Procedia Computer Science
Volume 53, 2015, Pages 308–315
2015 INNS Conference on Big Data

Scalability of Stochastic Gradient Descent based on “Smart” Sampling Techniques

Stephan Clémençon[1], Aurélien Bellet[1], Ons Jelassi[1], and Guillaume Papa[1]*

Institut Mines-Telecom LTCI UMR Telecom ParisTech & CNRS No. 5141, 75013 Paris, France
{stephan.clemencon,aurelien.bellet,ons.jelassi,guillaume.papa}@telecom-paristech.fr

Abstract
Various appealing ideas have been recently proposed in the statistical literature to scale-up machine learning techniques and solve predictive/inferential problems from “Big Datasets”. Beyond the massively parallelized and distributed approaches exploiting hardware architectures and programming frameworks that have received increasing interest these last few years, several variants of the Stochastic Gradient Descent (SGD) method based on “smart” sampling procedures have been designed for accelerating the model fitting stage. Such techniques exploit either the form of the objective functional or some supposedly available auxiliary information, and have been thoroughly investigated from a theoretical viewpoint. Though attractive, such statistical methods must be also analyzed from a computational perspective, bearing the possible options offered by recent technological advances in mind. It is thus of vital importance to investigate how to implement efficiently these inferential principles in order to achieve the best trade-off between computational time and accuracy. In this paper, we explore the scalability of the SGD techniques introduced in [11, 9] from an experimental perspective. Issues related to their implementation on distributed computing platforms such as Apache Spark are also discussed and experimental results based on large-scale real datasets are displayed in order to illustrate the relevance of the promoted approaches.

Keywords: Large-scale machine-learning, sampling schemes, stochastic gradient descent

1 Introduction

Motivated by a wide variety of predictive problems (recommender systems, fraud detection,maintenance of large energy/transportation networks, customer relationship management...), scaling up machine learning methods to the massive datasets collected by the large number of sensors around us is currently the subject of a good deal of attention. Taking up this challenge is critical so that the ”Big Data” phenomenon can keep its promises. This requires developing novel lines of research which involve both statistics and computer science,

*This work is supported by the Chair Machine-Learning for Big Data of Telecom ParisTech

doi:10.1016/j.procs.2015.07.308

as pointed out in [20] for instance. More specifically, the effectiveness of novel ideas proposed in the field of statistical science should also be evaluated from a computational perspective, taking into account the recent evolution of distributed and parallelized computing platforms and programming frameworks.

The availability of hardware architecture allowing for distributed and parallel processing of very large datasets together with the increasing popularity of machine learning applications in a broad range of fields have recently lead to the design of *parallelized/distributed* variants of certain statistical learning algorithms, refer to [1, 24, 13, 23, 17, 4] among others. In parallel to these attempts, the advantages of undersampling and randomized techniques have also attracted a lot of interest, see [16, 18, 27, 26] for instance. For instance, it has been shown in [11] and [9] (see also [10] and [8]) that, in contrast to naive subsampling strategies, specific sampling schemes can be incorporated to iterative statistical learning methods based on Stochastic Gradient Descent (SGD) in order to drastically reduce the number of observations involved in the gradient estimation steps, while preserving theoretical learning rates or rate bounds. More precisely, when the computation of natural estimates of the objective function involves the summation over pairs of observations in a dataset of very large size $n \geq 1$, it has been proved in [8] that a SGD implementation based on $O(n)$ pairs selected by a simple Monte-Carlo procedure (*i.e.* drawing with replacement) at each iteration yields exactly the same performance in terms of rate bounds than a gradient descent based on the $n(n-1)$ pairs available. In the same vein, the impact of subsampling with unequal weights at each iteration of the SGD method has been studied at length in [9], revealing that a significant gain (in terms of asymptotic variance) can be reached when the weights are positively correlated with the modulus of the local gradient estimate based on the whole dataset, surpassing the usual mini-batch SGD implementation from the asymptotic analysis viewpoint.

