\tau\text{-nice})}$ and v^* are different quantities. The alternating optimization procedure for computation of (v^*, p^*) is costly, as one iteration takes a pass over all data. Therefore, in the next subsection we propose a closed form formula which, as we found empirically, offers nearly optimal convergence rate.

6.2 Approach 2: practical formula

For each group B_l, let us choose for all $i \in B_l$ the probabilities as follows:

$$\boxed{p_i^* = \frac{n\lambda\gamma + v_i^{(\text{unif})}}{\sum_{k \in B_l} n\lambda\gamma + v_k^{(\text{unif})}}} \tag{15}$$

where $v_i^{(\text{unif})}$ is given by (10). Note that computing all $v_i^{(\text{unif})}$ can be done by visiting every non-zero entry of $\mathbf{X}$ once and and computing all p_i^* is a simple re-weighting. This is the same computational cost as for standard serial importance sampling. Also, this process can be straightforwardly parallelized, fully utilizing all the cores, which leads to τ times faster computations. The overhead of using this sampling approach is therefore at most one pass over the data, which is negligible in most scenarios considered.

After doing some simplifications, the associated complexity result is

$$\frac{1}{\theta^{(\tau\text{-imp})}} = \max_l \left\{ \left(\frac{n}{\tau} + \frac{\frac{\tau}{n}\sum_{i \in B_l} v_i^{(\text{unif})}}{\tau\lambda\gamma} \right) \beta_l \right\}, \tag{16}$$

where

$$\beta_l := \max_{i \in B_l} \frac{n\lambda\gamma + s_i}{n\lambda\gamma + v_i^{(\text{unif})}}, \qquad s_i := \sum_{j=1}^{d} \left(1 + \left(1 - \frac{1}{\omega_j'}\right) \sum_{k \in J_j} p_k^* \right) \mathbf{X}_{ji}^2.$$

We would ideally want to have $\beta_l = 1$ for all l (this is what we get for importance sampling without minibatches). If $\beta_l \approx 1$ for all l, then the complexity $1/\theta^{(\tau\text{-imp})}$ is an improvement on the complexity of the uniform minibatch sampling since the maximum of group averages is always better than the maximum of all elements $v_i^{(\text{uni})}$:

$$\frac{n}{\tau} + \frac{\max_l \left(\frac{\tau}{n} \sum_{i \in B_l} v_i^{(\text{unif})} \right)}{\tau\lambda\gamma} \leq \frac{n}{\tau} + \frac{\max_i v_i^{(\text{unif})}}{\tau\lambda\gamma}.$$

Indeed, the difference can be very large.

Finally, we would like to comment on the choice of the partitions $B_1, \ldots, B_\tau$, as they clearly affect the convergence rate. The optimal choice of the partitions is given by minimizing in $B_1, \ldots, B_\tau$ the maximum over group sums in (16), which is a complicated optimization problem. Instead, we used random partitions of the same size in our experiments, which we believe is a good solution for the partitioning problem. The logic is simple: the minimum of the maximum over the group sums will be achieved, when all the group sums have similar values. If we set the partitions to the same size and we distribute the examples randomly, there is a good chance that the group sums will have similar values (especially for large amounts of data).

7. Experiments

We now comment on the results of our numerical experiments, with both synthetic and real data sets. We plot the optimality gap $P(w^{(t)}) - P(w^*)$ and in the case of real data also the test error (vertical axis) against the computational effort (horizontal axis). We measure computational effort by the number of effective passes through the data divided by τ. We divide by τ as a normalization factor; since we shall compare methods with a range of values of τ. This is reasonable as it simply indicates that the τ updates are performed in parallel. Hence, what we plot is an implementation-independent model for time.

We compared two algorithms:

1) **τ-nice**: dfSDCA using the τ-nice sampling with stepsizes given by (7) and (6),

2) **τ-imp**: dfSDCA using τ-importance sampling (i.e., importance minibatch sampling) defined in Subsection 6.2.

As the methods are randomized, we always plot the average over 5 runs. For each data set we provide two plots. In the left figure we plot the convergence of τ-nice for different values of τ, and in the right figure we do the same for τ-importance. The horizontal axis has the same range in both plots, so they are easily comparable. The values of τ we used to plot are $\tau \in \{1, 2, 4, 8, 16, 32\}$. In all experiments we used the logistic loss: $\phi_i(z) = \log(1+e^{-y_i z})$ and set the regularizer to $\lambda = \max_i \|\mathbf{X}_{:i}\|/n$. We will observe the theoretical and empirical ratio $\theta^{(\tau\text{-imp})}/\theta^{(\tau\text{-nice})}$. The theoretical ratio is computed from the corresponding theory. The empirical ratio is the ratio between the horizontal axis values at the moments when the algorithms reached the precision 10^{-10}.

