Optimization for Large-scale Statistical Modeling

The recent success of machine learning techniques such as deep learning stems from a number of fundamental algorithmic improvements [11, 15]. Since most of the fundamental techniques underlying modern machine learning have their roots in the 80's, what really makes the difference, nowadays, is the availability of unprecedented amounts of training data which, coupled with the commoditization of general purpose GPUs, makes it possible to build models capable of achieving accuracies that in some cases surpass human ability [20].

While a single computer equipped with powerful GPUs provides algorithmic simplicity and speed up to a given scale of training data and model size [17, 10, 5], today we have reached an operating point where both datasets and model sizes no longer fit the capacity of a single machine. Thus, a distributed implementation of training algorithms for modern statistical models, such as deep learning, becomes truly necessary.

In general, the success of deep learning as well as modern Bayesian non parametrics [6], stems from the application of stochastic optimization algorithms [2, 8, 14], that iteratively work on randomized subsets of the training data to determine model parameters that minimize a given loss function. Many learning problems can be expressed as (stochastic) optimization problems given a dataset X = {x1, x2, ... xn}, the goal is to learn the parameters theta of a model with respect to an empirical loss function L(theta, X).

A first-order stochastic optimization algorithm achieves this by iteratively updating using a stochastic gradient of L(theta,X) computed at a randomly sampled xi. This iterative procedure, called Stochastic Gradient Descent (SGD), produces a sequence of models, that is guaranteed to converge to a local optimum of the loss function [18]. Stochastic optimization can be seen as a unifying methodology to attack general learning problems at an unprecedented scale. It is therefore important to define novel algorithms and system implementations to distribute optimization algorithms across a cluster of machines that cooperate to minimize a given loss function. Despite several efforts in this direction [4, 16, 1], distributed optimization is still in its infancy, and more research is required to improve performance, scalability, and generality of optimization algorithms.

Apache Spark represents a natural candidate for large scale statistical modeling, due to its natural predisposition to process distributed data structures (which can hold model parameters as well as gradient updates). Nevertheless, Apache Spark is built on the Bulk Synchronous Parallel processing model [21], which currently restricts its application to synchronous optimization algorithms. The synchronization phase is time consuming also because of stragglers, so that asynchronous solutions have been proposed, where nodes may work on stale data and read-write race conditions may arise. Empirically, on single-node systems, these asynchronous algorithms have yielded order-of-magnitude improvements in performance (see e.g. [23]). While a significant performance improvement can be obtained, convergence is not guaranteed, and a faster computational throughput does not necessarily guarantee a faster convergence [25]. In particular, faster nodes can bias the convergence.

REFERENCES
Mission confiée

GOALS

The goal of this PhD is to address the problem of stragglers in two different ways.

The first deceptively simple approach is to complement regular workers by “backup” workers, which contribute to the computation of gradient updates [3]. More formally, given a set of $n$ workers and $b$ backup workers, the parameter server coordinate a total of $n + b$ workers, but waits for any $n$ gradient updates for computing the next gradient aggregate. Although in practice this approach has shown to partially mitigate the problems of slow workers, it suffers from three main problems. First, depending on the value of $b$, a lot of redundant work is eventually lost. Second, depending on low-level infrastructure details, backup workers might be scheduled concurrently to regular workers on the same machine, which artificially creates resource contention, thus producing new “stragglers”. Third, determining an appropriate number of backup workers for a given setup (which includes also the particular statistical model being optimized) remains elusive. In this research project we aim to address such issues, by working on efficient speculative execution mechanisms that avoid wasting work, while making sure to mitigate the “straggler” problem rather than possibly exacerbating it.

The second approach is to introduce asynchronous primitives in Spark. [24] shows a possible implementation of this approach, but convergence is not guaranteed. [25] proposes another form of fine-grained synchronization, where each executor only depends on a few other executors, in the sense that requires only their state updates to be able to proceed in the computation. This partial dependency reduces communication overhead and may mitigate the effect of stragglers, but may also increase the number of iterations required for convergence. There are a number of open research directions. First, as we mentioned, convergence to the optimum is not guaranteed for many of the proposed solutions. Second, performance models quantifying convergence speed improvement from asynchronous approaches are missing. Finally, when partial dependencies among executors can be introduced as in [25], it is not clear what is the optimal dependency graph.

Principales activités

Activity Research.

Compétences

We are looking for candidates who are self-motivated and would like to conduct high quality research, and publish in top venues. Candidates should have a Master's Degree (or equivalent) in Electrical Engineering, Computer Science, or a closely related area, preferably with a focus on networking or...
communications. They are also expected to have very good analytical skills (Probability Theory, Algorithms, Optimization) and some background in the area of Machine Learning. Previous exposure to Spark is a significant plus. A good level of written and spoken English is mandatory (knowledge of French is not required). Finally, the selected candidate will be well organized and able to integrate and work well in groups. The position duration is normally 3 years, with a maximum duration of 4 years.

Avantages sociaux

- Restauration subventionnée
- Transports publics remboursés partiellement
- Sécurité sociale
- Congés payés
- Aménagement du temps de travail
- Installations sportives

Rémunération

Durée: 36 mois
Localisation: Sophia Antipolis, France
Rémunération: 1982€ brut mensuel (année 1 & 2) et 2085€ brut mensuel (année 3)