In this paper, we investigate the effective scalability of the SGD variants mentioned above for some learning tasks serving as running examples throughout the paper (*i.e.* fitting a high-dimensional logistic regression model and metric learning for image recognition). We discuss their efficient implementations, in particular in the context of the Spark cluster computing framework (see [25]) and its distributed memory-based architecture, taking into account the type of parallelization they allow and the latencies they induce. Furthermore, we evaluate these approaches in terms of the trade-off between computational time and accuracy on large real-world datasets and show that they outperform classical sampling strategies.

The article is organized as follows. Section 2 introduces the background material. We briefly recall some basics of the SGD method for Empirical Risk Minimization (ERM) along with two important use-cases, and review the sampling techniques of interest to speed-up SGD. Section 3 discusses their efficient implementation, in particular in the Spark framework. Section 4 presents our experimental results and we conclude in Section 5.

2 Background

We start off with recalling a few notions, which the subsequent experimental study crucially relies on: in particular, certain concepts pertaining to the stochastic approximation theory and involved in the description of the iterative statistical learning methods analyzed in this article.

2.1 Stochastic Gradient Descent for Empirical Risk Minimization

In many situations, the goal pursued in statistics and machine learning is to find the model parameters that minimize a function $L_F(\theta)$ depending on an unknown probability distribution

F on $\mathbb{R}^d$ with $d \geq 1$ say, referred to as the *risk*, over a space $\Theta \subset \mathbb{R}^q$ with $q \geq 1$. In the simplest setting, the risk takes the form of the expectation of a loss function $\psi(Z,\ \theta)$, where Z is a r.v. with distribution $F(dz)$, which measures the performance of the learning system with parameter θ when observing Z: $L_F(\theta) = \mathbb{E}[\psi(Z,\ \theta)] = \int \psi(z,\ \theta)F(dz)$. In practice, the risk is unknown, just like the probability distribution F, and must be replaced by an estimate $\hat{L}_N(\theta)$ based on a training dataset $\mathcal{D}_N$ supposedly available (a sample $Z_1,\ \ldots,\ Z_N$ of independent copies of the generic r.v. Z in the previous example) and the Empirical Risk Minimization (ERM) principle selects the model with parameters:

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \hat{L}_N(\theta).$$

For smooth objectives, this can be done using gradient descent, following the iterations:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \hat{L}_N(\theta_t),$$

where ∇_θ denotes the gradient w.r.t. θ (see [5]). The initial value $\theta_0 \in \Theta$ is chosen arbitrarily and the learning rate (or step size) $\eta_t \geq 0$ satisfies $\sum_{t=1}^{+\infty} \eta_t = +\infty$ and $\sum_{t=1}^{+\infty} \eta_t^2 < +\infty$. Observe incidentally that, in the previously mentioned example, we have $\nabla_\theta \hat{L}_N(\theta) = (1/N)\sum_{i=1}^N \nabla_\theta \psi(Z_i, \theta)$. Here we place ourselves in the large-scale setting, where the sample size N of the training dataset is so large that computing the gradient of $\hat{L}_N(\theta)$ at each iteration is too costly regarding available computational resources. A natural approach, referred to as Stochastic Gradient Descent (SGD), consists in replacing $\nabla_\theta \hat{L}_N(\theta_t)$ by a version computed from a subset $S \subset \mathcal{D}_N$ of reduced size $m \ll N$ drawn uniformly at random, see [5]. Iterative statistical learning techniques based on SGD include many popular algorithms, such as SVM, Neural Networks or soft K-means. One may refer to [21] for an excellent account of theoretical results related to SGD convergence.

Before describing two statistical methods to improve upon the standard SGD technique in certain specific situations and providing details about their implementation, we describe two ERM problems that serve as running examples in this paper.

