Towards Geo-Distributed Machine Learning

Ignacio Cano∗∗
Markus Weimer∗
Dhruv Mahajan∗
Carlo Curino∗
Giovanni Matteo Fumarola∗

∗University of Washington
icano@cs.washington.edu
∗∗Microsoft
{mweimer, dhrumaha, curino, gifuma}@microsoft.com

Abstract

In large organizations, data is “born” in data centers all around the world. Learning requires a global view of such data. This new class of geo-distributed machine learning (GDML) applications need to cope with: 1) scarce and expensive cross-data center bandwidth, and 2) growing privacy concerns that are pushing for stricter data sovereignty regulations. In this paper, we formalize this problem, show that the current state-of-the-art lacks proper support for GDML applications, and propose an initial system and algorithm that perform training in a geo-distributed fashion. Our empirical evaluation confirms the general validity of our approach, but many research challenges remain open.

1 Introduction

Modern organizations have a planetary footprint. Data is born where users and systems are located, all around the globe. On the other hand, many machine learning applications require access to all the data at once to achieve the best results. For example, fraud prevention benefits tremendously from the global picture in both finance and communication networks, recommender systems rely on the maximum breadth of data to overcome cold start problems, and the predictive maintenance revolution is only possible because it relies on data from all markets. This type of applications form a new class of learning problems, which we call Geo-Distributed Machine Learning (GDML).

The geographically distributed nature of the data introduces two new fundamental challenges: 1) cross data center (X-DC) connectivity is scarce, expensive, and less reliable than intra data center (in-DC) connectivity, and 2) increasing data sovereignty regulations limit where data can be stored/processed (e.g., German’s citizen data cannot be stored outside EU).

Recent literature has described similar problems in the context of relational analytics workloads [1,2], general big-data applications [3], and streaming systems [4]. To the best of our knowledge, no previous work has dealt with this geo-distributed problem in iterative/convergence-oriented applications such as machine learning. Since no practical system supports GDML today, practitioners resort to centralizing the problem by copying all the data into a single data center, where training is performed.

In this paper, we show that the novel challenges of GDML render the centralized state-of-the-art approach too costly in terms of X-DC bandwidth for large datasets, and infeasible when subject to strict sovereignty constraints. We speculate that both challenges will persist or grow in the future [3,5], highlight some of the shortcomings of the current practice, and propose a system that performs distributed training across data centers without moving raw data.

∗This work was done while the author was interning at Microsoft.
Contribution Our system builds upon Apache Hadoop/YARN and Apache REEF, and is capable of coordinating learning tasks running in separate data centers to construct a unified model. We offset the generally communication-intensive nature of machine learning algorithms by employing and extending communication-sparse ones [6]. Our experimental evaluation indicates that our approach can outperform the state-of-the-art by several orders of magnitude when measuring X-DC transfers, as well as respect stricter sovereignty constraints. Finally, we highlight several challenges that remain open for GDML applications.

2 Problem Formulation

We consider the $l_2$ regularized linear classification problem. We assume a dataset $D$ of $N$ examples $(x_i, y_i)$ where $x_i \in \mathbb{R}^d$ denotes the features and $y_i \in \{-1, 1\}$ denotes the label of example $i$. Further, we assume that $D$ is randomly partitioned across $P$ data centers. The portion of the dataset $D$ hosted by data center $p$ is denoted as $D_p$.

Let $l(w \cdot x_i, y_i)$ be a continuously differentiable loss function with Lipschitz continuous gradient, where $w \in \mathbb{R}^d$ is the weight vector. Let $L_p(w) = \sum_{i \in D_p} l(w \cdot x_i, y_i)$ be the loss associated with data center $p$, and $L(w) = \sum_p L_p(w)$ be the total loss over all data centers. Our goal is to find $w$ that minimizes the following objective function, which decomposes per data center:

$$f(w) = \frac{\lambda}{2} ||w||^2 + L(w) = \frac{\lambda}{2} ||w||^2 + \sum_p L_p(w)$$

(1)

where $\lambda > 0$ is the regularization constant.

We aim to optimize this objective function while keeping the data in place, which poses two interesting challenges: a) we need an algorithm that minimizes X-DC communication, and b) we need a system that allows such an algorithm to be implemented, especially considering the fault tolerance and network latency characteristics of such setup.

The strong assumption we made in this problem definition on random partitioning of the data holds true in some important production use cases we observe. In such cases, load balancing across data centers forces data to be “randomly” spread across them, independently of the learning task. However, this is not fully general, as other important GDML workloads require data to be close to the users (to achieve low latency interactions), thus strong geographically biases emerge. Supporting this second class of workloads is still an open problem.

