Journal of Machine Learning Research 10 (2009) 1755-1758 Submitted 10/08; Revised 4/09; Published 7/09

Dlib-ml: A Machine Learning Toolkit

Davis E. King DAVISKING@USERS.SOURCEFORGE.NET
Northrop Grumman ES, ATR and Image Exploitation Group
Baltimore, Maryland, USA

Editor: Soeren Sonnenburg

Abstract

There are many excellent toolkits which provide support for developing machine learning software in Python, R, Matlab, and similar environments. Dlib-ml is an open source library, targeted at both engineers and research scientists, which aims to provide a similarly rich environment for developing machine learning software in the C++ language. Towards this end, dlib-ml contains an extensible linear algebra toolkit with built in BLAS support. It also houses implementations of algorithms for performing inference in Bayesian networks and kernel-based methods for classification, regression, clustering, anomaly detection, and feature ranking. To enable easy use of these tools, the entire library has been developed with contract programming, which provides complete and precise documentation as well as powerful debugging tools.

Keywords: kernel-methods, svm, rvm, kernel clustering, C++, Bayesian networks

1. Introduction

Dlib-ml is a cross platform open source software library written in the C++ programming language. Its design is heavily influenced by ideas from design by contract and component-based software engineering. This means it is first and foremost a collection of independent software components, each accompanied by extensive documentation and thorough debugging modes. Moreover, the library is intended to be useful in both research and real world commercial projects and has been carefully designed to make it easy to integrate into a user's C++ application.

There are a number of well known machine learning libraries. However, many of these libraries focus on providing a good environment for doing research using languages other than C++. Two examples of this kind of project are the Shogun (Sonnenburg et al., 2006) and Torch (Collobert and Bengio, 2001) toolkits which, while they are implemented in C++, are not focused on providing support for developing machine learning software in that language. Instead they are primarily intended to be used with languages like R, Python, Matlab, or Lua. Then there are toolkits such as Shark (Igel et al., 2008) and dlib-ml which are explicitly targeted at users who wish to develop software in C++. Given these considerations, dlib-ml attempts to help fill some of the gaps in tool support not already filled by libraries such as Shark. It is hoped that these efforts will prove useful for researchers and engineers who wish to develop machine learning software in this language.

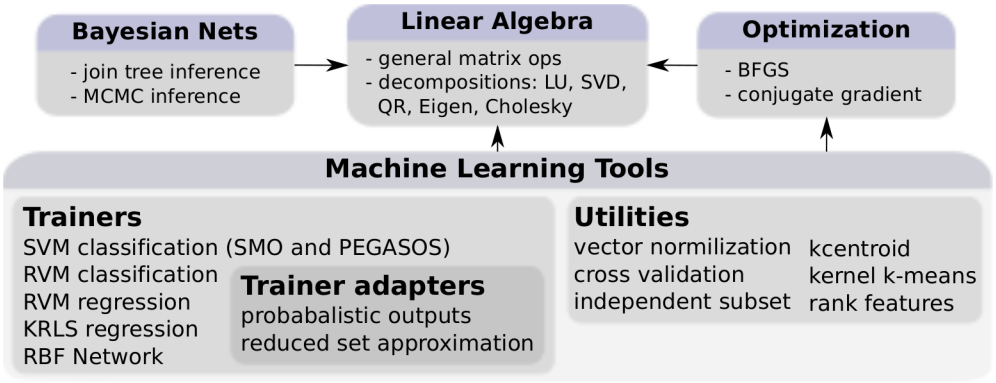

Figure 1: Elements of dlib-ml. Arrows show dependencies between components.

2. Elements of the Library

The library is composed of the four distinct components shown in Figure 1. The linear algebra component provides a set of core functionality while the other three implement various useful tools. This paper addresses the two main components, linear algebra and machine learning tools.

2.1 Linear Algebra

The design of the linear algebra component of the library is based on the template expression techniques popularized by Veldhuizen and Ponnambalam (1996) in the Blitz++ numerical software. This technique allows an author to write simple Matlab-like expressions that, when compiled, execute with speed comparable to hand-optimized C code. The dlib-ml implementation extends this original design in a number of ways. Most notably, the library can use the BLAS when available, meaning that the performance of code developed using dlib-ml can gain the speed of highly optimized libraries such as ATLAS or the Intel MKL while still using a very simple syntax. Consider the following example involving matrix multiplies, transposes, and scalar multiplications:

```
(1) result = 3*trans(A*B + trans(A)*2*B);
(2) result = 3*trans(B)*trans(A) + 6*trans(B)*A;
```

The result of expression (1) could be computed using only two calls to the matrix multiply routine in BLAS but first it is necessary to reorder the terms into form (2) to fit the form expected by the BLAS routines. Performing these transformations by hand is tedious and error prone. Dlib-ml automatically performs these transformations on all expressions and invokes the appropriate BLAS calls. This enables the user to write equations in the form most intuitive to them and leave these details of software optimization to the library. This is a feature not found in the supporting tools of other C++ machine learning libraries.