Example 2.1. (Linear logistic regression modeling) *We place ourselves in the usual binary classification framework, where Y is a binary random output, taking its values in $\{-1,\ +1\}$ say, and X is an input random vector valued in a high-dimensional space $\mathbb{R}^d$, modeling some (hopefully) useful observation for predicting Y. Let $\langle .,\ . \rangle$ denote the usual scalar product in $\mathbb{R}^d$. Based on training data $\{(X_1, Y_1),\ \ldots,\ (X_N, Y_N)\}$, the goal pursued is to find $\theta = (\alpha,\ \beta) \in \Theta \subset \mathbb{R} \times \mathbb{R}^d$, so as to minimize the smooth functional*

$$\Lambda_N(\theta) = -\sum_{i=1}^N \frac{Y_i + 1}{2} \log\left(\frac{\exp(\alpha + \langle X_i,\ \beta\rangle)}{1 + \exp(\alpha + \langle X_i,\ \beta\rangle)}\right) - \sum_{i=1}^N \frac{1 - Y_i}{2} \log\left(\frac{1}{1 + \exp(\alpha + \langle X_i,\ \beta\rangle)}\right), \tag{1}$$

which the opposite of the conditional log-likelihood given the X_i's related to the parametric (linear) logistic regression model: $\mathbb{P}_\theta\{Y = +1 \mid X\} = exp(\alpha + \langle X_i,\ \beta\rangle)/(1 + exp(\alpha + \langle X_i,\ \beta\rangle))$, $(\alpha,\ \beta) \in \Theta$. This model has been widely used in various applications such as credit-risk screening, medical diagnosis support, churn analysis or targeted advertising. The increasing input dimensionality d of the model combined with a possibly very large number N of data instances involved in (1) *make it computationally challenging to fit such a model.*

Example 2.2. (Metric learning for image recognition) *Like in many other problems in machine learning and data mining (e.g. recommender systems, clustering, ranking), choosing* *an appropriate distance measure is crucial to the performance of automatic image recognition methods [3]. Suppose that images are stored in a database in the form of a feature vector* X *taking its values in* $\mathcal{X} \subset \mathbb{R}^d$ *and are assigned to a label/class* Y*, in* $\mathcal{Y} = \{1, \ldots, C\}$ *with* $C \geq 2$ *say, describing the types of object they represent [14]. Based on a collection of manually annotated images* $(X_1, Y_1), \ldots, (X_N, Y_N)$ *(viewed as independent copies of the random pair* (X, Y)*), the objective is to select a distance measure on* $\mathcal{X}$ *among the collection* $D_M(x, x') = (x - x')M(x - x')^T$ *indexed by the set of* $d \times d$ *symmetric positive semi-definite (PSD) matrices* M*. A popular approach consists in minimizing the empirical risk defined by*

$$R_n(M) = \frac{2}{N(N-1)} \sum_{1 \leq i < j \leq N} \left[1 + (D_M(X_i, X_j) - b) \cdot \mathbb{I}\{Y_i = Y_j\}\right]_+ , \tag{2}$$

where $b \geq 0$ *is a threshold,* $[a]_+ = \max(0, a)$ *for any* $a \in \mathbb{R}$ *and* $\mathbb{I}\{\mathcal{E}\}$ *denotes the indicator function of any event* $\mathcal{E}$*. In statistics, such a risk functional is known as a one sample U-statistic of degree two (see [22]). Although the accuracy of ERM has been extensively studied in this context in recent years (see [19, 2, 7]), the computation of the empirical risk* (2) *requires a summation over* $O(N^2)$ *terms, which is quickly prohibitive as* N *grows.*

2.2 Review of Two "Smart" Sampling Schemes for SGD

We now describe two subsampling techniques, recently proposed as a remedy to the computational difficulties experienced when trying to learn high dimensional models such as those mentioned in Examples 2.1 and 2.2 from datasets of explosive size.

