Scalable Stochastic Gradient Descent with Improved Confidence

Sangkyun Lee
Department of Computer Science
University of Technology
Dortmund, Germany
sangkyun.lee@tu-dortmund.de

Christian Bockermann
Department of Computer Science
University of Technology
Dortmund, Germany
christian.bockermann@tu-dortmund.de

Abstract

Stochastic gradient descent methods have been quite successful for solving large-scale and online learning problems. We provide a simple parallel framework to obtain solutions of high confidence, where the confidence can be easily controlled by the number of processes, independently of the length of learning processes. Our framework is implemented as a scalable open-source software which can be configured for a single multicore machine or for a cluster of computers, where the training outcomes from independent parallel processes are combined to produce the final output.

1 Introduction

Stochastic online learning algorithms have been quite successful for large-scale and online learning problems [1, 9, 11]. They typically consist of inexpensive iterations, each involving a single input point or a tiny subset of training examples. Although a very large number of iterations might be required to obtain a solution with high accuracy, their solutions of moderate accuracy are often enough for learning purposes.

Due to the stochastic nature of the algorithms, we typically have to repeat learning processes until we observe a solution of satisfactory quality. Although algorithm runs can be carried out in parallel, the effort for the runs without acceptable outcome is nearly wasted. The chance of failure typically decreases as we use more iterations, but doing so is often undesirable in large-scale settings.

We provide a scalable online learning framework to utilize all independent repetitions of learning endeavor to obtain a solution with enhanced confidence, based on the stochastic gradient descent (SGD) algorithm. Our approach is implemented as an open-source Java software for training the support vector machines (SVMs), which can be configured for a single computer or multiple machines.

To simplify our notation, we use $\|\cdot\|$ to denote the Euclidean norm.

1.1 Confidence of a Single Stochastic Gradient Descent Run

In stochastic online learning, we assume that a continuous convex function $F(\cdot;\xi_t)$ is revealed at time t, which is determined by a random index $\xi_t \in \Xi = \{1,2,\dots\}$. For random variables $\xi_1,\xi_2,\dots,\xi_T$ sampled from a probability distribution P, we define the *regret* R_T as a function of decisions $w_1,w_2,\dots,w_T$ with respect to a single decision w^*:

$$R_T := \sum_{t=1}^{T} F(w_t;\xi_t) - \sum_{t=1}^{T} F(w^*;\xi_t). \tag{1}$$

We define the optimal decision w^* as a minimizer of the following risk minimization problem,

$$\min_{w\in\mathcal{W}} \; f(w) := \mathbb{E}\left[F(w;\xi)\right] = \int_{\Xi} F(w;\xi)\, dP(\xi) \tag{2}$$

where $\mathcal{W}$ is a compact convex set in $\mathbb{R}^p$ and the expectation is assumed to be well-defined and finite for every $w \in \mathcal{W}$. The goal of stochastic online learning is to find a sequence of decisions $w_1, w_2, \dots$ in $\mathbb{R}^p$ so that we have $\lim_{t\to\infty} \mathbb{E}[f(w_t)] - f(w^*) = 0$.

In the SGD algorithm, we obtain the next iterate w_{t+1} using the following update rule:

$$w_{t+1} := \Pi_{\mathcal{W}}\left(w_t - \eta_t G(w_t;\xi_t)\right), \;\; G(w_t;\xi_t) \in \partial F(w_t;\xi_t), \;\; t \ge 1, \tag{3}$$

where $\Pi_{\mathcal{W}}(\cdot)$ is an Euclidean project on $\mathcal{W}$. As in [6], we assume that $G(w;\xi)$ is an unbiased estimate of a subgradient of $f(w)$ for each $w \in \mathcal{W}$, that is, $\mathbb{E}[G(w;\xi)] \in \partial f(w)$. Note that w_{t+1} depends on $\xi_1, \xi_2, \dots, \xi_t$ but not on $\xi_{t+1}, \xi_{t+2}, \dots, \xi_T$.

