Journal of Machine Learning Research 11 (2010) 1799-1802 Submitted 10/09; Revised 3/10; Published 6/10

The SHOGUN Machine Learning Toolbox

Sören Sonnenburg[*] SOEREN.SONNENBURG@TU-BERLIN.DE
Berlin Institute of Technology
Franklinstr. 28/29, 10587 Berlin, Germany

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Sebastian Henschel SHOGUN@KODEAFFE.DE
Christian Widmer CWIDMER@TUEBINGEN.MPG.DE
Jonas Behr JONAS.BEHR@TUEBINGEN.MPG.DE
Alexander Zien[†] ALEXANDER.ZIEN@TUEBINGEN.MPG.DE
Fabio de Bona FABIO.DE.BONA@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory, Max Planck Society
Spemannstr. 39, 72076 Tübingen, Germany

Alexander Binder ALEXANDER.BINDER@TU-BERLIN.DE
Christian Gehl[‡] CHRISTIAN.GEHL@TRIFENSE.DE
Berlin Institute of Technology
Franklinstr. 28/29, 10587 Berlin, Germany

Vojtěch Franc XFRANCV@CMP.FELK.CVUT.CZ
Center for Machine Perception, Czech Technical University
Technicka 2, 166 27 Praha 6, Czech Republic

Editor: Cheng Soon Ong

Abstract

We have developed a machine learning toolbox, called SHOGUN, which is designed for unified large-scale learning for a broad range of feature types and learning settings. It offers a considerable number of machine learning models such as support vector machines, hidden Markov models, multiple kernel learning, linear discriminant analysis, and more. Most of the specific algorithms are able to deal with several different data classes. We have used this toolbox in several applications from computational biology, some of them coming with no less than 50 million training examples and others with 7 billion test examples. With more than a thousand installations worldwide, SHOGUN is already widely adopted in the machine learning community and beyond. SHOGUN is implemented in C++ and interfaces to *MATLAB*,™ *R*, *Octave*, *Python*, and has a stand-alone command line interface. The source code is freely available under the GNU General Public License, Version 3 at `http://www.shogun-toolbox.org`.

Keywords: support vector machines, kernels, large-scale learning, Python, Octave, R

1. Introduction

With the great advancements of machine learning in the past few years, many new learning algorithms have been proposed, implemented in C/C++, and made publicly available. Their availability

∗. Also at Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany.
†. Current Address LIFE Biosystems GmbH, Poststr. 34, 69115 Heidelberg, Germany.
‡. Also at Trifense GmbH, Germendorfer Str. 79, 16727 Velten.

has enabled systematic comparisons between newly developed methods, leading to an increased visibility, and supporting their broad adoption in the community of machine learning as well as in many others (see discussion in Sonnenburg *et al.*, 2007). However, for example, there currently exist more than 20 different publicly available implementations of Support Vector Machine (SVM) solvers. Each one comes with its own interface, a small set of available kernel functions, and unique benefits and drawbacks. There is no single unified way of interfacing with these implementations, even though they all are based on essentially the same methodology of supervised learning. This restraints users from fully taking advantage of the recent developments in machine learning algorithms.

This motivated us to develop a machine learning toolbox that provides an easy, unified way for solving certain types of machine learning problems. The result is a toolbox, called SHOGUN, with a focus on large-scale learning using kernel methods and SVMs. It provides a generic interface to 15 SVM implementations, among them SVMlight, LibSVM, GPDT, SVMLin, LibLinear, and OCAS. The SVMs can be easily combined with more than 35 different kernel functions.[1] The toolbox not only provides efficient implementations of the most common kernels, like the linear, polynomial, Gaussian, and sigmoid, but also comes with several recently developed kernels such as the locality improved, Fisher, TOP, spectrum, and the WD kernels for sequence analysis (see Sonnenburg et al., 2007; Ben-Hur et al., 2008; Schweikert *et al*., 2009, and references therein). Moreover, it offers options for using precomputed kernels and allows easy integration of new implementations of kernels. One of SHOGUN's key features is the *combined kernel* to construct weighted linear combinations of multiple kernels that may even be defined on different input domains. Also, several Multiple Kernel Learning algorithms based on different regularization strategies are available to optimize the weighting of the kernels (e.g., Sonnenburg et al., 2006a; Kloft et al., 2010).

Currently, two- and multiclass classification and regression are best supported. In addition to kernel and distance based methods, SHOGUN implements many linear methods and features algorithms to train hidden Markov models. Furthermore, it provides the basic functionality for solving label sequence learning problems (Hidden Semi-Markov SVMs; cf. Rätsch and Sonnenburg, 2007; Schweikert *et al*., 2009). The input feature objects can be dense or sparse vectors of strings, integers (8, 16, 32 or 64 bit; signed or unsigned), or floating point numbers (32 or 64 bit), and can be converted into different feature types. Chains of "pre-processors" (e.g., subtracting the mean) can be attached to each feature object allowing on-the-fly pre-processing. Finally, several commonly used performance measures, like accuracy and area under ROC or precision-recall curves, are implemented in SHOGUN.

An important aspect in the design of SHOGUN was to enable very large-scale learning. It is structured in a way that there is as little as possible overhead for storing the data and intermediate results. Moreover, whenever possible, we implemented auxiliary routines that allow faster computation of combinations of kernel elements that lead to significant speedup during training (for some SVM implementations, e.g., SVMlight and GPDT) and evaluation (Sonnenburg et al., 2007). Furthermore, linear SVMs can be efficiently trained using on the fly computed feature spaces, even mixing sparse, dense and other data types.

