Federated Optimization: Distributed Optimization Beyond the Datacenter

Jakub Konečný*
School of Mathematics
University of Edinburgh
J.Konecny@sms.ed.ac.uk

H. Brendan McMahan
Google, Inc.
Seattle, WA 98103
mcmahan@google.com

Daniel Ramage
Google, Inc.
Seattle, WA 98103
dramage@google.com

Abstract

We introduce a new and increasingly relevant setting for distributed optimization in machine learning, where the data defining the optimization are distributed (unevenly) over an extremely large number of nodes, but the goal remains to train a high-quality centralized model. We refer to this setting as *Federated Optimization*. In this setting, communication efficiency is of utmost importance.

A motivating example for federated optimization arises when we keep the training data locally on users' mobile devices rather than logging it to a data center for training. Instead, the mobile devices are used as nodes performing computation on their local data in order to update a global model. We suppose that we have an extremely large number of devices in our network, each of which has only a tiny fraction of data available totally; in particular, we expect the number of data points available locally to be much smaller than the number of devices. Additionally, since different users generate data with different patterns, we assume that no device has a representative sample of the overall distribution.

We show that existing algorithms are not suitable for this setting, and propose a new algorithm which shows encouraging experimental results. This work also sets a path for future research needed in the context of federated optimization.

1 Introduction and Problem Formulation

The optimization community has seen an explosion of interest in solving problems with finite-sum structure in recent years. In general, the objective is formulated as

$$\min_{w\in\mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w). \tag{1}$$

The main source of motivation are problems arising in machine learning. The problem structure (1) covers linear or logistic regressions, support vector machines, but also more complicated algorithms such as conditional random fields or neural networks. The functions f_i are typically specified via a loss function ℓ dependent on a pair of input-output data point $\{x_i, y_i\}$, $f_i(w) = \ell(w; x_i, y_i)$.

The amount of data that businesses, governments and academic projects collect is rapidly increasing. Consequently, solving the problem (1) is becoming impossible with a single node, as merely storing the whole dataset on a single node becomes infeasible. This gives rise to the distributed optimization setting, where we solve the problem (1) even though no single node has direct access to all the data.

In the distributed setting, communication cost is by far the largest bottleneck, as exhibited by a large amount of existing work. Further, many state-of-the-art optimization algorithms are inherently sequential, relying on a large number of

*Work performed while at Google, Inc.

very fast iterations. The problem stems from the fact that if one needs to perform a round of communication after each iteration, practical performance drops down dramatically, as the round of communication is much more time-consuming than a single iteration of the algorithm.

Works including [2, 8, 3, 9] have established basic theoretical results and developed distributed versions of existing methods, but as this is a relatively new area, many open questions remain. Recently, the idea of communication efficient algorithms has gained traction. Such algorithms perform a large amount of computation locally, before each round of communication, ideally balancing the cost of computation and communication[10, 11, 4, 7, 14]. There has been an attempt at a thorough understanding of communication lower bounds [1], but the authors note that significant gaps remain in a number of settings.

2 Federated Optimization — The Challenge of Learning from Decentralized Data

With the exception of CoCoA [4, 7], the existing work concerning communication efficient optimization [10, 14] and/or statistical estimation [13, 12] presupposes, and in fact requires, that the number of nodes is (much) smaller than the number of datapoints available to each node, and that each node has access to a random sample from the same distribution, and typically that each node has an identical number of data points.

We are hoping to bring to attention a new and increasingly relevant setting for distributed optimization, where typically none of the above assumption are satisfied, and communication efficiency is of utmost importance. In particular, algorithms for federated optimization must handle training data with the following characteristics:

- **Massively Distributed**: Data points are stored across a very large number of nodes K. In particular, the number of nodes can be much bigger than the average number of training examples stored on a given node (n/K).
- **Non-IID**: Data on each node may be drawn from a different distribution; that is, the data points available locally are far from being a representative sample of the overall distribution.
- **Unbalanced**: Different nodes may vary by many orders of magnitude in the number of training examples they hold.

In this work, we are particularly concerned with **sparse** data, where some features occur on only a small subset of nodes or data points. We show that the sparsity structure can be used to develop an effective algorithm for federated optimization.

We are motivated by the setting where training data lives on users' mobile devices (phones and tablets), and the data may be privacy sensitive. The data $\{x_i, y_i\}_{i=1}^n$ are generated by device usage, e.g. interaction with apps. Examples include predicting the next word a user will type (language modeling for smarter keyboard apps), predicting which photos a user is most likely to share, or predicting which notifications are most important. To train such models using traditional distributed learning algorithms, one would collect the training examples in a centralized location (data center) where it could be shuffled and distributed evenly over compute nodes. Federated optimization provides an alternative model potentially saving significant network bandwidth and providing additional privacy protection. In exchange, users allow some use of their devices' computing power.