For our analysis we define two constants D and V as follows,

$$D := \sup_{w\in\mathcal{W}} \|w - w_1\| \;\text{ and }\; V := \sup_{w\in\mathcal{W},\xi\in\Xi} \|G(w;\xi)\|.$$

Then we obtain a bound on the regret for the SGD algorithm [2, Appendix C].

Theorem 1. *A single run of the stochastic gradient descent algorithm using* (3)*, with a variable steplength $\eta_t = D/(V\sqrt{t})$ for each $t = 1, 2, \dots, T$, generates a realization of the sequence $w_1, w_2, \dots, w_T$ satisfying*

$$R_T \le 2\sqrt{2}DV\sqrt{T}, \;\; T = 1, 2, \dots.$$

The bound on the expected regret $\mathbb{E}[R_T]$ of the sequence $w_1, w_2, \dots, w_T$ is closely aligned with the convergence rate of an averaged iterate $\bar{w}_T = \frac{1}{T}\sum_{t=1}^T w_t$ in terms of the expected objective function value. We use this property to show the following result (see Appendix A for details):

Corollary 1 (Confidence of a Single Average)**.** *For $\bar{w}_T = \frac{1}{T}\sum_{t=1}^T w_t$, we have $\mathbb{E}[f(\bar{w}_T) - f(w^*)] \le 2\sqrt{2}\frac{DV}{\sqrt{T}}$ for $T \ge 1$. This implies that for a given error $\epsilon > 0$ and a confidence level $\beta \in (0,1)$,*

$$\mathbb{P}(f(\bar{w}_T) - f(w^*) \ge \epsilon) \le 1 - \beta \;\; \textit{when} \;\; T = \left\lceil \frac{8(DV)^2}{\epsilon^2(1-\beta)^2} \right\rceil. \tag{4}$$

That is, both accuracy and confidence of $\bar{w}_T$ from a single SGD run depend on the length T of the training process. Moreover, we would have to use quadratically larger value for T to increase the chance to obtain an ϵ-error solution.

1.2 Confidence of an Aggregate Stochastic Gradient Descent Run

Suppose that we have performed multiple *independent* SGD algorithm runs, each for a fixed time duration T (which is the number of examples used for each run in our case), producing M sequences of iterates $\{w_1^i, w_2^i, \dots, w_T^i\}$ and their averages $\bar{w}_T^i$ for $i = 1, 2, \dots, M$. We combine the averaged iterates to acquire an aggregate average $\bar{w}_M^\circ$:

$$\bar{w}_M^\circ = \frac{1}{M}\sum_{i=1}^M \bar{w}_T^i.$$

We show that $\bar{w}_M^\circ$ can provide a high confidence solution. The proof is in Appendix B.

Theorem 2 (Confidence of an Aggregate Average)**.** *Suppose that we have M independent SGD runs with decreasing steplengths $\eta_t = D/(V\sqrt{t})$ for $t = 1, 2, \dots, T$, to obtain $w_1^i, w_2^i, \dots, w_T^i$ and their averages $\bar{w}_T^i = \frac{1}{T}\sum_{t=1}^T w_t^i$ for $i = 1, 2, \dots, M$. For given $\epsilon > 0$ and $\beta \in (0,1)$, the aggregate average $\bar{w}_M^\circ = \frac{1}{M}\sum_{i=1}^M \bar{w}_T^i$ over the M runs satisfies*

$$\mathbb{P}\left(f(\bar{w}_M^\circ) - f(w^*) \ge \epsilon\right) \le 1 - \beta$$

when $T = \left\lceil \frac{32(DV)^2}{\epsilon^2} \right\rceil$ and $M = \left\lceil \frac{8(DV)^2}{\epsilon^2} \ln\left(\frac{1}{1-\beta}\right) \right\rceil$.

Note that T depends only on ϵ, and M depends on ϵ and β. Therefore the confidence of $\bar{w}_M^\circ$ can be controlled solely by the number of SGD runs M, for a fixed value of ϵ determined by T.

The resulting framework is essentially the same as the parallel SGD [12] and the worker/aggregator algorithm in [10], except for the fact that we allow diminishing stepsizes in addition to small constant ones demanded by the other two algorithms. Our confidence analysis in Theorem 2 provides an alternative quality measure of outcomes to the existing expected error bounds [12, Theorem 12].

