

The Java Data Mining Package – A Data Processing Library for Java

Holger Arndt, Department of Computer Science, Technical University of Munich, 85747 Garching, Germany, Email: mail@holger-arndt.com

I. Introduction

Data analysis is an important task for which many software tools have been developed, e.g. Matlab, Octave, or R. Open source Java software is becoming more and more common, e.g. Weka [4], LibSVM, LibLinear, Apache Mahout, MALLET, KNIME, Java-ML or Rapid-Miner formerly known as YALE.

However, it remains difficult to combine different libraries with each other, handle data which exceeds main memory, or to process a task in parallel on a computer cluster. The **Java Data Mining Package (JDMP)** [2] addresses these problems and provides

- a modular framework for the representation of algorithms and data sources,
- support for very large data sets,
- tools for distributed processing,
- integration of existing data processing libraries,
- visualization methods.

II. Data Storage

Consistent data representation, using matrices from the **Universal Java Matrix Package (UJMP)** [1], forms the basis of all data objects. The present library offers methods to handle

- dense and sparse matrices,
- multi-dimensional matrices,
- arbitrary data types,
- up to 2^{63} rows or columns,
- data exceeding main memory,
- import and export filters.

III. Object Model

In order to represent higher-level objects, JDMP introduces a modular interface hierarchy:

- **Matrix**: data storage,
- **Variable**: collection of matrices,
- **Sample**: collection of variables,
- **DataSet**: collection of samples,
- **Algorithm**: manipulation of other objects,
- **Module**: logical grouping of objects, e.g. per task, per thread, etc.

An example is illustrated in Fig. 1:

