WWW 2012 Invited Tutorial Overview
New Templates for Scalable Data Analysis

Amr Ahmed
Yahoo! Research
4301 Great America Pky
Santa Clara, CA 94043, USA
amr.m.ahmed@gmail.com

Alexander J. Smola
Yahoo! Research
4301 Great America Pky
Santa Clara, CA 94043, USA
alex@smola.org

Markus Weimer
Yahoo! Research
4301 Great America Pky
Santa Clara, CA 94043, USA
weimer@acm.org

ABSTRACT

Scalable data analysis has come a long way since the introduction of the MapReduce paradigm a decade ago. In this tutorial we present algorithms for synchronous and asynchronous data processing. They are are capable of dealing with the amounts of data typically available on the internet.

We given a brief description of the problems one faces when performing scalable machine learning on the internet. To motivate matters we provide a number of scenarios from spam filtering, advertising and collaborative filtering. This is followed by an extensive discussion of current and novel synchronous data processing techniques. In particular we emphasize how insights from systems research and databases can be used to achieve significant improvements both in terms of expressiveness and in terms of efficiency of the deployed algorithms.

This is followed by a description of asynchronous data analysis and inference methods. The latter are particularly necessary whenever the estimation problem requires the use of a significant number of latent variables. This includes cases such as clustering, topic models, or graph factorization. We provide an ample number of motivating examples and applications, ranging from user profiling to the analysis of communication networks. Special emphasis is placed on approximations needed to scale algorithms to hundreds of millions of users and billions of documents.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Monte Carlo methods*; I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Supervised Learning, MapReduce, Bulk Synchronous Processing, Graphical models, Asynchronous Inference, Distributed Optimization

Keywords

Sampling, latent variables, topic models, clustering, systems, databases, scalable data analysis

WWW 2012, April 16-20, 2012, Lyon, France.
ACM 978-1-4503-0637-9/11/03.

1. LARGE SCALE MACHINE LEARNING

Machine Learning is rapidly becoming a key enabling tool for intelligent services on the internet. This ranges from adaptive search algorithms to computational advertising, to the analysis of large amounts of news content, recommendations, collaborative filtering, and user modeling at the scale of large social networks. The opportunities are tremendous since targeted high quality content significantly improves the quality of a service, such as mail, news, advertising, search, or the management of user generated content.

A large and important family of problems arising in this context are those of supervised learning. That is, classification, categorization, tagging, the extraction of named entities, supply forecasting, or the detection of unusual events. A commonality of these problems is that they result in convex optimization problems which are amenable to iterative methods. We present online and batch templates for solving these problems.

2. SYNCHRONIZED TEMPLATES

Systems such as MapReduce led to a proliferation of big data sets across the industry. Hadoop and applications built on top of it like Pig, Hive and Cascading are at the core of the derivation of value from those large data sets. At the same time, Machine Learning is increasingly becoming a key technology in unlocking the value of the recorded data. This makes big data platforms such as Hadoop a desirable target environment for machine learning, as many uses of the data (statistical analysis, explorative analysis, feature extraction and ultimately machine learning) can be collocated in the same infrastructure.

Level 1 — Hadoop, Pig, Hive. Executing machine learning algorithms on top of e.g. Hadoop is enabled through their very structure: In many cases, machine learning algorithms can be cast using *statistical queries* (means, loss function values, etc.) as their core computational primitive. Those queries themselves decompose per example in the training data set. This makes MapReduce an attractive programming model for these statistical queries. One simple example are gradient descent algorithms: The gradient of the loss function is the sum of the gradients of the loss function over all data points. This, in turn, lends itself to use the `map` step of MapReduce to compute the per-datapoint gradient and the `reduce` step to sum those up.

Level 2 — Algorithm specific solutions. The MapReduce programming model does not account for the second defining attribute of nearly all machine learning algorithms: They are iterative, while e.g. Hadoop is optimized for jobs

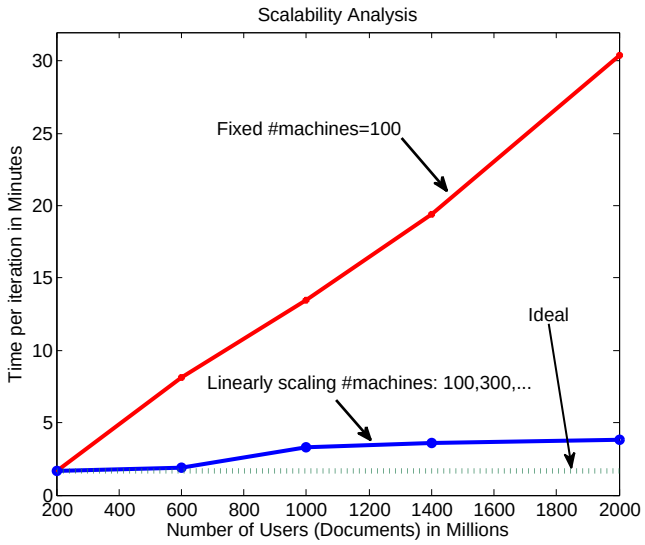

Figure 1: Scaling behavior for distributed latent variable inference. We increase the amount of data linearly with the number of processors (blue curve). The red curve shows that our algorithm scales linearly in the amount of data when the computational resources are fixed.