2.2 Machine Learning Tools

A major design goal of this portion of the library is to provide a highly modular and simple architecture for dealing with kernel algorithms. In particular, each algorithm is parameterized to allow a user to supply either one of the predefined dlib-ml kernels, or a new user defined kernel. Moreover, the implementations of the algorithms are totally separated from the data on which they operate.

This makes the dlib-ml implementation generic enough to operate on any kind of data, be it column vectors, images, or some other form of structured data. All that is necessary is an appropriate kernel.

This is a feature unique to dlib-ml. Many libraries allow arbitrary precomputed kernels and some even allow user defined kernels but have interfaces which restrict them to operating on column vectors. However, none allow the flexibility to operate *directly* on arbitrary objects, making it much easier to apply custom kernels in the case where the kernels operate on objects other than fixed length vectors.

The library provides implementations of popular algorithms such as RBF networks and support vector machines for classification. It also includes algorithms not present in other major ML toolkits such as relevance vector machines for classification and regression (Tipping and Faul, 2003). All of these algorithms are implemented as generic trainer objects with a standard interface. This design allows trainer objects to be used by a number of generic meta-algorithms that do tasks such as performing cross validation, reducing the number of output support vectors (Suttorp and Igel, 2007), or fitting a sigmoid to the output decision function to make decisions interpretable in probabilistic terms (Platt, 1999). This generic trainer interface, along with the contract programming approach, makes the library easily extensible by other developers.

Another good example of a generic kernel algorithm provided by the library is the kernel RLS technique introduced by Engel et al. (2004). It is a kernelized version of the famous recursive least squares filter, and functions as an excellent online regression method. With it, Engel introduced a simple but very effective technique for producing sparse outputs from kernel learning algorithms.

Engel's sparsification technique is also used by one of dlib-ml's most versatile tools, the kcentroid object. It is a general utility for representing a weighted sum of sample points in a kernel induced feature space. It can be used to easily kernelize any algorithm that requires only the ability to perform vector addition, subtraction, scalar multiplication, and inner products.

The kcentroid object enables the library to provide a number of useful kernel-based machine learning algorithms. The most straightforward of which is online anomaly detection, which simply marks data samples as novel if their distance from the centroid of a previously observed body of data is large (e.g., 3 standard deviations from the mean distance). A similarly simple but still powerful application is in feature ranking, where features are considered good if their inclusion results in a large distance between the centroids of different classes of data.

Another straightforward application of this technique is in kernelized cluster analysis. Using the kcentroid it is easy to create sparse kernel clustering algorithms. To demonstrate this, the library comes with a sparse kernel k-means algorithm.

Finally, dlib-ml contains two SVM solvers. One is essentially a reimplementation of LIBSVM (Chang and Lin, 2001) but with the generic parameterized kernel approach used in the rest of the library. This solver has roughly the same CPU and memory utilization characteristics as LIBSVM. The other SVM solver is a kernelized version of the Pegasos algorithm introduced by Shalev-Shwartz et al. (2007). It is built using the kcentroid and thus produces sparse outputs.

3. Availability and Requirements

The library is released under the Boost Software License, allowing it to be incorporated into both open-source and commercial software. It requires no additional libraries, does not need to be configured or installed, and is frequently tested on MS Windows, Linux and MacOS X but should work with any ISO C++ compliant compiler.

Note that dlib-ml is a subset of a larger project named dlib hosted at http://dclib.sourceforge.net. Dlib is a general purpose software development library containing a graphical application for creating Bayesian networks as well as tools for handling threads, network I/O, and numerous other tasks. Dlib-ml is available from the dlib project's download page on SourceForge.

References

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Ronan Collobert and Samy Bengio. Svmtorch: support vector machines for large-scale regression problems. *J. Mach. Learn. Res.*, 1:143–160, 2001. ISSN 1533-7928.

Yaakov Engel, Shie Mannor, and Ron Meir. Kernel recursive least squares. *IEEE Transactions on Signal Processing*, 52:2275–2285, 2004.

Christian Igel, Tobias Glasmachers, and Verena Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML '07*, pages 807–814, New York, NY, USA, 2007. ACM.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *J. Mach. Learn. Res.*, 7:1531–1565, 2006. ISSN 1533-7928.

Thorsten Suttorp and Christian Igel. Resilient approximation of kernel classifiers. volume 4668 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2007.

Michael E. Tipping and Anita C. Faul. Fast marginal likelihood maximisation for sparse Bayesian models. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 3–6, 2003.

Todd Veldhuizen and Kumaraswamy Ponnambalam. Linear algebra with C++ template metaprograms. *Dr. Dobb's Journal of Software Tools*, 21(8):38–44, 1996.