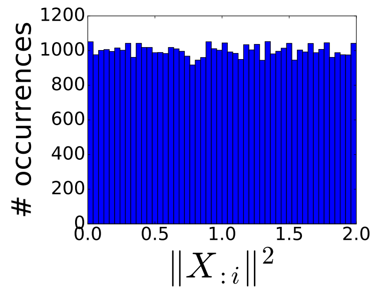
7.1 Artificial data

We start with experiments using artificial data, where we can control the sparsity pattern of $\mathbf{X}$ and the distribution of $\{\|\mathbf{X}_{:i}\|^2\}$. We fix $n = 50,000$ and choose $d = 10,000$ and $d = 1,000$. For each feature we sampled a random sparsity coefficient $\omega'_i \in [0, 1]$ to have the average sparsity $\omega' := \frac{1}{d}\sum_i^d \omega'_i$ under control. We used two different regimes of sparsity: $\omega' = 0.1$ (10% nonzeros) and $\omega' = 0.8$ (80% nonzeros). After deciding on the sparsity pattern, we rescaled the examples to match a specific distribution of norms $L_i = \|\mathbf{X}_{:i}\|^2$; see Table 2. The code column shows the corresponding code in Julia to create the vector of norms L. The distributions can be also observed as histograms in Figure 1.

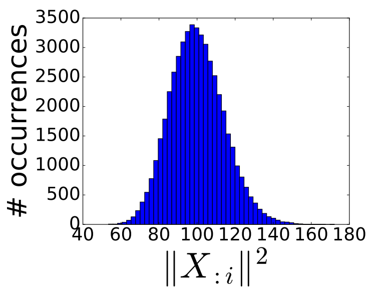
label	code	σ
extreme	L = ones(n);L[1] = 1000	980.4
chisq1	L = rand(chisq(1),n)	17.1
chisq10	L = rand(chisq(10),n)	3.9
chisq100	L = rand(chisq(100),n)	1.7
uniform	L = 2*rand(n)	2.0

Table 2: Distributions of $\|\mathbf{X}_{:i}\|^2$ used in artificial experiments.

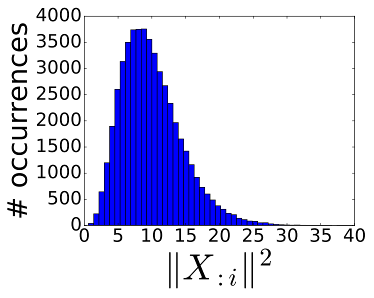
The corresponding experiments can be found in Figure 4 and Figure 5. The theoretical and empirical speedup are also summarized in Tables 3 and 4.



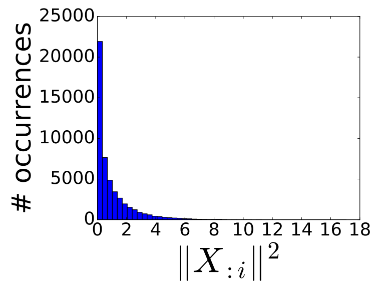
(a) uniform



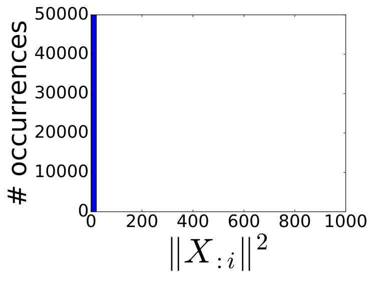
(b) chisq100



(c) chisq10



(d) chisq1



(e) extreme

Figure 1: The distribution of $\|\mathbf{X}_{:i}\|^2$ for synthetic data

Data	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 32$
uniform	1.2 : 1.0	1.2 : 1.1	1.2 : 1.1	1.2 : 1.1	1.3 : 1.1	1.4 : 1.1
chisq100	1.5 : 1.3	1.5 : 1.3	1.5 : 1.4	1.6 : 1.4	1.6 : 1.4	1.6 : 1.4
chisq10	1.9 : 1.4	1.9 : 1.5	2.0 : 1.4	2.2 : 1.5	2.5 : 1.6	2.8 : 1.7
chisq1	1.9 : 1.4	2.0 : 1.4	2.2 : 1.5	2.5 : 1.6	3.1 : 1.6	4.2 : 1.7
extreme	8.8 : 4.8	9.6 : 6.6	11 : 6.4	14 : 6.4	20 : 6.9	32 : 6.1