The first technique, originally proposed in [9] (see also [8]), consists in incorporating a subsampling scheme with unequal weights in the SGD method in order to compute, at each iteration, an estimate of the local (sub-)gradient with reduced variance called *Horvitz-Thompson gradient estimator.* Consider the situation where the empirical risk is a basic i.i.d. empirical average, so that the gradient estimate based on the whole data set $\mathcal{D}_N = \{Z_1, \ldots, Z_N\}$ takes the form $\nabla_\theta \hat{L}_N(\theta) = (1/N) \sum_{i=1}^N \nabla_\theta \psi(Z_i, \theta)$. In the simplest situation (*i.e.* Poisson sampling), the computation of the (unbiased) Horvitz-Thompson gradient estimator is based on a subsample obtained by picking independently observations in $\mathcal{D}_N$ with probability weights $(p_1, \ldots, p_N) \in]0, 1[^N$ (possibly depending on some auxiliary information). It is given by:

$$\nabla_\theta \tilde{L}_N(\theta) \overset{def}{=} \frac{1}{N} \sum_{i=1}^N \frac{\epsilon_i}{p_i(\theta)} \nabla_\theta \psi(Z_i, \theta),$$

where ϵ_i is the binary variable indicating whether Z_i belongs to the subsample ($\epsilon_i = 1$, which happens with probability $p_i(\theta)$) or not ($\epsilon_i = 0$). The expected size of the subsample is then $N_0 = \sum_{i=1}^N p_i \leq N$. Consistency and asymptotic normality of the parameter estimator $\hat{\theta}_T$ output by the Horvitz-Thompson SGD as the number of iterations grows to infinity has been established in [9]. The optimal choice for the $p_i(\theta)$'s in terms of asymptotic efficiency (*i.e.* minimum asymptotic variance) given the expected size N_0 (fixed in advance) is given by:

$$p_i^*(\theta) = N_0 \frac{||\nabla_\theta \psi(Z_i, \theta)||}{\sum_{j=1}^N ||\nabla_\theta \psi(Z_j, \theta)||}, \quad \text{for all } i \in \{1, \ldots, N\}. \tag{3}$$

However, this choice depends on $\nabla_\theta \hat{L}_N(\theta))$ whose computation is precisely what we would like to avoid. Fortunately, it has been shown that one can decrease the asymptotic variance of $\hat{\theta}_T$

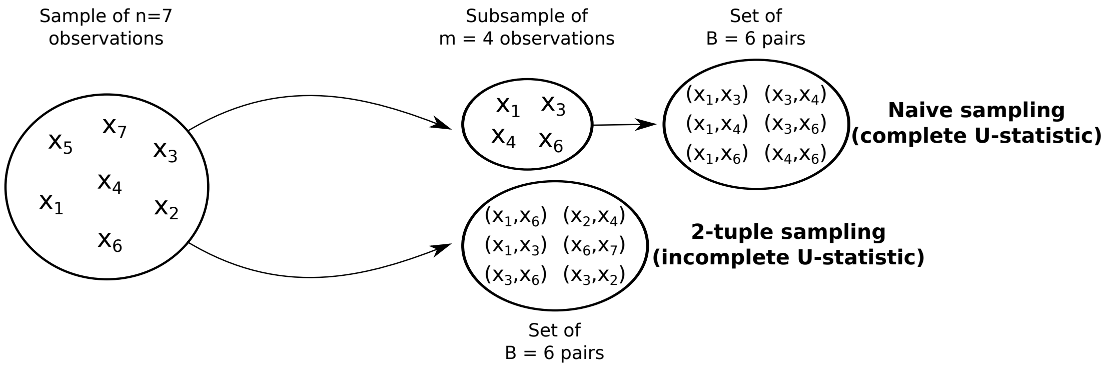

Sample of n=7 observations
x5 x7 x3 x4 x1 x2 x6
Subsample of m = 4 observations
x1 x3 x4 x6
Set of B = 6 pairs
(x1,x3) (x3,x4) (x1,x4) (x3,x6) (x1,x6) (x4,x6)
Naive sampling (complete U-statistic)
(x1,x6) (x2,x4) (x1,x3) (x6,x7) (x3,x6) (x3,x2)
2-tuple sampling (incomplete U-statistic)
Set of B = 6 pairs