2 Software

We implemented the SGD algorithm for the support vector machines as a part of our open-source *streams* package for online learning[1]. The streams package provides an abstraction layer for implementing various online learning algorithms.

2.1 A Thin Layer for Parallelization

We designed a thin layer implementing the *map-and-reduce* paradigm in our software to leverage modern multi-processor architectures. The map-and-reduce paradigm fits well for training with large volumes of data, using parallel processors on independent random partitions $X_1, X_2, \ldots, X_M$ of an input data X. This concept mainly derives from functional programming and has been adopted by Google and others such as the Apache Hadoop [2] system.

Figure 1 depicts the general concept of the map-and-reduce. It requires a map function m that can be computed on each partition of data, and a reduction function r combining the m outcomes from the map function evaluations. The map function for the SGD algorithm typically has a small memory footprint, and thus fits well even for a single multicore machine with limited memory.

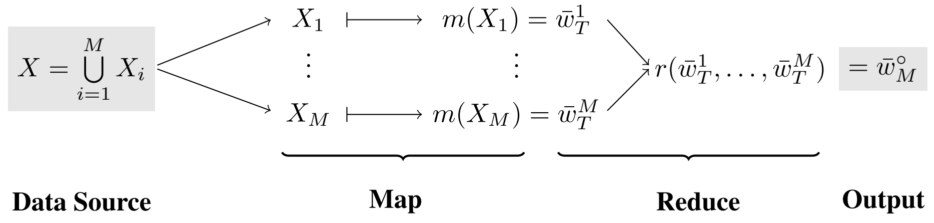
$X = \bigcup_{i=1}^{M} X_i$
$X_1 \longmapsto m(X_1) = \bar{w}_T^1$
$X_M \longmapsto m(X_M) = \bar{w}_T^M$
$r(\bar{w}_T^1, \ldots, \bar{w}_T^M)$
$= \bar{w}_M^\circ$
Data Source
Map
Reduce
Output

Figure 1: The map-and-reduce paradigm.

Our parallelization software layer allows for running online algorithms on a single multicore machine, or on a cluster of computers using the Hadoop Streaming APIs.

2.2 The Support Vector Machines

Consider a data source $X = \{(x_i, y_i) : x_i \in \mathbb{R}^{p-1}, y_i \in \mathbb{R}, i = 1, 2, \ldots, N\}$ of i.i.d. samples and its independent partitions $X_1, X_2 \ldots, X_M$. The support vector machines can be described using $w_t = (v_t, b_t)$ with $v_t \in \mathbb{R}^{p-1}$ and $b_t \in \mathbb{R}$, and

$$F((v_t, b_t); \xi_t) = \frac{\lambda}{2}\|v_t\|^2 + \max\{0, 1 - y_{\xi_t}(\langle v_t, \phi(x_{\xi_t})\rangle + b_t)\}, \quad \xi_t \in \{1, 2, \ldots, N\}.$$

Here we focus on linear kernels so that $\phi(x) = x$, but our framework can be easily extended for nonlinear kernels using random features [8]. Each SVM training via the SGD algorithm for a partition X_i serves as a map function m, producing $\bar{w}_T^i$, for $i = 1, 2, \ldots, M$. The reduction function r computes a simple average of $\bar{w}_T^1, \bar{w}_T^2, \ldots, \bar{w}_T^M$, producing an aggregate average $\bar{w}_M^\circ$.

[1] The *streams* package is available at `http://github.com/cbockermann/streams`.

[2] Available at `http://hadoop.apache.org`.

3 Experiments

For experiments we use three large data sets: (1) CCAT from the RCV1-v2 collection [3] (804414 examples, 47236 features), (2) URL from the malicious URL detection project [5] (2.4 million examples, 3.2 million features), and (3) MNIST, a binary task classifying digits $0 \sim 4$ against the rest, from an extended version of the original set [4] (8.1 million examples, 784 features).