Table 3: The **theoretical** : **empirical** ratios $\theta^{(\tau\text{-imp})}/\theta^{(\tau\text{-nice})}$ for sparse artificial data ($\omega' = 0.1$)

Data	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 32$
uniform	1.2 : 1.1	1.2 : 1.1	1.4 : 1.2	1.5 : 1.2	1.7 : 1.3	1.8 : 1.3
chisq100	1.5 : 1.3	1.6 : 1.4	1.6 : 1.5	1.7 : 1.5	1.7 : 1.6	1.7 : 1.6
chisq10	1.9 : 1.3	2.2 : 1.6	2.7 : 2.1	3.1 : 2.3	3.5 : 2.5	3.6 : 2.7
chisq1	1.9 : 1.3	2.6 : 1.8	3.7 : 2.3	5.6 : 2.9	7.9 : 3.2	10 : 3.9
extreme	8.8 : 5.0	15 : 7.8	27 : 12	50 : 16	91 : 21	154 : 28

Table 4: The **theoretical** : **empirical** ratios $\theta^{(\tau\text{-imp})}/\theta^{(\tau\text{-nice})}$. Artificial data with $\omega' = 0.8$ (dense)

7.2 Real data

We used several publicly available data sets[3], summarized in Table 5, which we randomly split into a train (80%) and a test (20%) part. The test error is measured by the empirical risk (1) on the test data without a regularizer. The resulting test error was compared against the best achievable test error, which we computed by minimizing the corresponding risk. Experimental results are in Figure 7.3 and Figure 7.3. The theoretical and empirical speedup table for these data sets can be found in Table 6.

Data set	#samples	#features	sparsity	σ
ijcnn1	35,000	23	60.1%	2.04
protein	17,766	358	29.1%	1.82
w8a	49,749	301	4.2%	9.09
url	2,396,130	3,231,962	0.04 %	4.83
aloi	108,000	129	24.6%	26.01

Table 5: Summary of real data sets (σ = predicted speedup).

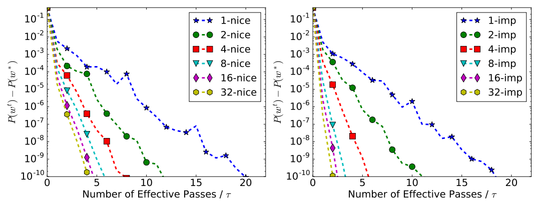
Data	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 32$
ijcnn1	1.2 : 1.1	1.4 : 1.1	1.6 : 1.3	1.9 : 1.6	2.2 : 1.6	2.3 : 1.8
protein	1.3 : 1.2	1.4 : 1.2	1.5 : 1.4	1.7 : 1.4	1.8 : 1.5	1.9 : 1.5
w8a	2.8 : 2.0	2.9 : 1.9	2.9 : 1.9	3.0 : 1.9	3.0 : 1.8	3.0 : 1.8
url	3.0 : 2.3	2.6 : 2.1	2.0 : 1.8	1.7 : 1.6	1.8 : 1.6	1.8 : 1.7
aloi	13 : 7.8	12 : 8.0	11 : 7.7	9.9 : 7.4	9.3 : 7.0	8.8 : 6.7

Table 6: The **theoretical** : **empirical** ratios $\theta^{(\tau\text{-imp})}/\theta^{(\tau\text{-nice})}$.