Figure 1: Subsampling schemes and gradient estimation: complete U-statistic based on a subsample *vs* incomplete U-statistic

if p_i is positively correlated with p_i^* for $i = 1, \ldots, N$. Due to the specific form of the gradients for the logistic regression problem, we will use the probabilities weights $p_i^*(\theta) \sim N_0 \|X_i\|$. This choice presents several advantages as the weights can be computed during the preprocessing step, and drawing from this distribution can be done in advance. Note that sampling from a nonuniform discrete distribution takes $O(\log_2(N))$ operations [15], which is negligible for the datasets considered in Section 4.

The second technique to improve upon the usual SGD applies in the situation where the empirical risk is of the form of a U-statistics *i.e.* when its computation requires to sum up over k-tuples of the data instances, like in Example 2.2. Considering the case of pairs ($k = 2$) for simplicity, the idea promoted in [10] (see also [11]) is to replace the summation over $O(N^2)$ terms in the computation of the gradient of (2) by a summation over $O(N)$ pairs drawn independently with replacement in the set of all possible pairs of data instances, producing an estimate usually called an *incomplete U-statistic*. As revealed by the theoretical analysis presented in [10], such a procedure fully preserves the universal learning rate of the ERM (*i.e.* $O(1/\sqrt{N})$). In contrast, a more naive approach, consisting in forming all possible pairs from a subsample of size $O(\sqrt{N})$ (yielding also a number of pairs involved in the gradient estimate of order $O(N)$) selected uniformly at random leads to a significantly slower learning rate, *i.e.* of order $O(1/N^{1/4})$. Figure 1 depicts the difference between these undersampling schemes.

3 Implementation - Details and Issues

We now provide some details about the implementation of the SGD variants described in the previous section. For Example 2.1, we compare the performance of the usual SGD with mini-batches of size N_0 selected at random with that of the Horvitz-Thompson version, with N_0 as expected sample size. The extra computational cost of the latter variant is due to the evaluation of the weights at each iteration and the corresponding Bernoulli draws. However, we point out that the Poisson sampling stage can be easily parallelized, since the draws are independent. For Example 2.2, we compare the SGD-Incomplete described previously with $N_0(N_0 - 1)$ Monte Carlo draws among the collection of all possible pairs of points with the SGD-Complete strategy. The latter consists in uniformly drawing a sample of $N_0 \leq N$ data points without replacement and construct the complete U-statistic based on this reduced sample.

Combining smart sampling and distributed computing. An interesting line of research would be to investigate how to combine the SGD-based ERM procedures above with efficient distribution of the computational and memory costs across several machines (nodes) of a cluster to deal with extremely massive datasets. For instance, the Apache Spark[1] implementation, an open-source cluster computing framework, is particularly suitable for iterative algorithms. Unlike MapReduce [12] which heavily relies on disk I/O, Spark is based on a distributed memory architecture abstraction called *Resilient Distributed Dataset* (RDD) [25]. It allows much fewer reads and writes on disk as well as partial results caching across its memory. Fault tolerance is achieved through *lineage*, which consists in keeping track of how an RDD was built so that relevant parts can be reconstructed in the event of a node failure for instance. When the empirical risk is a simple average over training observations, the standard distributed version of SGD [28] is as follows. The data is first partitioned across the cluster nodes. At each iteration, each node takes a random sample of its local data to compute an estimate of the gradient in parallel and sends it to a master node, which next averages the gradient estimates received from all nodes, updates the parameter vector and sends it back to the computing nodes.