We denote by `SGD-M` and `SGD-1` the aggregate SGD approach producing $\bar{w}_M^{\circ}$ and the single SGD approach, respectively. `SGD-M` consists of M independent and parallel SGD runs, each with a fixed number of examples (iterations) T. Therefore the total number of training examples used by `SGD-M` becomes $N = TM$. For CCAT we also compared its performance to `SGD-1` using the same amount of training data. All experiments are repeated 30 times with random permutations.

We illustrate the results in Figure 2. The performance is measured by prediction accuracy on test sets (about 20% of total), since it is of more practical interest than objective function values. The red curves with error bars show the variability of prediction accuracy of `SGD-M` for different M values. The training time of `SGD-1` increases linearly with N, but it remains almost constant for `SGD-M` when its M processes run concurrently.

The plot on the left shows that the variation of `SGD-M` tends to be smaller than that of `SGD-1` for each training size tried, and the former decays faster than the latter as M (or the corresponding N) increases. We see similar behavior for URL and MNIST on the right. Training time of `SGD-M` was less than 50 seconds on a 32-core machine (when $M \leq 32$) for both URL and MNIST with 1.28 million and 6.4 million examples, respectively. Our software managed with the high dimensionality of the URL set (3.2 million features) as well by utilizing the sparsity of input vectors.

An interesting observation for `SGD-M` is that not only the confidence but also the test accuracy of solutions tends to improve up to certain points with larger M. This phenomenon can be understood by the expected error bound of [12, Theorem 12], but improvement seems to be only marginal in our experiments.

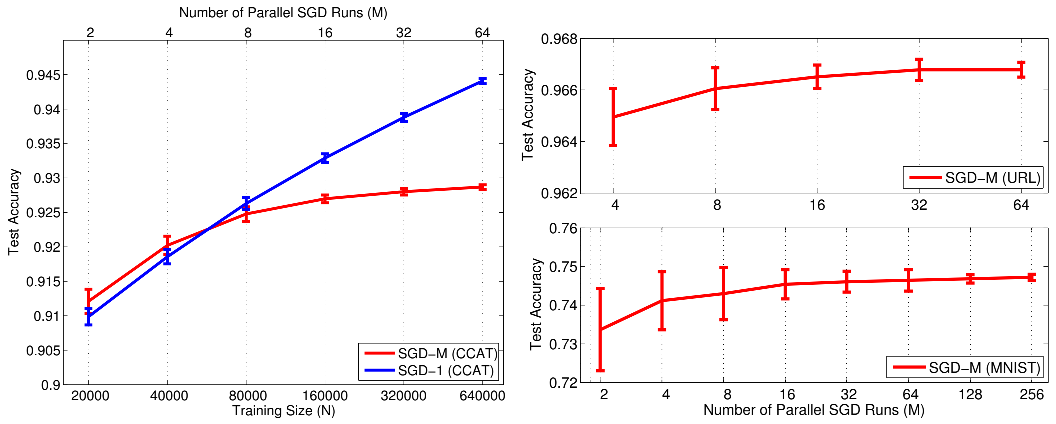

Figure 2: Prediction accuracy (mean and standard deviation) on separate test data sets of CCAT, URL, and MNIST. `SGD-M` utilizes M parallel SGD runs, each with T examples: $N = TM$ examples in total for each value of M. The value of T is chosen regarding the amount of available data: $T = 10000$ (CCAT), $T = 20000$ (URL), and $T = 25000$ (MNIST). For CCAT (the plot on the left), the performance of `SGD-1` (a single SGD) is measured as well (the blue curve) using the same number of examples N for each corresponding value of M. `SGD-1` failed to produce solutions for URL and MNIST in reasonable time.

Acknowledgments

The authors acknowledge the support of the German Research Foundation (DFG) grant for the Collaborative Research Center SFB 876: "Providing Information by Resource-Constrained Data Analysis" (`http://sfb876.tu-dortmund.de`).

References

[1] L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[2] S. Lee and S. J. Wright. Manifold identification of dual averaging algorithm for regularized stochastic online learning. Technical report, University of Wisconsin-Madison, April 2011.

[3] D. D. Lewis, Y. Yang, T. G. Rose, G. Dietterich, F. Li, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[4] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*, pages 301–320. MIT Press, 2007.