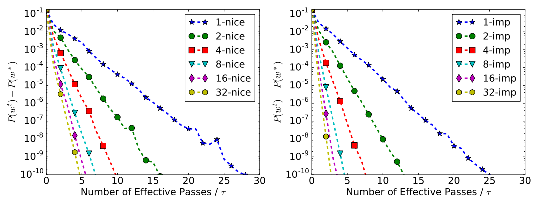
7.3 Conclusion

In all experiments, τ-importance sampling performs significantly better than τ-nice sampling. The theoretical speedup factor computed by $\theta^{(\tau\text{-imp})}/\theta^{(\tau\text{-nice})}$ provides an excellent estimate of the actual speedup. We can observe that on denser data the speedup is higher than on sparse data. This matches the theoretical intuition for v_i for both samplings. Similar behaviour can be also observed for the test error, which is pleasing. As we observed for artificial data, for extreme data sets the speedup can be arbitrary large, even several orders of magnitude. *A rule of thumb: if one has data with large σ, practical speedup from using importance minibatch sampling will likely be dramatic.*

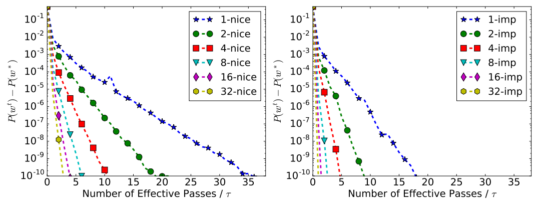
3. https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/



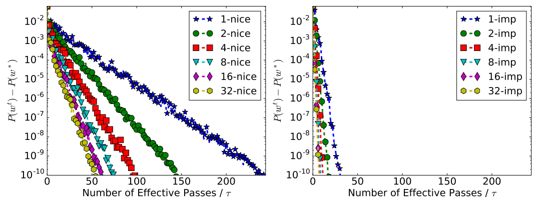
(a) ijcnn1, τ-nice (left), τ-importance (right)



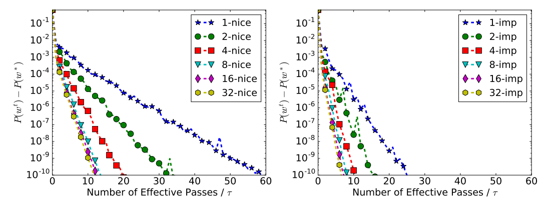
(b) protein, τ-nice (left), τ-importance (right)



(c) w8a, τ-nice (left), τ-importance (right)

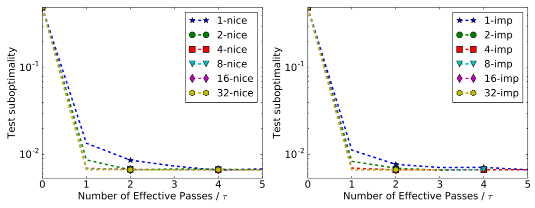


(d) aloi, τ-nice (left), τ-importance (right)

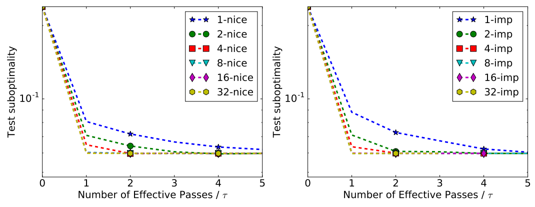


(e) url, τ-nice (left), τ-importance (right)

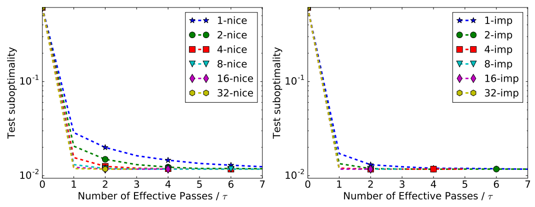
Figure 2: Train error over iterations for data sets from Table 5



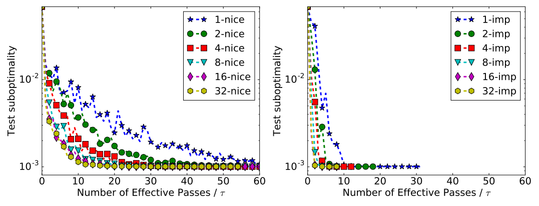
(a) ijcnn1, τ-nice (left), τ-importance (right)



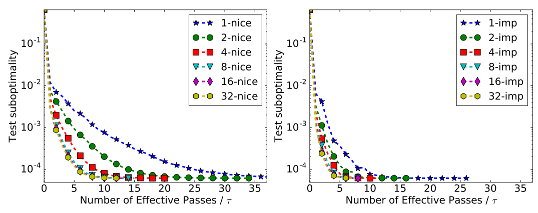
(b) protein, τ-nice (left), τ-importance (right)



(c) w8a, τ-nice (left), τ-importance (right)