Communication constraints arise naturally in the massively distributed setting, as network connectivity may be limited (e.g., we may wish to defer all communication until the mobile device is charging and connected to a wi-fi network). Thus, in realistic scenarios we may be limited to only a single round of communication per day. This implies that, within reasonable bounds, we have access to essentially unlimited local computational power. As a result, the practical objective is solely to minimize the rounds of communication.

Formally, let K be the number of nodes. Let $\{\mathcal{P}_k\}_{k=1}^K$ denote a partition of data point indices $\{1, \ldots, n\}$, so $\mathcal{P}_k$ is the set of data points stored on node k, and define $n_k = |\mathcal{P}_k|$. We can then rephrase the objective (1) as a linear combination of the local empirical objectives $F_k(w)$, defined as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \stackrel{\text{def}}{=} \sum_{k=1}^{K} \frac{n_k}{n} \cdot \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

3 An Algorithm for Federated Optimization

The main motivation for the algorithm we propose comes from a perhaps surprising connection between two algorithms — SVRG [5, 6] and DANE [10]. SVRG is a stochastic method with variance reduction for solving the problem (1) on a single node. DANE is an algorithm for distributed optimization which on every iteration solves exactly a new subproblem on each node based on the local data and the gradient of the entire function f.

One could modify the DANE algorithm in the following way: instead of solving the DANE subproblem exactly, use any optimization algorithm to produce an approximate solution. In particular, if we use SVRG as the local solver, the sequence of updates is *equivalent* to the following version of distributed SVRG.

Algorithm 1 A distributed version of SVRG

1: **parameters:** m = # of stochastic steps per epoch, h = stepsize, data partition $\{\mathcal{P}_k\}_{k=1}^K$, starting point $\tilde{w}_0$
2: **for** $s = 0, 1, 2, \ldots$ **do** ▷ Overall iterations
3: Compute $\nabla f(\tilde{w}_s) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{w}_s)$
4: **for** $k = 1$ to K **do in parallel** over nodes k ▷ Distributed loop
5: Initialize: $w_k = \tilde{w}_s$
6: **for** $t = 1$ to m **do** ▷ Actual update loop
7: Sample $i \in \mathcal{P}_k$ uniformly at random
8: $w_k \leftarrow w_k - h\left(\nabla f_i(w_k) - \nabla f_i(\tilde{w}_s) + \nabla f(\tilde{w}_s)\right)$
9: **end for**
10: **end for**
11: $\tilde{w} \leftarrow \tilde{w} + \frac{1}{K}\sum_{k=1}^K (w_k - \tilde{w})$ ▷ Aggregate updates from nodes
12: **end for**

The algorithm 1 indeed works very well in many "simple" settings, but fails in the challenging setting of federated optimization, particularly with sparse data. In order to make the algorithm more robust, we modify it in a number of ways. The most important changes include

1. Flexible stepsize h_k — different for each node, inversely proportional to size of the local data, n_k
2. Scaling of stochastic updates by a diagonal matrix S_k (defined below). Step 8 in Algorithm 1 is replaced by $w_k = w_k - h_k\left(S_k\left[\nabla f_i(w_k) - \nabla f_i(\tilde{w})\right] + \nabla f(\tilde{w})\right)$
3. The aggregation procedure is adaptive to the data distribution. For some diagonal matrix A (defined below), the step 11 in Algorithm 1 is replaced by $\tilde{w} = \tilde{w} + A\sum_{k=1}^K \frac{n_k}{n}(w_k - \tilde{w})$

The matrices S_k, and A concern sparsity patterns in the data, and are identities in the case of fully dense data. To motivate their use, imagine that the data are distributed in the following way. All of the data points that have non-zero value of a certain feature are stored on a single node k. Consequently, computing stochastic gradient $\nabla f_i(w_k)$ will greatly overestimate the gradient in this direction, because it appears much more frequently on node k than in the entire dataset. Diagonal elements of the matrix S_k are ratios of frequencies of appearance of each feature globally and locally. One can interpret this as scaling the estimates so that they are of the correct magnitude in expectation, taken *conditioned* on knowledge of the distribution of the sparsity patterns.

In order to motivate use of the matrix A, let us assume our problem is separable in the sense that each data point depends only on features belonging to one of a few disjoint clusters of coordinates, and data are distributed according to these clusters. In the case of linear predictors, we could simply solve the problems independently, and add up the results, instead of averaging, to obtain the optimal solution. Although this in not the case in reality, we try to interpolate, and average updates for features that appear on every node, while we take longer steps for features that appear only on a few nodes. Formally, $A_{ii} = K/\omega_i$, where ω_i is the number of nodes on which feature i is present. Although this is a heuristic modification, it's omission greatly degrades performance of the algorithm.

4 Experiments

The dataset presented here was generated based on public posts on the Google+ network. We randomly picked $10,000$ authors that have at least 100 public posts in English, and try to predict whether a post will receive at least one comment (that is, a binary classification task).

