Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Apache Mahout: Machine Learning on Distributed Dataflow Systems

Robin Anil ROBINANIL@APACHE.ORG
Gokhan Capan GCAPAN@APACHE.ORG
Isabel Drost-Fromm ISABEL@APACHE.ORG
Shannon Quinn SQUINN@APACHE.ORG
Paritosh Ranjan PRANJAN@APACHE.ORG
Sebastian Schelter SSC@APACHE.ORG

Editor:

Abstract

Apache Mahout is a library for scalable machine learning (ML) on distributed dataflow systems, offering various implementations of classification, clustering, dimensionality reduction and recommendation algorithms. It originated in 2008 and targeted MapReduce, which was the predominant abstraction for scalable computing in industry at that time. It has since then migrated to a general framework for linear algebraic computations on dataflow backends such as Apache Spark, Apache Flink and H20. Mahout is maintained as a community-driven, top-level, open source project at the Apache Software Foundation.

1. Introduction

Mahout was started in 2008 as a subproject of the open source search engine *Apache Lucence* (Owen et al. (2012); McCandless et al. (2010)), whose community encountered a growing need for applying ML techniques on large text corpora. In 2010, Mahout became a top-level Apache project. A critical component of modern large-scale machine learning (ML) is operating on, accessing, and analyzing datasets stored in distributed filesystems running on a cluster of machines. In such an environment, data analysis is often conducted using distributed dataflow engines, that allow for scalable, data-parallel execution of programs.

2. Legacy: MapReduce-based Algorithms

TODO: Clarify: Not many choices for ML in Hadoop ecosystem, quick introduction to MapReduce, (Chu et al. (2007)) showed that a large family of popular ML algorithms can be reformulated under the MapReduce paradigm

Classification MR implementation of Rennie et al. (2003)

Clustering MR implementation of canopy clustering McCallum et al. (2000)

Collaborative Filtering

SVD Lanczos + Stochastic SVD Halko (2012)

item-based collaborative filtering (Sarwar et al. (2001)) Dunning (1993); Schelter et al. (2012); Dunning and Friedman (2014), ALS Schelter et al. (2013) todo add original paper Zhou et al. (2008)

3. Mahout Samsara

TODO: Intro, rewrite Examples of such systems include Apache Spark (Zaharia et al. (2012)), Apache Flink (Alexandrov et al. (2014)) and H2o (H2o). Unfortunately, these systems are difficult to program, as their programming model is heavily influenced by the underlying data-parallel execution scheme. Usually, programs consist of a sequence of parallelizable second-order functions (such as `map`, `reduce` or `groupBy`) that dictate how the system should execute user-defined first-order functions on partitioned data Zaharia et al. (2012). Such programming models are non-intuitive for users without a background in distributed systems, and are in general hard to program without a detailed understanding of the underlying execution model. Furthermore, the available programming abstractions typically rely on partitioned, unordered sets; this is a mismatch for ML applications that mostly operate on linear algebra constructs (e.g., vectors and matrices). Therefore, implementing ML algorithms on dataflow systems is a tedious and difficult task. Mahout rebuilt itself on top of *Samsara* (Lyubimov and Palumbo (2016)), a domain-specific language for declarative machine learning in cluster environments. Samsara allows its users to specify programs using a set of common matrix abstractions and linear algebraic operations, similar to R or MATLAB. Samsara then compiles, optimizes and executes these programs on distributed dataflow systems (Schelter et al. (2016)). The aim of Samsara is to allow mathematicians and data scientists to leverage the scalability of distributed dataflow systems via common declarative abstractions, while drastically reducing the need for detailed knowledge of the programming model and execution scheme of the underlying systems. Samsara is part of the Apache Mahout library and supports backends like Apache Spark and Apache Flink. TODO: 2 sentences for the optimization techniques

Architecture and Execution. Figure 1 illustrates the architecture of Samsara. Applications are written using the Scala DSL. The in-memory operations are immediately executed, while operations on DRMs are deferred. The system records the actions to perform on these distributed matrices, and internally builds a directed acyclic graph (DAG) of logical operations from them, where vertices refer to matrices and edges correspond to transformations between them. Materialization barriers (e.g., persisting a result or collecting a matrix into local memory) implicitly trigger execution. Upon execution, the DAG of logical operators is optimized and transformed into a DAG of physical operators to execute. These physical operators are specific to one of the backends that Samsara supports (currently Apache Spark and Apache Flink), and run the distributed parts of the program using the respective backend. TODO: explain example in 3-4 sentences