This simple approach breaks down when dealing with a risk function of the form of a U-statistic as in Example 2.2, since distributing all pairs of observations across machines is unfeasible even for a moderate sample size N as this requires $O(N^2d)$ memory in total. One should only distribute observations as in the standard case. Then, if each node is only allowed to sample from its local data (to construct the pairs involved in the local gradient estimates), the vast majority of the $N(N-1)/2$ possible pairs will never be involved in the gradient computation, increasing significantly the variance of the gradient estimates and thus jeopardizing the convergence rate. A possible strategy could be as follows: each node only samples from its local data to avoid a significant communication burden at each iteration, but every $T_{shuff} \geq 1$ iterations the entire statistical population of observations/pairs is randomly re-distributed across nodes to avoid convergence issues. The value of T_{shuff} can be used to adjust the trade-off between a proper uniform sampling with large network latency ($T_{shuff} = 1$) and restricted sampling with efficient communication ($T_{shuff} = +\infty$).

4 Experimental Results

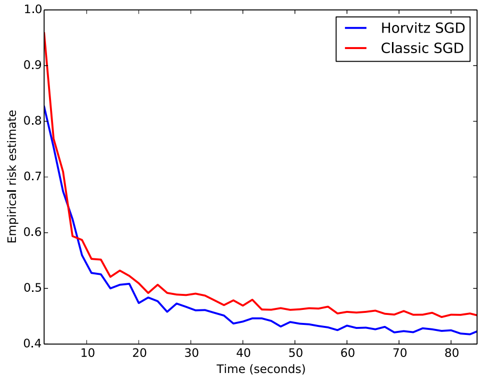
We now describe the numerical experiments we carried out to investigate the benefits of using the “smart sampling” techniques presented earlier to accelerate SGD convergence.

4.1 Poisson Sampling for Speeding-up Linear Logistic Regression

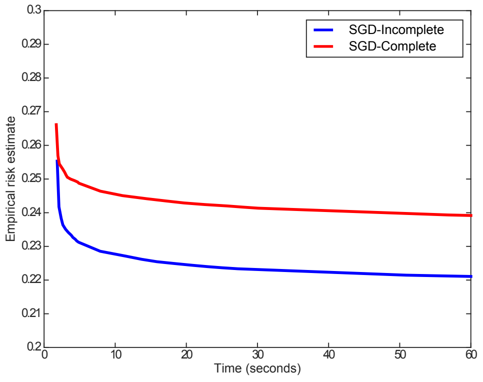
In the context of logistic regression, we evaluate the performance of the ”sampling with unequal weights” scheme presented earlier on the *Criteo Display Advertising Challenge* dataset, which was part of a Kaggle challenge on predicting ad click-through rate.[2] The training set consists of about 45M log entries, each of them corresponding to an ad display and whether it has been clicked or not. An ad display is represented by context features describing the user and the web page he/she is visiting: 13 count features and 26 categorical features that have been hashed onto bits, resulting in a total of $d = 845$ features. The test set has about 6M entries. More specifically, we consider the l_2-regularized logistic regression problem with the regularization parameter set to $1/N$ and $\gamma_t = \gamma_1/(1+\gamma_1\lambda t)$ as proposed in [6] , where γ_1 is determined using a small sample of the training set. As shown in Figure 2(a), Horvitz-Thompson SGD outperforms the classic SGD.

[1]`https://spark.apache.org/`

[2]`https://www.kaggle.com/c/criteo-display-ad-challenge/`



(a) Logistic regression on Criteo dataset ($N_0 = 1$)



(b) Metric learning on MNIST dataset ($N_0 = 23$)

Figure 2: Temporal evolution of the objective function throughout the execution of SGD.

4.2 Monte-Carlo Sampling for Accelerating Metric Learning

We used the *MNIST* dataset: a handwritten digit classification dataset which has 10 classes and consists of 60,000 training images and 10,000 test images.[3] This dataset has been used extensively to benchmark metric learning. As done by previous authors, we reduce the dimension from 784 to 164 using PCA so as to retain 95% of the variance, and normalize each sample to unit norm. Note that merely computing the empirical risk for a given matrix M involves averaging over 1.8×10^9 pairs. The variants of the SGD are implemented with a step size $\eta_t = 1/25t$ and $N_0 = 23$, leading to a total of $23 \times 22/2 = 253$ pairs of observations for each gradient estimate. As depicted by Figure 2(b), the SGD based on incomplete U-statistics estimates (Monte-Carlo sampling scheme) outperforms that based on complete U-statistics by a very large margin, although they use the same number of pairs for estimating the gradient.

