Journal of Machine Learning Research 14 (2013) 801-805 Submitted 9/12; Revised 2/13; Published 3/13

MLPACK: A Scalable C++ Machine Learning Library

Ryan R. Curtin RYAN.CURTIN@CC.GATECH.EDU
James R. Cline JAMES.CLINE@GATECH.EDU
N. P. Slagle NPSLAGLE@GMAIL.COM
William B. March MARCH@GATECH.EDU
Parikshit Ram P.RAM@GATECH.EDU
Nishant A. Mehta NICHE@CC.GATECH.EDU
Alexander G. Gray AGRAY@CC.GATECH.EDU
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Editor: Balázs Kégl

Abstract

MLPACK is a state-of-the-art, scalable, multi-platform C++ machine learning library released in late 2011 offering both a simple, consistent API accessible to novice users and high performance and flexibility to expert users by leveraging modern features of C++. MLPACK provides cutting-edge algorithms whose benchmarks exhibit far better performance than other leading machine learning libraries. MLPACK version 1.0.3, licensed under the LGPL, is available at `http://www.mlpack.org`.

Keywords: C++, dual-tree algorithms, machine learning software, open source software, large-scale learning

1. Introduction and Goals

Though several machine learning libraries are freely available online, few, if any, offer efficient algorithms to the average user. For instance, the popular Weka toolkit (Hall et al., 2009) emphasizes ease of use but scales poorly; the distributed Apache Mahout library offers scalability at a cost of higher overhead (such as clusters and powerful servers often unavailable to the average user). Also, few libraries offer breadth; for instance, libsvm (Chang and Lin, 2011) and the Tilburg Memory-Based Learner (TiMBL) are highly scalable and accessible yet each offer only a single method.

MLPACK, intended to be the machine learning analog to the general-purpose LAPACK linear algebra library, aims to combine efficiency and accessibility. Written in C++, MLPACK uses the highly efficient Armadillo matrix library (Sanderson, 2010) and is freely available under the GNU Lesser General Public License (LGPL). Through the use of C++ templates, MLPACK both eliminates unnecessary copying of data sets and performs expression optimizations unavailable in other languages. Also, MLPACK is, to our knowledge, unique among existing libraries in using generic programming features of C++ to allow customization of the available machine learning methods without incurring performance penalties.

In addition, users ranging from students to experts should find the consistent, intuitive interface of MLPACK to be highly accessible. Finally, the source code provides references and comprehensive documentation.

Four major goals of the development team of MLPACK are

- to implement **scalable, fast** machine learning algorithms,
- to design an **intuitive, consistent, and simple** API for non-expert users,
- to implement a **variety** of machine learning methods, and
- to provide **cutting-edge** machine learning algorithms unavailable elsewhere.

This paper offers both an introduction to the simple and extensible API and a glimpse of the superior performance of the library.

2. Package Overview

Each algorithm available in MLPACK features both a set of C++ library functions and a standalone command-line executable. Version 1.0.3 includes the following methods:

- nearest/furthest neighbor search with cover trees or kd-trees (k-nearest-neighbors)
- range search with cover trees or kd-trees
- Gaussian mixture models (GMMs)
- hidden Markov models (HMMs)
- LARS / Lasso regression
- k-means clustering
- fast hierarchical clustering (Euclidean MST calculation)[1] (March et al., 2010)
- kernel PCA (and regular PCA)
- local coordinate coding[1] (Yu et al., 2009)
- sparse coding using dictionary learning
- RADICAL (Robust, Accurate, Direct ICA aLgorithm) (Learned-Miller and Fisher, 2003)
- maximum variance unfolding (MVU) via LRSDP[1] (Burer and Monteiro, 2003)
- the naive Bayes classifier
- density estimation trees[1] (Ram and Gray, 2011)

The development team manages MLPACK with Subversion and the Trac bug reporting system, allowing easy downloads and simple bug reporting. The entire development process is transparent, so any interested user can easily contribute to the library. MLPACK can compile from source on Linux, Mac OS, and Windows; currently, different Linux distributions are reviewing MLPACK for inclusion in their package managers, which will allow users to install MLPACK without needing to compile from source.

