Journal of Machine Learning Research 17 (2016) 1-7 Submitted 5/15; Published 4/16

MLlib: Machine Learning in Apache Spark

Xiangrui Meng† MENG@DATABRICKS.COM
Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Joseph Bradley JOSEPH@DATABRICKS.COM
Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Burak Yavuz BURAK@DATABRICKS.COM
Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Evan Sparks SPARKS@CS.BERKELEY.EDU
UC Berkeley, 465 Soda Hall, Berkeley, CA 94720

Shivaram Venkataraman SHIVARAM@EECS.BERKELEY.EDU
UC Berkeley, 465 Soda Hall, Berkeley, CA 94720

Davies Liu DAVIES@DATABRICKS.COM
Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Jeremy Freeman FREEMANJ11@JANELIA.HHMI.ORG
HHMI Janelia Research Campus, 19805 Helix Dr, Ashburn, VA 20147

DB Tsai DBT@NETFLIX.COM
Netflix, 970 University Ave, Los Gatos, CA 95032

Manish Amde MANISH@ORIGAMILOGIC.COM
Origami Logic, 1134 Crane Street, Menlo Park, CA 94025

Sean Owen SOWEN@CLOUDERA.COM
Cloudera UK, 33 Creechurch Lane, London EC3A 5EB United Kingdom

Doris Xin DORX0@ILLINOIS.EDU
UIUC, 201 N Goodwin Ave, Urbana, IL 61801

Reynold Xin RXIN@DATABRICKS.COM
Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Michael J. Franklin FRANKLIN@CS.BERKELEY.EDU
UC Berkeley, 465 Soda Hall, Berkeley, CA 94720

Reza Zadeh REZAB@STANFORD.EDU
Stanford and Databricks, 475 Via Ortega, Stanford, CA 94305

Matei Zaharia MATEI@MIT.EDU
MIT and Databricks, 160 Spear Street, 13th Floor, San Francisco, CA 94105

Ameet Talwalkar† AMEET@CS.UCLA.EDU
UCLA and Databricks, 4732 Boelter Hall, Los Angeles, CA 90095

Editor: Mark Reid
† Corresponding authors.

Abstract

Apache Spark is a popular open-source platform for large-scale data processing that is well-suited for iterative machine learning tasks. In this paper we present MLlib, Spark's open-source distributed machine learning library. MLlib provides efficient functionality for a wide range of learning settings and includes several underlying statistical, optimization, and linear algebra primitives. Shipped with Spark, MLlib supports several languages and provides a high-level API that leverages Spark's rich ecosystem to simplify the development of end-to-end machine learning pipelines. MLlib has experienced a rapid growth due to its vibrant open-source community of over 140 contributors, and includes extensive documentation to support further growth and to let users quickly get up to speed.

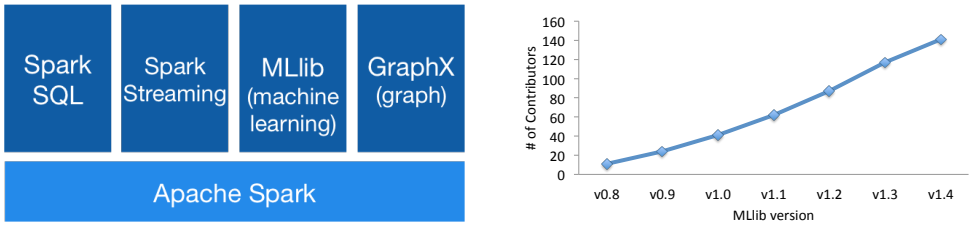
Keywords: scalable machine learning, distributed algorithms, apache spark

1. Introduction

Modern datasets are rapidly growing in size and complexity, and there is a pressing need to develop solutions to harness this wealth of data using statistical methods. Several 'next generation' data flow engines that generalize MapReduce (Dean and Ghemawat, 2004) have been developed for large-scale data processing, and building machine learning functionality on these engines is a problem of great interest. In particular, Apache Spark (Zaharia et al., 2012) has emerged as a widely used open-source engine. Spark is a fault-tolerant and general-purpose cluster computing system providing APIs in Java, Scala, Python, and R, along with an optimized engine that supports general execution graphs. Moreover, Spark is efficient at iterative computations and is thus well-suited for the development of large-scale machine learning applications.