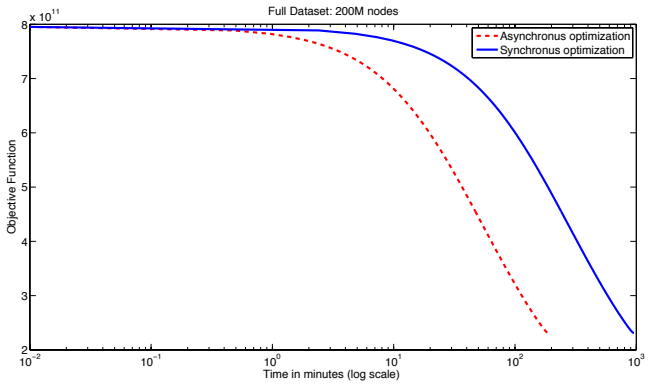

Figure 2: Speed of convergence for synchronous and asynchronous inference when factorizing a graph of 200 million vertices. The asynchronous algorithm is approximately one order of magnitude faster than its synchronous counterpart.

that consist of a single MapReduce job. This mismatch in optimization goals has significant ramifications in terms of the efficiency of machine learning on top of Hadoop and forced successful machine learning systems like Vowpal Wabbit to forfeit the productivity benefits of a distributed runtime environment in favor of a more low-level implementation in order to be effective.

Level 3 — Systems level support in Spark and Pregel. More recently, this iteration weakness of MapReduce has been adressed in a more systematic way by Graphlab, Pregel and Spark. The latter allows the programmer to explicitly optimize for iterative algorithms through caching. The former two add the notion of graph computations as a first class citizen to distributed computing. However, they do not support the complete large scale machine learning pipeline including feature extraction and data cleansing. Additionally, they do not isolate the programmer completely from runtime considerations. This is especially troubling on shared clusters / clouds where the physical environment changes all the time.

Level 4 — Hyracks, Scalops Most recently, this problem has been adressed by declarative approaches like Scalops which include iteration and graph computation as a first class citizen at the declarative layer. This enables the kind of distributed computing "magic" for this class of algorithms that made MapReduce a sucess in its respective application domain.

3. ASYNCHRONOUS TEMPLATES

Unfortunately many inference problems are addressed much more efficiently by asynchronous distributed updates. A common attribute of such problems is that they exhibit multimodality, e.g. in latent variable models where there are several locally optimal assignments and partitions. In this case algorithms which have significant delay between synchronization of the computers involved exhibit considerably worse speed of convergence. It is this very aspect that asynchronous inference templates address (see Figures 1 and 2 for examples of the quality and speed of convergence).

Distributed Variable Storage and Synchronization A key issue in large scale latent variable models is the fact that the number of variables considerably exceeds the amount of memory available on individual machines. We describe a mechanism based on consistent hashing that ensures that we obtain even load balancing over many machines without expensive distribution and planning. The algorithm works by creating centralized copies of random variables and synchronizing them with machine-local instances in a protocol very much akin to message passing.

Method of Multipliers Whenever latency of synchronization becomes a significant contributor to model quality we resort to Bertsekas' method of multipliers. That is, we relax the original estimation problem by adding consensus copies as *separate* variables with Lagrange multipliers to the estimation problem. These variables are updated asynchronously in parallel to the original state variables.

4. APPLICATIONS

We provide three examples of how the above algorithms can be used in a large scale setting: Firstly, we describe time-dependent topic models for user profiling at enterprise level. Secondly, we describe a large scale clustering algorithm, and finally, we give details on factorizing a real-world messaging and communication graph.

Target Audience

The primary audience are researchers in industry and academia who are interested in learning about scalable data analysis frameworks. We assume that the audience have some prior knowledge of statistics and probability theory (we assume familiarity with Bayes rule). Some basic notions of large scale computation such as MapReduce are useful but not required. In some cases we assume that the readers be familiar with belief propagation and message passing in graphical models (a brief introduction will be provided to keep the tutorial self-contained).