3. A Consistent, Simple API

MLPACK features a highly accessible API, both in style (such as consistent naming schemes and coding conventions) and ease of use (such as templated defaults), as well as stringent documentation standards. Consequently, a new user can execute algorithms out-of-the-box often with little or no adjustment to parameters, while the seasoned expert can expect extreme flexibility in algorithmic

1. This algorithm is not available in any other comparable software package.

Data Set	MLPACK	Weka	Shogun	MATLAB	mlpy	sklearn
wine	**0.0003**	0.0621	0.0277	0.0021	0.0025	0.0008
cloud	**0.0069**	0.1174	0.5000	0.0210	0.3520	0.0192
wine-qual	**0.0290**	0.8868	4.3617	0.6465	4.0431	0.1668
isolet	**13.0197**	213.4735	37.6190	46.9518	52.0437	46.8016
miniboone	**20.2045**	216.1469	2351.4637	1088.1127	3219.2696	714.2385
yp-msd	**5430.0478**	>9000.0000	>9000.0000	>9000.0000	>9000.0000	>9000.0000
corel	**4.9716**	14.4264	555.9600	60.8496	209.5056	160.4597
covtype	**14.3449**	45.9912	>9000.0000	>9000.0000	>9000.0000	651.6259
mnist	**2719.8087**	>9000.0000	3536.4477	4838.6747	5192.3586	5363.9650
randu	**1020.9142**	2665.0921	>9000.0000	1679.2893	>9000.0000	8780.0176

Table 1: *k*-NN benchmarks (in seconds).

Data Set	**wine**	**cloud**	**wine-qual**	**isolet**	**miniboone**
UCI Name	Wine	Cloud	Wine Quality	ISOLET	MiniBooNE
Size	178x13	2048x10	6497x11	7797x617	130064x50

Data Set	**yp-msd**	**corel**	**covtype**	**mnist**	**randu**
UCI Name	YearPredictionMSD	Corel	Covertype	*N/A*	*N/A*
Size	515345x90	37749x32	581082x54	70000x784	1000000x10

Table 2: Benchmark data set sizes.

tuning. For example, the following line initializes an object which will perform the standard k-means clustering in Euclidean space:

```
KMeans<> k();
```

However, an expert user could easily use the Manhattan distance, a different cluster initialization policy, and allow empty clusters:

```
KMeans<ManhattanDistance, KMeansPlusPlusInitialization, AllowEmptyClusters> k();
```

Users can implement these custom classes in their code, then simply link against the MLPACK library, requiring no modification within the MLPACK library. In addition to this flexibility, Armadillo 3.4.0 includes sparse matrix support; sparse matrices can be used in place of dense matrices for the appropriate MLPACK methods.

4. Benchmarks

To demonstrate the efficiency of the algorithms implemented in MLPACK, we present a comparison of the running times of *k*-nearest-neighbors and the *k*-means clustering algorithm from MLPACK, Weka (Hall et al., 2009), MATLAB, the Shogun Toolkit (Sonnenburg et al., 2010), mlpy (Albanese et al., 2012), and scikit.learn ('sklearn') (Pedregosa et al., 2011), using a modest consumer-grade workstation containing an AMD Phenom II X6 1100T processor clocked at 3.3 GHz and 8 GB of RAM.

Eight data sets from the UCI data sets repository (Frank and Asuncion, 2010) are used; the MNIST handwritten digit database is also used ('mnist') (LeCun et al., 2001), as well as a uniformly distributed random data set ('randu'). Information on the sizes of these ten data sets appears in Table 2. Data set loading time is not included in the benchmarks. Each test was run 5 times; the average is shown in the results.