This allowed us to use SHOGUN for solving several large-scale learning problems in biological sequence analysis, for example, splice site recognition with up to 50 million example sequences for training (Sonnenburg et al., 2007; Franc and Sonnenburg, 2009) and transcription start site recog-

1. Complete lists of SVM and kernel implementations together with user and developer documentation is available at `http://www.shogun-toolbox.org/doc`.

nition with almost 7 billion test sequences (Sonnenburg et al., 2006b). SHOGUN's core functions are encapsulated in a library (`libshogun`) and are easily accessible and extendible by C++ application developers. What sets SHOGUN apart from many other machine learning toolboxes, is that it provides interactive user interfaces to most major scripting languages that are currently used in scientific computing, in particular *Python*, *MATLAB*, *Octave*, *R*, and a command-line version.

All classes and functions are documented and come with over 600 examples and a tutorial for new users and developers is part of the release.[2] Furthermore, there is an open access tutorial on SVMs (Ben-Hur et al., 2008) that provides a SHOGUN-based command-line tool for illustrative examples (available at `http://svmcompbio.tuebingen.mpg.de`). Maintaining high code quality is ensured by a test suite that supports running the algorithms for each interface on predefined inputs in order to detect breakage. SHOGUN runs on POSIX platforms, such as *Linux*, *BSD*, *Mac OS X*, *Cygwin*, and *Solaris*.

2. Modular, Extendible Object-Oriented Design

SHOGUN is implemented in an object-oriented way using C++ as the programming language. All objects inherit from `CSGObject`, which provides means for garbage collection via reference counting, serialization, and versioning of the object. The implementations of many classes employ *templates* enabling SHOGUN's support of many different data types without code duplication. As the source code and user documentation is automatically generated using `doxygen` and written in-place in the header files, it also drastically reduces the amount of work needed to maintain the documentation. As an example of SHOGUNs object-oriented design, consider the class `CClassifier`: From this class, `CKernelMachine`, for example, is derived and provides basic functions for applying a trained kernel classifier (computing $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \, \mathrm{k}(\mathbf{x}, \mathbf{x}_i) + b$) thus enabling code re-use whenever possible. The same holds for `CDistanceMachine` and `CLinearClassifier` (which provides $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ etc.). Currently, SHOGUN implements 329 classes (see Figure 1 in supplementary material available at `http://www.shogun-toolbox.org/jmlr10` for a sketch of the main classes).[3]

3. Interfaces to Scripting Languages and Applications

Built around SHOGUN's core are two types of interfaces: A modular interface that makes use of the SWIG (`http://www.swig.org`), and a static interface. Thanks to SWIG, the modular interface provides the exact same objects in a modular object-oriented way that are available from C++ to other languages, such as *R*, *Python*, and *Octave*. Using so-called `typemaps`, it is convenient to provide type mappings from the native datatype used in the interface to SHOGUN. For example, a function `void set_features(double* features, int` *n*`, int` *m*`)` can be called directly from *Octave* with a single matrix argument, for example, `set_features(randn(3,4))`. The variables *n* and *m* are then automatically set to the matrix dimensions and together with a data pointer passed to the SHOGUN core.

SHOGUN also provides a static interface with the same structure for all supported plattforms , including the command-line interface where inputs are provided as either strings or files. It is

2. Examples are also accessible online at `http://www.shogun-toolbox.org/doc/examples.html`.
3. See `http://www.shogun-toolbox.org/doc/annotated.html` for the full annotated class listing.

implemented independent of the target language through the class `CSGInterface`, which provides abstract functions to deliver or obtain data from any particular plattform.

A community around SHOGUN is continuously developing, with a growing number of projects building on it (cf. `http://mloss.org/software/tags/shogun`) and a mailing list with more than 100 subscribed users.[4] By 10/2009, there had been at least $1,100$ installations under the *Linux* distributions *Debian* and *Ubuntu*. We will continue to develop SHOGUN and are confident that it is and will continue to be useful, and will make an increasing impact beyond the machine learning community by benefiting diverse applications.

Acknowledgments

We acknowledge partial support by the EU under the PASCAL2 Network of Excellence (ICT-216886) as well as DFG Grants MU 987/6-1 and RA-1894/1-1. VF was supported by the EU Reintegration grant SEMISOL (PERG04-GA-2008-239455).

References

A Ben-Hur, CS Ong, S Sonnenburg, B Schölkopf, and G Rätsch. Support vector machines and kernels for computational biology. *PLoS Computat Biol*, 4(10):e1000173, 2008.

V Franc and S Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2192, 2009.

M Kloft, U Brefeld, S Sonnenburg, P Laskov, K-R Müller, and A Zien. Efficient and accurate l_p-norm multiple kernel learning. In *Proc. NIPS 21*. MIT, Cambridge, MA, 2010.

G Rätsch and S Sonnenburg. Large-scale hidden semi-Markov SVMs. In *Proc. NIPS 19*, pages 1161–1168. MIT Press, Cambridge, MA, 2007.

G Schweikert *et al.* mGene: accurate SVM-based gene finding with an application to nematode genomes. *Genome Research*, 19(11):2133–43, Nov 2009.

S Sonnenburg, G Rätsch, C Schäfer, and B Schölkopf. Large-scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006a.

S Sonnenburg, A Zien, and G Rätsch. ARTS: Accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–480, 2006b.

S Sonnenburg, G Rätsch, and K Rieck. Large-scale learning with string kernels. In L Bottou et al., editor, *Large-Scale Kernel Machines*, pages 73–103. MIT Press, 2007.

S Sonnenburg *et al.* The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 2007.

4. See `http://www.shogun-toolbox.org` for instructions on how to subscribe and `http://news.gmane.org/gmane.comp.ai.machine-learning.shogun` for an archive