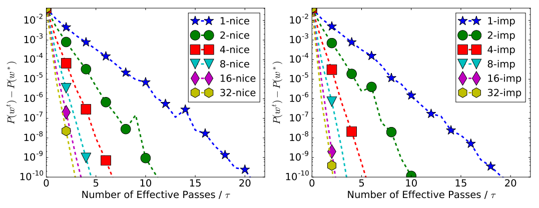


(d) aloi, τ-nice (left), τ-importance (right)

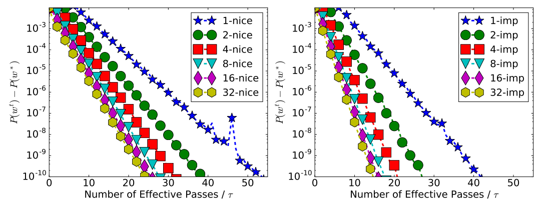


(e) url, τ-nice (left), τ-importance (right)

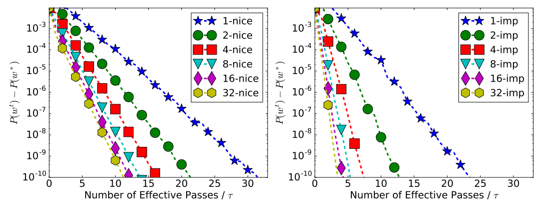
Figure 3: Test error over iterations for data sets from Table 5



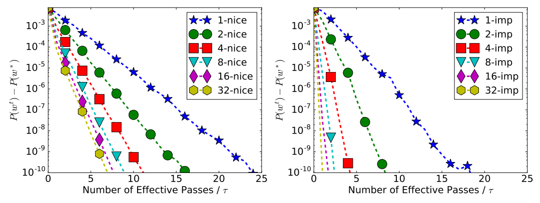
(a) uniform, τ-nice (left), τ-importance (right)



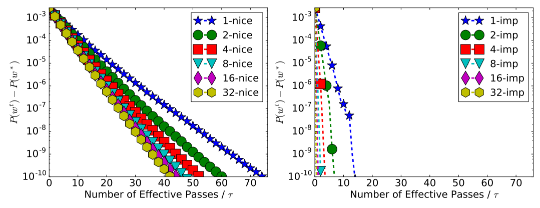
(b) chisq100, τ-nice (left), τ-importance (right)



(c) chisq10, τ-nice (left), τ-importance (right)

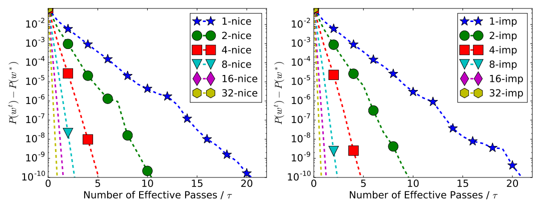


(d) chisq1, τ-nice (left), τ-importance (right)

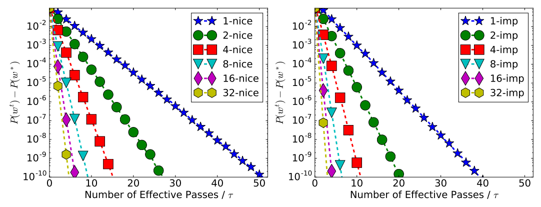


(e) extreme, τ-nice (left), τ-importance (right)

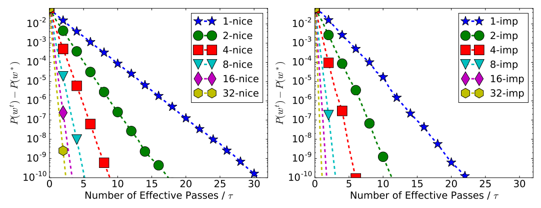
Figure 4: Artificial data sets from Table 2 with $\omega = 0.8$



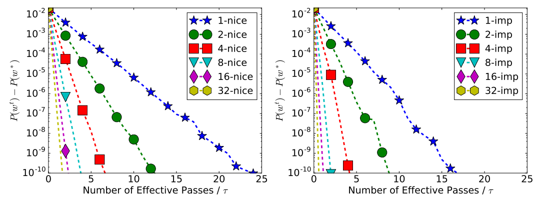
(a) uniform, τ-nice (left), τ-importance (right)