Data Set	Clusters	MLPACK	Shogun	MATLAB	sklearn
wine	3	**0.0006**	0.0073	0.0055	0.0064
cloud	5	**0.0036**	0.1240	0.0194	0.1753
wine-qual	7	**0.0221**	0.6030	0.0987	4.0407
isolet	26	**4.9762**	8.5093	54.7463	7.0902
miniboone	2	**0.1853**	8.0206	0.7221	*memory*
yp-msd	10	**34.8223**	135.8853	269.7302	*memory*
corel	10	**0.4672**	2.4237	1.6318	*memory*
covtype	7	**13.5997**	71.1283	54.9034	*memory*
mnist	10	**80.2092**	163.7513	133.9970	*memory*
randu	75	**727.1498**	7443.2675	3117.5177	*memory*

Table 3: k-means benchmarks (in seconds).

k-NN was run with each library on each data set, with $k = 3$. The results for each library and each data set appears in Table 1. The k-means algorithm was run with the same starting centroids for each library, and 1000 iterations maximum. The number of clusters k was chosen to reflect the structure of the data set. Benchmarks for k-means are given in Table 3. Weka and mlpy are excluded because they do not allow specification of the starting centroids. '*memory*' indicates that the system ran out of memory during the test.

MLPACK's k-nearest neighbors and k-means are faster than the competitors in all test cases. Benchmarks for other methods, omitted due to space constraints, also show similar speedups over competing implementations.

5. Future Plans and Conclusion

The favorable benchmarks exhibited above are not necessarily the global optimum; MLPACK's active development team includes several core developers and many contributors. Because MLPACK is open-source, contributions from outsiders are welcome, including feature requests and bug reports. Thus, the performance, extensibility, and breadth of algorithms within MLPACK are all certain to improve.

The first releases of MLPACK lacked parallelism, but experimental parallel code using OpenMP is currently in testing. This parallel support must maintain a simple API and avoid large, reverse-incompatible API changes. Other useful planned features include using on-disk databases (rather than requiring loading the data set entirely into RAM) and validation of saved models (such as trees or distributions). Refactoring work continues on existing code, providing more flexible abstractions and greater extensibility. Nevertheless, MLPACK's future growth will mostly be the addition of new machine learning methods; since the original release (1.0.0), there are five new methods. Forthcoming methods include approximate nearest neighbors, locality-sensitive hashing (LSH), and support vector machines (SVMs).

In conclusion, we have shown that MLPACK is a state-of-the-art C++ machine learning library which leverages the powerful C++ concept of generic programming to give excellent performance on large data sets.

Acknowledgments

A full list of developers and researchers (other than the authors) who have contributed significantly to MLPACK are Sterling Peet, Vlad Grantcharov, Ajinkya Kale, Dongryeol Lee, Chip Mappus, Hua Ouyang, Long Quoc Tran, Noah Kauffman, Rajendran Mohan, and Trironk Kiatkungwanglai.

References

Davide Albanese, Roberto Visintainer, Stefano Merler, Samantha Riccadonna, Giuseppe Jurman, and Cesare Furlanello. mlpy: Machine Learning PYThon. 2012. Project homepage at `http://mlpy.fbk.eu/`.

Samuel Burer and Renato D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Andrew J. Frank and Arthur Asuncion. UCI machine learning repository [`http://archive.ics.uci.edu/ml`], 2010. University of California, Irvine, School of Information and Computer Sciences.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

Erik G. Learned-Miller and John W. Fisher. ICA using spacings estimates of entropy. *Journal of Machine Learning Research*, 4:1271–1295, December 2003. ISSN 1532-4435.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

William B. March, Parikshit Ram, and Alexander G. Gray. Fast Euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 603–612, 2010.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Parikshit Ram and Alexander G. Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 627–635, New York, NY, USA, 2011. ACM.

Conrad Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, 2010.

Soeren Sonnenburg, Gunnar Raetsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, June 2010.

Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 2223–2231, 2009.