3 Algorithm

We need an algorithm capable of minimizing X-DC communication costs. The Terascale method [7] is one of the most representative methods from the statistical query model class (SQM) [8] and is considered a state-of-the-art solver. Alternating Direction Method of Multipliers (ADMM) [9][10] is a popular dual method that solves approximate problems in the nodes and iteratively reaches the full batch solution.

Recently, many communication-efficient algorithms have been proposed that trade-off local computation with communication. CoCoA [11] represents the class of distributed dual methods that, in each outer iteration, solve (in parallel) several local dual optimization problems. In this work, we use the algorithm proposed by Mahajan et al. [6] to optimize Equation (1). Experiments show that this method performs better than the aforementioned ones, both in terms of communication passes and running time [6].

The main idea of the algorithm is to trade-off in-DC computation and communication with X-DC communication. Let $w^r$ and $g^r$ be the global model and gradient respectively at iteration $r$ available in all data centers. At data center $p$, this information is used together with the local data $D_p$ to construct an approximation $\hat{f}_p$ of $f$. To ensure convergence, $\hat{f}_p$ should satisfy a gradient consistency condition, $\nabla \hat{f}_p(w^r) = g^r$. The function $\hat{f}_p$ is approximately optimized to get the local weight vector $w_p$, which enables the computation of the direction $d_p = w_p - w^r$. The global update direction $d^r$
is chosen to be $d^r = \frac{1}{p} \sum_p d_p$. A line search is then performed along the direction $d^r$ to find the
next point $w^{r+1} = w^r + t d^r$.

Among the possible choices suggested in [6] for $\hat{f}_p$, we consider the following quadratic approximation in this work:

$$\hat{f}_p (w) = \frac{1}{2} ||w||^2 + g^r \cdot (w - w^r) + \frac{P}{2} (w - w^r)^T H^r_p (w - w^r)$$

(2)

where $H^r_p$ is the Hessian of $L_p$ at $w^r$ and is computed using the data $D_p$ available at data center $p$.

In each iteration, the computation of the gradient $g^r$ and the direction $d^r$ requires communication
across data centers. Since each data center has the global approximate view of the full objective
function, the number of iterations required are significantly less than traditional SQM methods,
resulting in orders of magnitude improvements in terms of X-DC communication.

4 System

Our system implements the algorithm above, and is built using Apache Hadoop/YARN and REEF.
YARN is being extended with a notion of federation\(^2\) which provides a single-cluster image across
multiple clusters\(^3\). We leverage these mechanisms to support computations spanning geographically
disperse data centers. Apache REEF provides us with the basic centralized control flow and the
group communication primitives. As part of this work, we extend REEF to support federated YARN,
including scheduling of resources to particular data centers.

The algorithm described in §3 can be implemented using BROADCAST and REDUCE operators
alone, which are commonly available in communication trees. However, in order to support the
constraints for physically separated data centers, where X-DC communication links are more ex-
pensive than in-DC links, we need multi-level communication trees. We therefore extend REEF’s
group communications library to be able to form these efficient cross-data center communication
structures.

Figure 1 shows an example of the multi-level communication tree we use. The data center masters,
represented by their respective nodes $M_P$, together with the global master $M^G$, form the global level
of the tree. The slave nodes within each data center and their masters $M_P$ form the local levels. A
BROADCAST originates from $M^G$ to the data center masters $M_P$, which in turn broadcast to the
slave nodes in their own data centers. Conversely, REDUCE operations originate on those slave
nodes, while the data center masters $M_P$ aggregate the data prior to sending it to $M^G$ for global
aggregation.

5 Evaluation

5.1 Experimental Setup

To assess our contributions we use the splice dataset for human splice site recognition [12]. It
consists of 50M examples with 47K sparse features, for a total of 200GB on disk.

\(^2\) Allows to map multiple autonomous systems into a single “federated” one.
\[^3\] https://issues.apache.org/jira/browse/YARN-2915
5.2 Methods

We compare our solution to several variants of the baseline approach, and propose different flavors of centralized and distributed methods. The former train models in a single data center while the latter perform X-DC training. The scarce resource to observe is X-DC transfer cost.

Our method, called Distributed-Enhanced, uses the multi-level master/slave tree that allows X-DC learning (Figure 1) and runs the algorithm described in §3. We consider the following baselines: the state-of-the-art Centralized, where we copy all the data to one data center prior to training, and Centralized-Quota where we do the same, but cap the amount of data that can be transferred to match the amount consumed by our solution. Both Centralized and Centralized-Quota use a single-level master/slave communication tree (e.g. the DC-P subtree in Figure 1) as they run in a single data center.

We also compare our method to the Distributed and Distributed-Quota baselines. As in Distributed-Enhanced, these baselines leave the data in place and train in a geo-distributed fashion. Distributed uses the multi-level master/slave tree for X-DC learning, and runs TRON without the communication-efficient algorithm explained in §3. Distributed-Quota does the same, but stops training after it reaches the X-DC transfer budget used by Distributed-Enhanced. The comparison between Distributed and Distributed-Enhanced better highlights the difference between the system (multi-level master/slave tree for X-DC learning) and the algorithm (communication-efficient) wins.