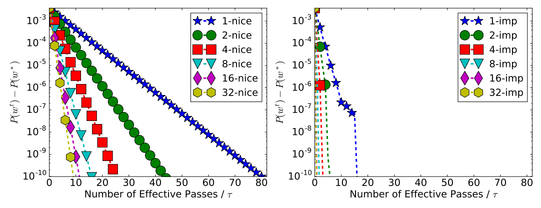
(b) chisq100, τ-nice (left), τ-importance (right)



(c) chisq10, τ-nice (left), τ-importance (right)



(d) chisq1, τ-nice (left), τ-importance (right)



(e) extreme, τ-nice (left), τ-importance (right)

Figure 5: Artificial data sets from Table 2 with $\omega = 0.1$

8. Proof of Theorem 3

8.1 Three lemmas

We first establish three lemmas, and then proceed with the proof of the main theorem. With each sampling $\hat{S}$ we associate an $n \times n$ "probability matrix" defined as follows: $\mathbf{P}_{ij}(\hat{S}) = \mathbf{Prob}(i \in \hat{S}, j \in \hat{S})$. Our first lemma characterizes the probability matrix of the bucket sampling.

Lemma 4 *If $\hat{S}$ is a bucket sampling, then*

$$\mathbf{P}(\hat{S}) = pp^\top \circ (\mathbf{E} - \mathbf{B}) + \mathrm{Diag}(p), \tag{17}$$

where $\mathbf{E} \in \mathbb{R}^{n\times n}$ is the matrix of all ones,

$$\mathbf{B} := \sum_{l=1}^{\tau} \mathbf{P}(B_l), \tag{18}$$

and $\circ$ denotes the Hadamard (elementwise) product of matrices. Note that $\mathbf{B}$ is the 0-1 matrix given by $\mathbf{B}_{ij} = 1$ if and only if i, j belong to the same bucket B_l for some l.

Proof Let $\mathbf{P} = \mathbf{P}(\hat{S})$. By definition

$$\mathbf{P}_{ij} = \begin{cases} p_i & i = j \\ p_i p_j & i \in B_l,\ j \in B_k,\ l \neq k \\ 0 & \text{otherwise.} \end{cases}$$

It only remains to compare this to (17). ■

Lemma 5 *Let J be a nonempty subset of $[n]$, let $\mathbf{B}$ be as in Lemma 4 and put $\omega'_J := |\{l : J \cap B_l \neq \emptyset\}|$. Then*

$$\mathbf{P}(J) \circ \mathbf{B} \succeq \frac{1}{\omega'_J}\mathbf{P}(J). \tag{19}$$

Proof For any $h \in \mathbb{R}^n$, we have

$$h^\top \mathbf{P}(J) h = \left(\sum_{i\in J} h_i\right)^2 = \left(\sum_{l=1}^{\tau} \sum_{i \in J\cap B_l} h_i\right)^2 \leq \omega'_J \sum_{l=1}^{\tau} \left(\sum_{i\in J\cap B_l} h_i\right)^2 = \omega'_J \sum_{l=1}^{\tau} h^\top \mathbf{P}(J \cap B_l) h,$$

where we used the Cauchy-Schwarz inequality. Using this, we obtain

$$\mathbf{P}(J) \circ \mathbf{B} \overset{(18)}{=} \mathbf{P}(J) \circ \sum_{l=1}^{\tau} \mathbf{P}(B_l) = \sum_{l=1}^{\tau} \mathbf{P}(J) \circ \mathbf{P}(B_l) = \sum_{l=1}^{\tau} \mathbf{P}(J \cap B_l) \overset{(8.1)}{\succeq} \frac{1}{\omega'}\mathbf{P}(J).$$

■

Lemma 6 *Let J be any nonempty subset of $[n]$ and $\hat{S}$ be a bucket sampling. Then*

$$\mathbf{P}(J) \circ pp^\top \preceq \left(\sum_{i \in J} p_i \right) \mathrm{Diag}(\mathbf{P}(J \cap \hat{S})). \tag{20}$$

Proof Choose any $h \in \mathbb{R}^n$ and note that

$$h^\top (\mathbf{P}(J) \circ pp^\top) h = \left(\sum_{i \in J} p_i h_i \right)^2 = \left(\sum_{i \in J} x_i y_i \right)^2,$$

where $x_i = \sqrt{p_i} h_i$ and $y_i = \sqrt{p_i}$. It remains to apply the Cauchy-Schwarz inequality:

$$\sum_{i \in J} x_i y_i \leq \sum_{i \in J} x_i^2 \sum_{i \in J} y_i^2$$

and notice that the i-th element on the diagonal of $\mathbf{P}(J \cap \hat{S})$ is p_i for $i \in J$ and 0 for $i \notin J$ ■