In this work we present MLlib, Spark's distributed machine learning library, and the largest such library. The library targets large-scale learning settings that benefit from data-parallelism or model-parallelism to store and operate on data or models. MLlib consists of fast and scalable implementations of standard learning algorithms for common learning settings including classification, regression, collaborative filtering, clustering, and dimensionality reduction. It also provides a variety of underlying statistics, linear algebra, and optimization primitives. Written in Scala and using native (`C++` based) linear algebra libraries on each node, MLlib includes Java, Scala, and Python APIs, and is released as part of the Spark project under the Apache 2.0 license.

MLlib's tight integration with Spark results in several benefits. First, since Spark is designed with iterative computation in mind, it enables the development of efficient implementations of large-scale machine learning algorithms since they are typically iterative in nature. Improvements in low-level components of Spark often translate into performance gains in MLlib, without any direct changes to the library itself. Second, Spark's vibrant open-source community has led to rapid growth and adoption of MLlib, including contributions from over 140 people. Third, MLlib is one of several high-level libraries built on top of Spark, as shown in Figure 1(a). As part of Spark's rich ecosystem, and in part due to MLlib's `spark.ml` API for pipeline development, MLlib provides developers with a wide range of tools to simplify the development of machine learning pipelines in practice.



(a) (b)

Figure 1: (a) Apache Spark ecosytem. (b). Growth in MLlib contributors.

2. History and Growth

Spark was started in the UC Berkeley AMPLab and open-sourced in 2010. Spark is designed for efficient iterative computation and starting with early releases has been packaged with example machine learning algorithms. However, it lacked a suite of robust and scalable learning algorithms until the creation of MLlib. Development of MLlib began in 2012 as part of the MLbase project (Kraska et al., 2013), and MLlib was open-sourced in September 2013. From its inception, MLlib has been packaged with Spark, with the initial release of MLlib included in the Spark 0.8 release. As an Apache project, Spark (and consequently MLlib) is open-sourced under the Apache 2.0 license. Moreover, as of Spark version 1.0, Spark and MLlib are on a 3-month release cycle.

The original version of MLlib was developed at UC Berkeley by 11 contributors, and provided a limited set of standard machine learning methods. Since this original release, MLlib has experienced dramatic growth in terms of contributors. Less than two years later, as of the Spark 1.4 release, MLlib has over 140 contributors from over 50 organizations. Figure 1(b) demonstrates the growth in MLlib's open source community as a function of release version. The strength of this open-source community has spurred the development of a wide range of additional functionality.

3. Core Features

In this section we highlight the core features of MLlib; we refer the reader to the MLlib user guide for additional details (MLlib, 2015).

Supported Methods and Utilities. MLlib provides fast, distributed implementations of common learning algorithms, including (but not limited to): various linear models, naive Bayes, and ensembles of decision trees for classification and regression problems; alternating least squares with explicit and implicit feedback for collaborative filtering; and k-means clustering and principal component analysis for clustering and dimensionality reduction. The library also provides a number of low-level primitives and basic utilities for convex optimization, distributed linear algebra, statistical analysis, and feature extraction, and supports various I/O formats, including native support for LIBSVM format, data integration via Spark SQL (Armbrust et al., 2015), as well as PMML (Guazzelli et al., 2009) and MLlib's internal format for model export.

Algorithmic Optimizations. MLlib includes many optimizations to support efficient distributed learning and prediction. We highlight a few cases here. The ALS algorithm for

recommendation makes careful use of blocking to reduce JVM garbage collection overhead and to leverage higher-level linear algebra operations. Decision trees use many ideas from the PLANET project (Panda et al., 2009), such as data-dependent feature discretization to reduce communication costs, and tree ensembles parallelize learning both within trees and across trees. Generalized linear models are learned via optimization algorithms which parallelize gradient computation, using fast `C++`-based linear algebra libraries for worker computations. Many algorithms benefit from efficient communication primitives; in particular tree-structured aggregation prevents the driver from being a bottleneck, and Spark broadcast quickly distributes large models to workers.

