Experimenting with Distributed Machine Learning Algorithms: Part 1

CPSC 490, Fall 2013
Patrick Chase
Advisors: Sekhar Tatikonda and Daniel Abadi

I. Abstract

This semester I researched and experimented with distributed systems and large-scale machine learning algorithms. In particular, I investigated Hadoop and Spark to learn about the two different architectures and the properties of each. Hadoop is the well-known, open-source implementation of MapReduce, while Spark is a new technology designed specifically to run multi-pass applications quickly on large amounts of data. Many of the common machine learning algorithms are iterative by nature, making Spark the perfect platform for large-scale machine learning. After gaining an understanding of the two systems, I investigated the current state of distributed machine learning algorithms. I read papers describing distributed algorithms for linear regression, singular value decomposition, and nonnegative matrix factorization. In addition, I examined the current state of the open-source implementations of algorithms on Hadoop and Spark. On Hadoop, there is an open-source library called Mahout that contains many data mining and machine learning methods, but several of the present algorithms have restrictions and shortcomings. On Spark, there is MLlib, which is a library of machine learning methods that is currently limited to only a few algorithms. After researching the different distributed machine learning methods, I decided to further investigate algorithms for Principle Component Analysis (PCA). In addition, I decided it would be interesting to experiment with Spark, as it is a new technology gaining a lot of traction. Next semester, I will design and analyze an algorithm for distributed PCA, then implement this algorithm on Spark. The following report describes the details of my findings this semester and my plans for next semester.
II. MapReduce Architecture

MapReduce is a computing paradigm used to manage and process large-scale data. In particular, the system was designed to be fault tolerant for hardware failures, which can be common if a large amount of commodity machines is used for computation. To specify a task on a MapReduce architecture, two functions must be written: the Map function and the Reduce function.

The Map Function

The map function takes in a chunk of data from the file system and outputs a sequence of key-value pairs. The way the key-value pairs are made is determined by the code written by the user for the Map function. One important distinction about the key-value pairs is that the keys do not have to be unique, and in many cases should not be. For example, if the key-value pairs need to end up at the same machine, all the pairs should have the same key.

The Reduce Function

The Reduce function takes in pairs consisting of a key and its list of associated values. It then combines the values, in some specified way, and outputs a key-value pair with the original key, but the newly constructed value from the list. All of these key-value pairs are then combined into a single file.

Execution

1. Data blocks enter Map nodes, which output a sequence of key-value pairs based on the data and the map function.
2. Key-value pairs are collected by the Master and grouped by key. The keys with all their values are then sent to the Reduce nodes.
3. The Reduce nodes take in pairs consisting of a key and its associated values and combine them. The way the data is combined is determined by the specified Reduce function.
III. Current State of Machine Learning on Hadoop

Mahout

The main open-source library for machine learning and data mining on Hadoop is Apache Mahout. Currently, Mahout has algorithms for clustering, classification, and collaborative filtering. Some of these algorithms include SVMs, logistic regression, random forests, k-means, and latent dirichlet allocation. There is an algorithm for PCA via stochastic singular value decomposition that gives an approximation of solution in just three MapReduce steps, however the general implementation of PCA is open.

IV. Spark Architecture

Spark is an open-source cluster-computing framework developed in the AMPLab at UC Berkeley. Unlike Hadoop, it does not have to read and write to disk after every iteration. Instead, Spark has the ability to cache data in memory and query or process it repeatedly. This makes Spark much faster than Hadoop for iterative algorithms, such as logistic regression, that need to process the same data over and over. In addition, Spark doesn't handle fault tolerance by replication, like Hadoop. Instead, Spark uses the idea of “lineage” to restore the data after a failure. To do this,
Spark stores all of the steps the data has taken since it was last stored on disk, then if the node fails, the system can go back and re-compute the lost data. An important note is that while Spark is much faster than Hadoop for iterative algorithms, there is little speed advantage to using Spark for single-pass queries or data transformations.

V. Current State of Machine Learning on Spark

MLlib

MLlib is the library of machine learning algorithms built on top of Spark. MLlib is currently very limited with only algorithms for linear regression, k-means, logistic regression, and linear SVM. However, contributors are looking to add more to MLlib in the next year, including PCA, random forests, time-series algorithms, and graphical methods.

VI. Plans for Next Semester

After examining the state of algorithms of Hadoop and Spark, I decided it would be best to focus on PCA. PCA remains an open, yet fundamental, algorithm that is used in a variety of settings to study large data sets. In addition, the matrix decomposition methods used for PCA, including SVD, have become a popular topic in the area of distributed computing.

For next semester, I have chosen to implement PCA on Spark. First, I think the added flexibility that will come from using Spark and having control of the memory could make for interesting algorithmic improvements. Second, Spark has gained a lot of traction in the last year, as Cloudera announced it would support Spark, and the Spark creators have established a new startup called Databricks.

Before I begin coding, I will investigate algorithms for distributed PCA, and design and analyze an algorithm that can be implemented on Spark. Then I will begin the implementation.

Acknowledgements:

Special thanks to Alp Kucukelbir, Michael Kane, Abhijit Pujare, and Henry Davidge for their contributions to the Team Big Data (TBD) group.

Citations