5 Conclusion

In this paper, we have investigated the computational efficiency of some variants of SGD based on sampling schemes that are more sophisticated than *sampling without replacement*. Our experiments on two learning tasks confirm the relevance of these approaches: the extra cost induced by the implementation of such "smart sampling" schemes is negligible compared to the gain in gradient estimation accuracy. Hence, they provide a better trade-off between statistical accuracy and computational time. The next step would be to investigate whether these techniques remain beneficial when implemented on distributed computing platforms.

References

[1] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling Up Machine Learning.* Cambridge, 2011.

[2] A. Bellet and A. Habrard. Robustness and Generalization for Metric Learning. *Neurocomputing*, 151(1):259–267, 2015.

[3] A. Bellet, A. Habrard, and M. Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. Technical report, arXiv:1306.6709, June 2013.

[3]`http://yann.lecun.com/exdb/mnist/`

[4] P. Bianchi, S. Clémençon, J. Jakubowicz, and G. Moral-Adell. On-Line Learning Gossip Algorithm in Multi-Agent Systems with Local Decision Rules. In *IEEE Big Data*, 2013.

[5] L. Bottou. *Online Algorithms and Stochastic Approximations: Online Learning and Neural Networks*. Cambridge University Press, 1998.

[6] L. Bottou. Stochastic Gradient Tricks. In *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445. Springer, 2012.

[7] Q. Cao, Z.-C. Guo, and Y. Ying. Generalization bounds for metric and similarity learning. *arXiv preprint arXiv:1207.5437*, 2012.

[8] S. Clémençon, A. Bellet, and I. Colin. Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics. Technical report, arXiv:1501.02629, 2015.

[9] S. Clémençon, P. Bertail, and E. Chautru. Scaling-up M-estimation via sampling designs: the horvitz-thompson stochastic gradient descent. In *IEEE Big Data*, 2014.

[10] S. Clémençon, P. Bertail, E. Chautru, and G. Papa. Survey schemes for stochastic gradient descent with applications to m-estimation. Technical report, arXiv:1501.02218, 2015.

[11] S. Clémençon, S. Robbiano, and J. Tressou. Maximal deviations of incomplete U-statistics with applications to empirical risk sampling. In *SDM*, 2013.

[12] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[13] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction. In *ICML*, pages 713–720, 2011.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[15] L. Devroye. Sample-based non-uniform random variate generation. In *WSC*, pages 260–265, 1986.

[16] D. Donoho, A. Maleki, and A. Montanari. Message passing algorithms for compressed sensing. *PNAS*, 106(45):18914–18919, 2009.

[17] J. Duchi, J. A. Agarwal, and M. Wainwright. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. *IEEE TAC*, 99(10):1–40, 2010.

[18] N. Halko, P. Martinsson, Y. Shkolnisky, and M. Tygert. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific Computing*, 33(5):2580–2594, 2013.

[19] R. Jin, S. Wang, and Y. Zhou. Regularized distance metric learning: Theory and algorithm. In *NIPS*, pages 862–870, 2009.

[20] M. Jordan. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, 2013.

[21] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.

[22] A. J. Lee. *U-statistics: Theory and practice*. Marcel Dekker, Inc., New York, 1990.

[23] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. *PVLDB*, 2012.

[24] G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE TSP*, 58(10):5262–5276, 2010.

[25] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. In *NSDI*, pages 2–2, 2012.

[26] P. Zhao and Tong Z. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In *ICML*, 2015.

[27] P. Zhao and T. Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. Technical report, arXiv:1405.3080, 2014.

[28] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *NIPS*, pages 2595–2603, 2010.