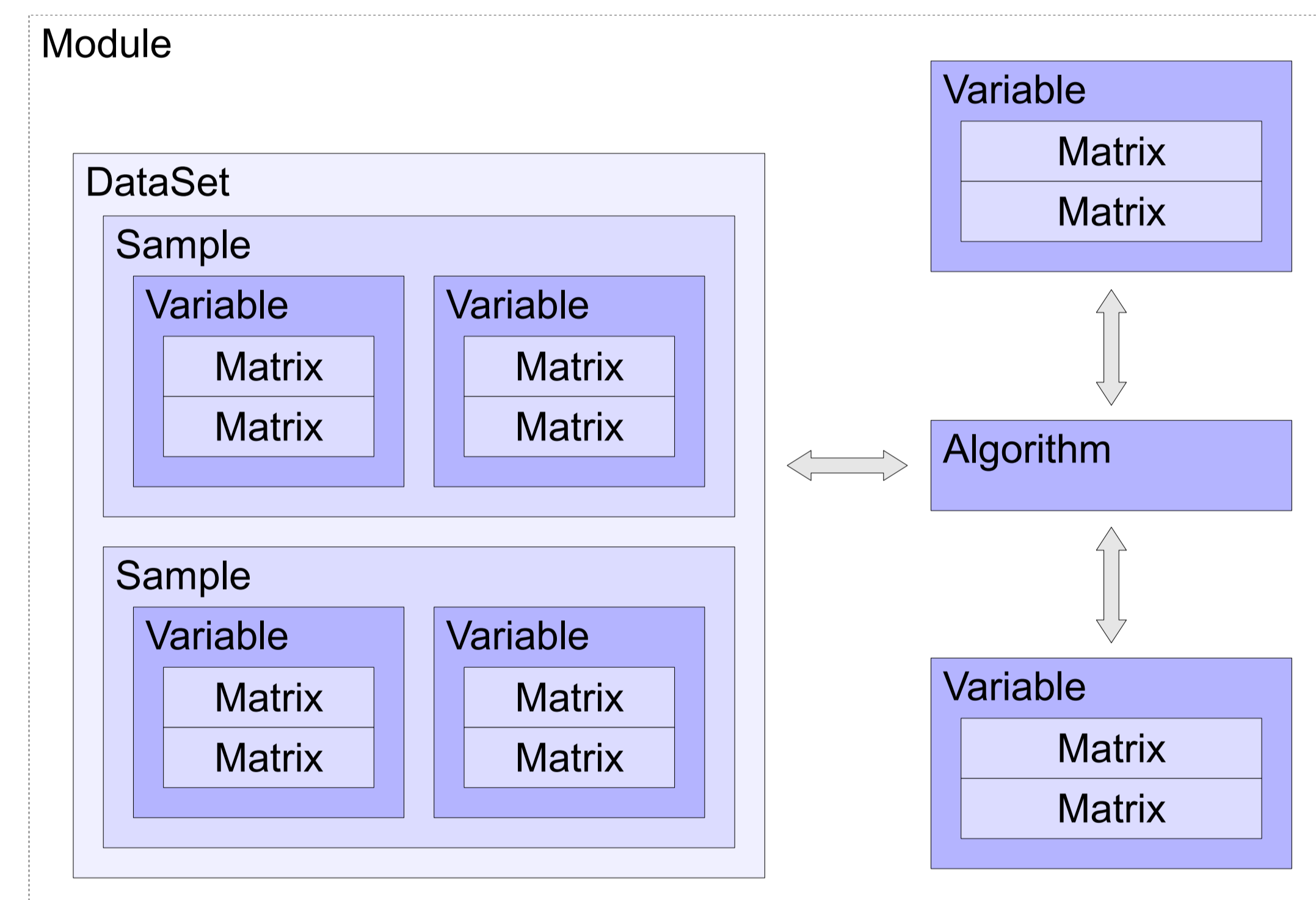


Figure 1: Interface hierarchy in JDMP.

Most objects support different representations, e.g. a dataset can be accessed as a list of individual samples, or as a single matrix containing the combined data. This concept allows the convenient re-use of UJMP's matrix methods.

IV. Distributed Computing

Due to the strict separation of calculation methods and data structures, distributed algorithms can be formulated easily. A local stub object is created for data on a remote machine, and read or write requests are redirected accordingly,

thus making it irrelevant to the programmer whether a variable or dataset is stored on the same computer or on another pc. The technique used for sharing data or invoking methods on a remote machine is exchangeable. Transmission could be realized through RMI (remote method invocation), HTTP requests over a SOAP interface, or through common files on a network drive. Another method is the use of additional frameworks such as JGroups, Terracotta, Apache Hadoop, or Google's Map-Reduce [3].

V. Integrating Other Libraries

JDMP is not intended to rebuild or substitute existing machine learning libraries, but rather to combine them efficiently. A structure of interfaces and abstract stub classes allows easy integration of additional algorithms and storage implementations. As a result, the actual implementation becomes secondary and exchangeable, as long as it can be mapped to the interface definition described in section III.

Fig. 2 illustrates the relation between matrices from the **Universal Java Matrix Package**, objects from the **Java Data Mining Package**, and other frameworks or libraries:

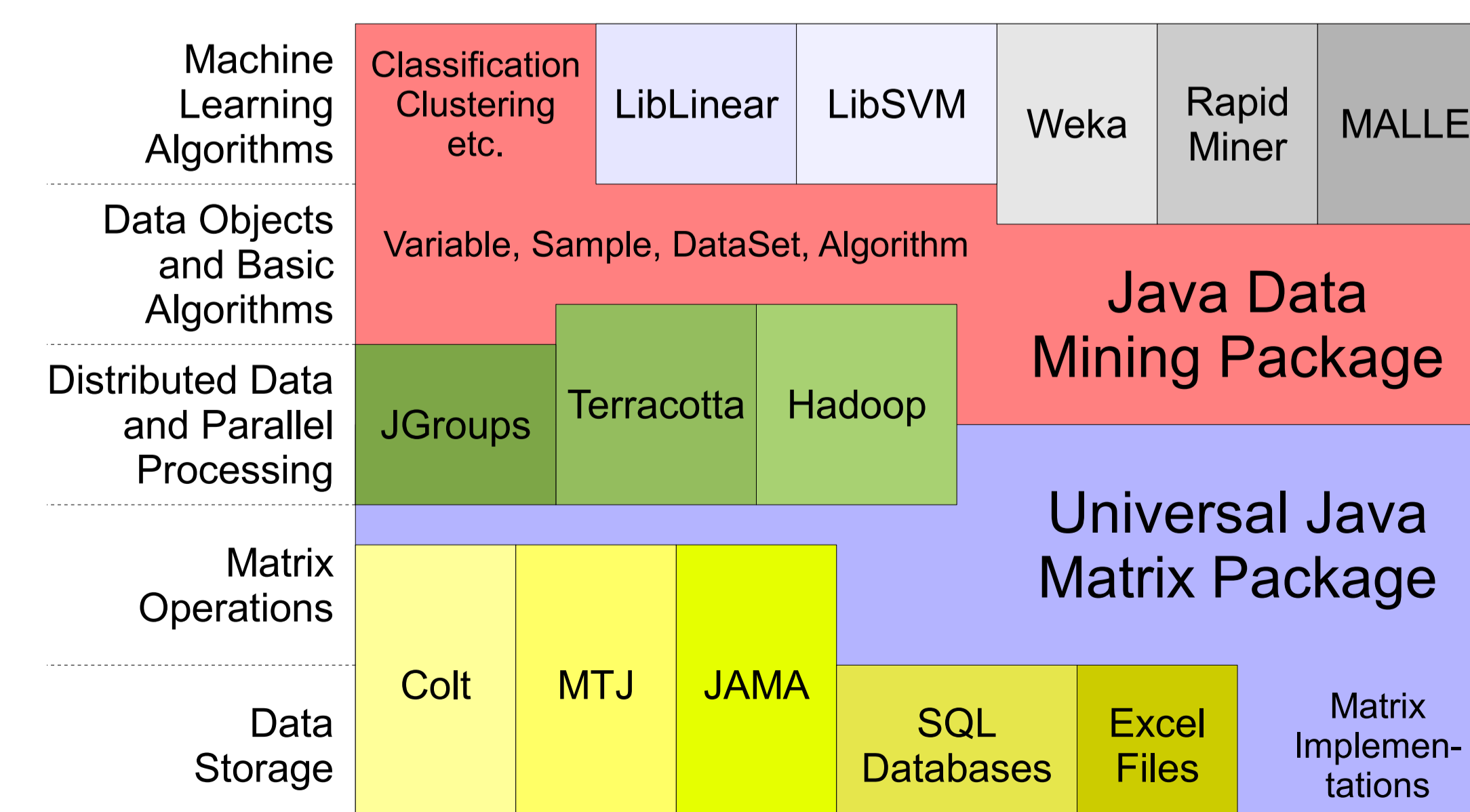


Figure 2: Layer model of JDMP.

VI. Summary

The **Java Data Mining Package** introduces an additional layer of abstraction between machine learning tools such as Weka, data storage implementations, and data processing tools for parallel computing such as Apache Hadoop. This facilitates the deployment of a task to a computer cluster and renders the implementation independent of the technology used for data processing and storage.

However, it is still at an early stage of development and cannot yet compete with other libraries in terms of functionality and stability. In addition, availability of documentation remains to be improved, which will make it easier for the user to become acquainted with the library.

It should be emphasized, that the **Java Data Mining Package** is licensed under LGPL, which allows its integration into commercial applications. Source code and jar files are freely available through our website [2] in the hope that it will attract many users and developers.

References

- [1] Holger Arndt, Markus Bundschuh, and Andreas Nägele, *Towards a next-generation matrix library for Java*, COMPSAC: International Computer Software and Applications Conference (2009), in press.
- [2] Holger Arndt, Andreas Nägele, and Markus Bundschuh, *Java Data Mining Package (JDMP)*, <http://www.jdmp.org>, 2009.
- [3] Cheng T. Chu, Sang K. Kim, Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun, *Map-reduce for machine learning on multicore*, NIPS, MIT Press, 2006, pp. 281–288.
- [4] Ian H. Witten and Eibe Frank, *Data mining: Practical machine learning tools and techniques*, 2nd ed., Morgan Kaufmann, San Francisco, 2005.