Pipeline API. Practical machine learning pipelines often involve a sequence of data preprocessing, feature extraction, model fitting, and validation stages. Most machine learning libraries do not provide native support for the diverse set of functionality required for pipeline construction. Especially when dealing with large-scale datasets, the process of cobbling together an end-to-end pipeline is both labor-intensive and expensive in terms of network overhead. Leveraging Spark's rich ecosystem and inspired by previous work (Pedregosa et al., 2011; Buitinck et al., 2013; Sparks et al., 2013, 2015), MLlib includes a package aimed to address these concerns. This package, called `spark.ml`, simplifies the development and tuning of multi-stage learning pipelines by providing a uniform set of high-level APIs (Meng et al., 2015), including APIs that enable users to swap out a standard learning approach in place of their own specialized algorithms.

Spark Integration. MLlib benefits from the various components within the Spark ecosystem. At the lowest level, Spark core provides a general execution engine with over 80 operators for transforming data, e.g., for data cleaning and featurization. MLlib also leverages the other high-level libraries packaged with Spark. Spark SQL provides data integration functionality, SQL and structured data processing which can simplify data cleaning and preprocessing, and also supports the DataFrame abstraction which is fundamental to the `spark.ml` package. GraphX (Gonzalez et al., 2014) supports large-scale graph processing and provides a powerful API for implementing learning algorithms that can naturally be viewed as large, sparse graph problems, e.g., LDA (Blei et al., 2003; Bradley, 2015). Additionally, Spark Streaming (Zaharia et al., 2013) allows users to process live data streams and thus enables the development of online learning algorithms, as in Freeman (2015). Moreover, performance improvements in Spark core and these high-level libraries lead to corresponding improvements in MLlib.

Documentation, Community, and Dependencies. The MLlib user guide provides extensive documentation; it describes all supported methods and utilities and includes several code examples along with API docs for all supported languages (MLlib, 2015). The user guide also lists MLlib's code dependencies, which as of version 1.4 are the following open-source libraries: Breeze, netlib-java, and (in Python) NumPy (Breeze, 2015; Halliday, 2015; Braun, 2015; NumPy, 2015). Moreover, as part of the Spark ecosystem, MLlib has active community mailing lists, frequent meetup events, and JIRA issue tracking to facilitate open-source contributions (Community, 2015). To further encourage community contributions, Spark Packages (Packages, 2015) provides a community package index to track the growing number of open source packages and libraries that work with Spark. To date, several of the contributed packages consist of machine learning functionality that builds on MLlib.

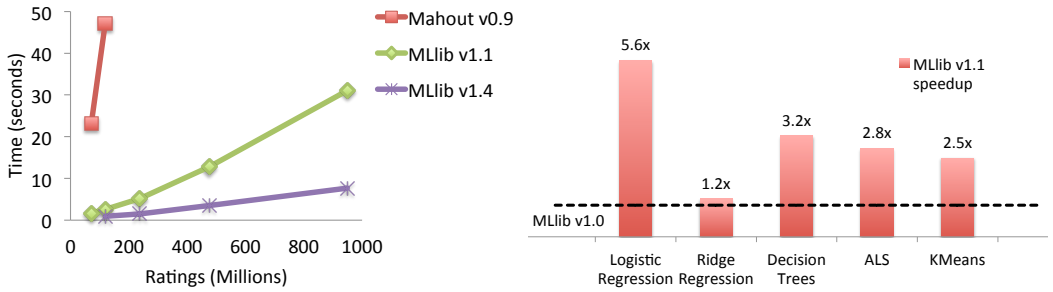


Figure 2: (a) Benchmarking results for ALS. (b) MLlib speedup between versions.

Finally, a massive open online course has been created to describe the core algorithmic concepts used to develop the distributed implementations within MLlib (Talwalkar, 2015).

4. Performance and Scalability

In this section we briefly demonstrate the speed, scalability, and continued improvements in MLlib over time. We first look at scalability by considering ALS, a commonly used collaborative filtering approach. For this benchmark, we worked with scaled copies of the Amazon Reviews dataset (McAuley and Leskovec, 2013), where we duplicated user information as necessary to increase the size of the data. We ran 5 iterations of MLlib's ALS for various scaled copies of the dataset, running on a 16 node EC2 cluster with m3.2xlarge instances using MLlib versions 1.1 and 1.4. For comparison purposes, we ran the same experiment using Apache Mahout version 0.9 (Mahout, 2014), which runs on Hadoop MapReduce. Benchmarking results, presented in Figure 2(a), demonstrate that MapReduce's scheduling overhead and lack of support for iterative computation substantially slow down its performance on moderately sized datasets. In contrast, MLlib exhibits excellent performance and scalability, and in fact can scale to much larger problems.