[5] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 681–688, 2009.

[6] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[7] Y. Nesterov and J.-P. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559 – 1568, 2008.

[8] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, pages 1177–1184. MIT Press, 2008.

[9] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

[10] M. Weimer, S. Rao, and M. Zinkevich. A convenient framework for efficient parallel multipass algorithms. In *LCCC : NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[11] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[12] M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, volume 23, pages 2595–2603. MIT Press, 2010.

A Confidence Bound for a Single Average

This provides a proof for Corollary 1.

Proof. From the convexity of f we have

$$\begin{aligned}\mathbb{E}[f(\bar{w}_T)] - f(w^*) &\leq \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} f(w_t) - \sum_{t=1}^{T} f(w^*)\right] \\ &= \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\mathbb{E}[F(w_t;\xi_t) \mid \xi_1,\xi_2,\ldots,\xi_{t-1}] - \sum_{t=1}^{T}\mathbb{E}[F(w^*;\xi_t) \mid \xi_1,\xi_2,\ldots,\xi_{t-1}]\right] \\ &= \frac{1}{T}\mathbb{E}[R_T]\end{aligned}$$

Combining the inequality with the regret bound in Theorem 1 leads to the first part of the claim. Applying Markov's inequality gives the second claim. □

B Confidence Bound for an Aggregate Average

This is the proof of Theorem 2, which utilizes a standard concentration bound and happens to be much simpler than the analysis for the expected error bounds in [12]. We borrow several ideas from [7] but refines the analysis therein.

Proof. First we define random variables $\zeta_i := f(\bar{w}_T^i) - f(w^*)$ for $\bar{w}_T^i = \frac{1}{T}\sum_{t=1}^T w_t^i$, $i = 1, 2, \ldots, M$. Then $0 \leq \zeta_i \leq 2DV$ with probability one, since w^* is a minimizer of $f(\cdot)$ and the convexity of $f(\cdot)$ implies that for $g(w_t^i) \in \partial f(w_t^i)$,

$$\zeta_i \leq \frac{1}{T}\sum_{t=1}^T f(w_t^i) - f(w^*) \leq \frac{1}{T}\sum_{t=1}^T g(w_t^i)^T(w_t^i - w^*) \leq \frac{1}{T}\sum_{t=1}^T \|g(w_t^i)\|\|w_t^i - w^*\| \leq 2DV.$$

Using the convexity of $f(\cdot)$ again, we have

$$f(\bar{w}_M^\circ) - f(w^*) \leq \frac{1}{M}\sum_{i=1}^M f(\bar{w}_T^i) - f(w^*) = \frac{1}{M}\sum_{i=1}^M \zeta_i =: \bar{\zeta}.$$

This implies that

$$\mathbb{P}\left(f(\bar{w}_M^\circ) - f(w^*) \geq 2\sqrt{2}\frac{DV}{\sqrt{T}} + \delta\right) \leq \mathbb{P}\left(\bar{\zeta} \geq 2\sqrt{2}\frac{DV}{\sqrt{T}} + \delta\right) \leq \mathbb{P}(\bar{\zeta} - \mathbb{E}[\bar{\zeta}] \geq \delta) \leq \exp\left(-\frac{\delta^2 M}{2(DV)^2}\right),$$

where the second inequality is from the fact that $\mathbb{E}[\bar{\zeta}] \leq \max_{i=1,2,\ldots,M} \mathbb{E}[\zeta_i] \leq 2\sqrt{2}\frac{DV}{\sqrt{T}}$ by Corollary 1, and the last one is due to the Hoeffding's inequality applied for the bounded independent random variables $\zeta_1, \zeta_2, \ldots, \zeta_M$. Replacing $\delta = \epsilon/2 = 2\sqrt{2}\frac{DV}{\sqrt{T}}$ gives the following, leading to the claim.

$$\mathbb{P}\left(f(\bar{w}_M^\circ) - f(w^*) \geq \epsilon\right) \leq \exp\left(-\frac{\epsilon^2 M}{8(DV)^2}\right) =: 1 - \beta. \tag{5}$$

□