6 Results and Discussion

6.1 X-DC transfer

Figure 2 illustrates the X-DC transfer (normalized with respect to the dataset size) of the different methods for different numbers of data centers. Our method performs almost 3 orders of magnitude better than the state-of-the-art (Centralized) in every scenario, achieving the biggest difference (~4 orders of magnitude) for 2 data centers. In this setting, Centralized transfers ~100GB (50% of the dataset) through the X-DC link, whereas Distributed-Enhanced just needs 14MB (less than 0.01% of the dataset) worth of transfers to train the model. Distributed also outperforms the current practice, Centralized. The quota versions are not shown in this plot because their X-DC transfers are the same as Distributed-Enhanced.

---

4Experiments on a real distributed deployment across different data centers are left to future work.
5Mahajan et al. [6] also show that TRON performs better than other popular SQM method, LBFGS, used in the Terascale method [7].
6.2 AUC

Table 1 shows the AUC in the test set for all methods. Centralized and Distributed achieve the same AUC, as they run the same algorithm on the same data. Distributed-Enhanced, matches their AUC, which is remarkable considering that the X-DC transfer rate is orders of magnitude smaller. The alternatives with the same X-DC transfer as Distributed-Enhanced, Centralized-Quota and Distributed-Quota, perform worse. In the case of Centralized-Quota, less training data is available when the number of data centers increases, therefore, worse models are learned. In Distributed-Quota, as the X-DC transfer quota allowed is small, the optimization runs for few iterations, producing a model that is still far from the optimal.

<table>
<thead>
<tr>
<th>Method</th>
<th>2 DC</th>
<th>4 DC</th>
<th>8 DC</th>
<th>16 DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>0.6660174</td>
<td>0.6660174</td>
<td>0.6660174</td>
<td>0.6660174</td>
</tr>
<tr>
<td>Centralized-Quota</td>
<td>0.6652307</td>
<td>0.6642873</td>
<td>0.6557136</td>
<td>0.6417300</td>
</tr>
<tr>
<td>Distributed</td>
<td>0.6660174</td>
<td>0.6660174</td>
<td>0.6660174</td>
<td>0.6660174</td>
</tr>
<tr>
<td>Distributed-Quota</td>
<td>0.5696233</td>
<td>0.5696233</td>
<td>0.5422752</td>
<td>0.5696233</td>
</tr>
<tr>
<td>Distributed-Enhanced</td>
<td>0.6661202</td>
<td>0.6661884</td>
<td>0.6661213</td>
<td>0.6662581</td>
</tr>
</tbody>
</table>

Table 1: AUC on the test set of the different methods

6.3 Training Loss/Bandwidth trade-offs

Figure 3 shows the relative training loss over time as a function of the X-DC transfer for 16 data centers. X-DC transfers remain constant in Centralized and Centralized-Quota methods as they start after they copy the data. Distributed-Quota loss follows the same shape as Distributed but stops early due to the X-DC quota. Distributed-Enhanced achieves lower training losses much sooner in terms of X-DC transfers, which means that our method can get some meaningful results faster. If we don’t need a very accurate model (e.g. $10^{-2}$ relative training loss), our approach also gives a quicker response.

7 Conclusion and Future Work

In addition to its volume and velocity, research on planetary scale machine learning has to concern itself with the geographic distribution of the training data. Here, we adapted and implemented a first geo-distributed learning algorithm and evaluated it in a simulated setting. We showed that the state-of-the-art—copy the data to a central data center for learning—is not always optimal, even when ignoring data sovereignty issues.

By challenging the current practice, this result provides motivation to address further questions related to geo-distributed learning, such as: 1) fault-tolerance: WAN network connection disruption can lead to temporarily unavailable data from certain DCs, 2) latency: exploring the trade-off between bandwidth and time-to-insight, 3) privacy: exploring the relationship between data sovereignty and privacy-preserving machine learning, towards constructing a system that can evolve as laws change, and 4) scheduling: to ensure timely access to resources across data centers. To answer these questions we foresee the need for substantial advances in theory, algorithms and system design, as well as engineering of a new practice of Geo-Distributed Machine Learning (GDML).

Acknowledgments

We are grateful to our reviewers and colleagues for their help and comments on earlier versions of this paper. This work was supported by Microsoft, and in part by the Argentine Ministry of Science, Technology and Productive Innovation with the program BEC.AR. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

---

6 Calculated on the full training data as $(f - f^*)/f^*$, where $f^*$ is the minimum value obtained across methods.

7 If the data is randomly distributed, recent work [14] can be leveraged, but if data loss is biased, new techniques must be developed.
References