Next, we compare MLlib versions 1.0 and 1.1 to evaluate improvement over time. We measure the performance of common machine learning methods in MLlib, with all experiments performed on EC2 using m3.2xlarge instances with 16 worker nodes and synthetic datasets from the spark-perf package (`https://github.com/databricks/spark-perf`). The results are presented in Figure 2(b), and show a 3× speedup on average across all algorithms. These results are due to specific algorithmic improvements (as in the case of ALS and decision trees) as well as to general improvements to communication protocols in Spark and MLlib in version 1.1 (Yavuz and Meng, 2014).

5. Conclusion

MLlib is in active development, and the following link provides details on how to contribute: `https://cwiki.apache.org/confluence/display/SPARK/Contributing+to+Spark`. Moreover, we would like to acknowledge all MLlib contributors. The list of Spark contributors can be found at `https://github.com/apache/spark`, and the `git log` command can be used to identify MLlib contributors.

References

Michael Armbrust, Reynold Xin, Cheng Lian, Yin Yuai, Davies Liu, Joseph Bradley, Xiangrui Meng, Tomer Kaftan, Michael Franklin, Ali Ghodsi, and Matei Zaharia. Spark SQL: Relational data processing in spark. In *ACM Special Interest Group on Management of Data*, 2015.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

Joseph Bradley. Topic modeling with LDA: MLlib meets GraphX. `https://databricks.com/?p=3135`, 2015.

Mikio Braun. jblas. `http://jblas.org/`, 2015.

Breeze. Breeze. `https://github.com/scalanlp/breeze/wiki`, 2015.

Lars Buitinck et al. API design for machine learning software: experiences from the scikit-learn project. *arXiv:1309.0238*, 2013.

Spark Community. Spark community. `https://spark.apache.org/community.html`, 2015.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *USENIX Symposium on Operating Systems Design and Implementation*, 2004.

Jeremy Freeman. Introducing streaming k-means in spark 1.2. `https://databricks.com/?p=2382`, 2015.

Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *Conference on Operating Systems Design and Implementation*, 2014.

Alex Guazzelli, Michael Zeller, Wen-Ching Lin, and Graham Williams. PMML: An open standard for sharing models. *The R Journal*, 1(1), 2009.

Sam Halliday. netlib-java. `https://github.com/fommil/netlib-java`, 2015.

Tim Kraska, Ameet Talwalkar, John Duchi, Rean Griffith, Michael Franklin, and Michael Jordan. MLbase: A Distributed Machine-learning System. In *Conference on Innovative Data Systems Research*, 2013.

Mahout. Apache mahout. `http://mahout.apache.org/`, 2014.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *ACM Recommender Systems Conference*, 2013.

Xiangrui Meng, Joseph Bradley, Evan Sparks, and Shivaram Venkataraman. Ml pipelines: A new high-level api for MLlib. `https://databricks.com/?p=2473`, 2015.

MLlib. MLlib user guide. `https://spark.apache.org/docs/latest/mllib-guide.html`, 2015.

NumPy. Numpy. http://www.numpy.org/, 2015.

Spark Packages. Spark packages. https://spark-packages.org, 2015.

Biswanath Panda, Joshua S. Herbach, Sugato Basu, and Roberto J. Bayardo. Planet: Massively parallel learning of tree ensembles with mapreduce. In *International Conference on Very Large Databases*, 2009.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

Evan R. Sparks, Ameet Talwalkar, Virginia Smith, Jey Kottalam, Xinghao Pan, Joseph E. Gonzalez, Michael J. Franklin, Michael I. Jordan, and Tim Kraska. MLI: An API for Distributed Machine Learning. In *International Conference on Data Mining*, 2013.

Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J. Franklin, Michael I. Jordan, and Tim Kraska. Automating model search for large scale machine learning. *Symposium on Cloud Computing*, 2015.

Ameet Talwalkar. BerkeleyX CS190-1x: Scalable machine learning. https://www.edx.org/course/scalable-machine-learning-uc-berkeleyx-cs190-1x, 2015.

Burak Yavuz and Xiangrui Meng. Spark 1.1: MLlib performance improvements. https://databricks.com/?p=1393, 2014.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *USENIX Symposium on Networked Systems Design and Implementation*, 2012.

Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Symposium on Operating Systems Principles*, 2013.