We split the data chronologically on a per-author basis, taking the earlier 75% for training and the following 25% for testing. The total number of training examples is $n = 2,166,693$. We created a simple bag-of-words language model, based on the $20,000$ most frequent words. We then use a logistic regression model to make a prediction based on these features.

We shape the distributed optimization problem as follows. Suppose that each user corresponds to one node, resulting in $K = 10,000$. The average n_k, number of data points on node k, is thus approximately 216. However, the actual numbers n_k range from 75 to $9,000$, showing the data distribution is in fact substantially unbalanced. It is natural to expect that different users can exhibit very different patterns in the data generated. This is indeed the case, and hence the distribution to nodes cannot be considered an IID sample from the overall distribution.

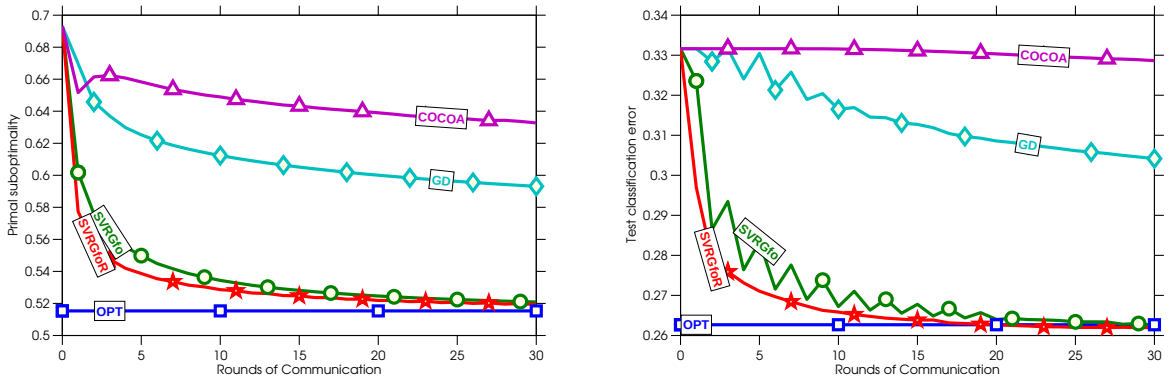

Figure 1: Rounds of communication vs. objective function (left) and test prediction error (right).

In Figure 1, we compare various optimization algorithms. Two other communication efficient algorithms, DANE [10] and DiSCO[14], diverge in this setting, and so are not included in the figure.

- The blue squares (OPT) represent the best possible offline value (the optimal value of the optimization task in the first plot, and the test error corresponding to the optimum in the second plot).
- The teal diamonds (GD) correspond to a simple distributed gradient descent.
- The purple triangles (COCOA) are for the CoCoA algorithm [4].
- The green circles (SVRGfo) give values for our proposed algorithm.
- The red stars (SVRGfoR) correspond to the same algorithm applied to the same problem with randomly reshuffled data. That is, we keep the unbalanced number of examples per node, but populate each node with randomly selected examples.

The experiments point at several important realities. First, existing communication efficient algorithms are completely inefficient for federated optimization. The only algorithm that converges, CoCoA, does so at a slower rate than a naive benchmark — distributed gradient descent. Our algorithm, DSVRG, reaches optimality in very few communication rounds, despite the challenging setting of federated optimization. Furthermore, very little performance is lost compared to the setting where the data are randomly reshuffled between nodes, which highlights the robustness of our proposed method to non-IID data.

We argue that the setting of federated optimization will be increasingly important for practical application, as mobile devices get computationally more powerful and privacy issues ever more pressing. A large number of question remain open, creating potential for a line of further work. Most importantly, there is no readily available dataset (from usual sources as libsvm or UCI repository) with naturally user-clustered data. Creation of a new, public dataset is crucial to lower the barrier of further engagement of the academic community. More rigorous experiments remain to be done, both on larger datasets and on more challenging problems, such as deep learning. Currently, there is no proper theoretical justification of the method, which would surely drive further improvements. Lastly, in order for this shift to "on-device intelligence" to truly happen, methods from differential privacy need to be incorporated and many parts of practical machine learning pipelines need to be redesigned.

References

[1] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. *arXiv preprint arXiv:1506.01900*, 2015.

[2] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *arXiv preprint arXiv:1204.3514*, 2012.

[3] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takác. Fast distributed coordinate descent for non-strongly convex losses. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[4] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[5] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[6] Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[7] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. *arXiv preprint arXiv:1502.03508*, 2015.

[8] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059*, 2013.

[9] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 850–857. IEEE, 2014.

[10] Ohad Shamir, Nathan Srebro, and Tong Zhang. Communication efficient distributed optimization using an approximate newton-type method. *arXiv preprint arXiv:1312.7853*, 2013.

[11] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

[12] Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013.

[13] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

[14] Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*, 2015.