8.2 Proof of Theorem 3

By Theorem 5.2 in Qu and Richtárik (2016), we know that inequality (2) holds for parameters $\{v_i\}$ set to

$$v_i = \sum_{j=1}^{d} \lambda'(\mathbf{P}(J_j \cap \hat{S})) \mathbf{X}_{ji}^2,$$

where $\lambda'(\mathbf{M})$ is the largest normalized eigenvalue of symmetric matrix $\mathbf{M}$ defined as

$$\lambda'(\mathbf{M}) := \max_h \left\{ h^\top \mathbf{M} h \ : \ h^\top \mathrm{Diag}(\mathbf{M}) h \leq 1 \right\}.$$

Furthermore,

$$\begin{aligned}
\mathbf{P}(J_j \cap \hat{S}) &= \mathbf{P}(J_j) \circ \mathbf{P}(\hat{S}) \\
&\overset{(17)}{=} \mathbf{P}(J_j) \circ pp^\top - \mathbf{P}(J_j) \circ pp^\top \circ \mathbf{B} + \mathbf{P}(J_j) \circ \mathrm{Diag}(p) \\
&\overset{(19)}{\preceq} \left(1 - \frac{1}{\omega'_J} \right) \mathbf{P}(J_j) \circ pp^\top + \mathbf{P}(J_j) \circ \mathrm{Diag}(p) \\
&\overset{(20)}{\preceq} \left(1 - \frac{1}{\omega'_J} \right) \delta_j \, \mathrm{Diag}(\mathbf{P}(J_j \cap \hat{S})) + \mathrm{Diag}(\mathbf{P}(J_j \cap \hat{S})),
\end{aligned}$$

whence $\lambda'(\mathbf{P}(J_j \cap \hat{S})) \leq 1 + (1 - 1/\omega'_J)\, \delta_j$, which concludes the proof.

References

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pages 177–186. Springer, 2010.

Dominik Csiba and Peter Richtárik. Primal method for ERM with flexible mini-batching schemes and non-convex losses. *arXiv:1506.02227*, 2015.

Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. ICML, 2015.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS 27*, pages 1646–1654, 2014.

Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for minimizing non-strongly convex losses. *IEEE International Workshop on Machine Learning for Signal Processing*, 2014.

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

Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stop wasting my gradients: Practical SVRG. In *NIPS 28*, pages 2251–2259, 2015.

Geoffrey E Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(2):513–529, 2012.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS 26*, 2013.

Jakub Konečný and Peter Richtárik. S2GD: Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3(9):1–14, 2017.

Jakub Konečný, Jie Lu, Peter Richtárik, and Martin Takáč. mS2GD: Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

Jakub Konečný, Zheng Qu, and Peter Richtárik. Semi-stochastic coordinate descent. *Optimization Methods and Software*, 32(5):993–1005, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS 25*, pages 1097–1105, 2012.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.

Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Ion Necoara and Andrei Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *NIPS 27*, pages 1017–1025, 2014.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Zheng Qu and Peter Richtárik. Coordinate descent methods with arbitrary sampling I: algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.

Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling II: expected separable overapproximation. *Optimization Methods and Software*, 31(5):858–884, 2016.

Zheng Qu, Peter Richtárik, and Tong Zhang. Quartz: Randomized dual coordinate ascent with arbitrary sampling. In *NIPS 28*, pages 865–873, 2015.

Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *JMLR*, 17(75):1–25, 2016a.

Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016b.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144 (2):1–38, 2014.

Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1):433–484, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 1951.

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

Shai Shalev-Shwartz. SDCA without duality. *arXiv:1502.06177*, 2015.

Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *JMLR*, 12:1865–1892, 2011.

Shai Shalev-Shwartz and Tong Zhang. Proximal stochastic dual coordinate ascent. *arXiv:1211.2717*, 2012.

Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *NIPS 26*, pages 378–385. 2013a.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *JMLR*, 14(1):567–599, 2013b.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

Martin Takáč, Avleen Singh Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. In *ICML*, 2013.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, 2004.

Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, 2015.

Peilin Zhao and Tong Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv:1405.3080*, 2014.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling. In *ICML*, 2015.