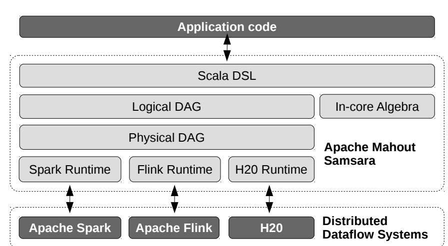

Figure 1: Architecture.

```
def dridge(data: DrmLike[Int], lambda: Double): Matrix {
```

```
  // slice out features, add column for bias term
  val drmX = data(::, 0 until data.ncol) cbind 1
  val drmY = data(::, data.ncol) // slice out target

  val drmXtX = drmX.t %*% drmX //distributed matrix
  val drmXtY = drmX.t %*% drmY // multiplications

  val XtX = drmXtX.collect // materialization of results
  val XtY = drmXty.collect // in driver memory

  XtX.diagv += lambda // add regularization
  solve(XtX, XtY) // compute parameters in-core on driver
}
```

Listing 1: Distributed Ridge Regression for tall & skinny matrices using Samsara.

TODO: Details on backends, e.g., Flink-backend (Alexandrov et al. (2014)) has problems with control-flow

4. Availability and Requirements

Mahout is run as a top-level project under the umbrella of the Apache Software Foundation, and developed in a community-driven, meritocratic fashion according to the *Apache Way*[1]. Mahout is available under an Apache License at `https://mahout.apache.org`. Mahout requires at least Java 7 and Scala 2.10 for Samsara. The legacy algorithms require Hadoop 2.4, while Samsara compiles to to Flink 1.1, Spark 1.6/2.x and H20 0.1.25.

5. Outlook

linear + relational algebra, df systems one extreme, dl systems other, middleground must be found

6. Acknowledgements

TODO: list anyone here who chose not to be a co-author Mahout is the product of the efforts of numerous people. We would like to thank all the project's committers who did not co-author this paper: Abdelhakim Deneche, Anand Avati, Andrew Musselman, Benson Margulies, Dan Filimon, David Hall, Dawid Weiss, Dmitriy Lyubimov, Drew Farris, Ellen Friedman, Erik Hatcher, Frank Scholten, Grant Ingersoll, Jake Mannix, Jeff Eastman, Karl Wettin, Niranjan Balasubramanian, Otis Gospodnetic, Ozgur Yilmazel, Pat Ferrel, Sean Owen, Stevo Slavi, Suneel Marthi, Ted Dunning, Tom Pierce, Trevor Grant. Of course we extend our thanks tos all users and contributors as well.

References

Alexander Alexandrov, Rico Bergmann, Stephan Ewen, Johann-Christoph Freytag, Fabian Hueske, Arvid Heise, Odej Kao, Marcus Leich, Ulf Leser, Volker Markl, et al. The

1. `https://www.apache.org/foundation/how-it-works.html`

stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.

Cheng-Tao Chu, Sang K Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Kunle Olukotun, and Andrew Y Ng. Map-reduce for machine learning on multicore. In *Advances in neural information processing systems*, pages 281–288, 2007.

Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

Ted Dunning and Ellen Friedman. *Practical Machine Learning: Innovations in Recommendation*. O'Reilly Media, 2014.

H2o. H2o prediction engine, http://www.h2o.ai/. URL `http://www.h2o.ai/`.

Nathan P Halko. *Randomized methods for computing low-rank approximations of matrices*. PhD thesis, University of Colorado, 2012.

Dmitriy Lyubimov and Andrew Palumbo. *Apache Mahout: Beyond MapReduce*. CreateSpace Independent Publishing Platform, 2016.

Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in action*. Manning Publications, 2010.

Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. Mahout in action, 2012.

Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 616–623, 2003.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Sebastian Schelter, Christoph Boden, and Volker Markl. Scalable similarity-based neighborhood methods with mapreduce. In *RecSys*, pages 163–170, 2012.

Sebastian Schelter, Christoph Boden, Martin Schenck, Alexander Alexandrov, and Volker Markl. Distributed matrix factorization with mapreduce using a series of broadcast-joins. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 281–284. ACM, 2013.

Sebastian Schelter, Andrew Palumbo, Shannon Quinn, Suneel Marthi, and Andrew Musselman. Samsara: Declarative machine learning on distributed dataflow systems. In *NIPS Workshop MLSystems*, 2016.

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 *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*, pages 337–348. Springer, 2008.