Journal of Machine Learning Research 13 (2012) 1967-1971 Submitted 2/12; Revised 5/12; Published 6/12

Jstacs: A Java Framework for Statistical Analysis and Classification of Biological Sequences

Jan Grau[*] JAN.GRAU@INFORMATIK.UNI-HALLE.DE
Institute of Computer Science
Martin Luther University Halle–Wittenberg
D-06099 Halle (Saale), Germany

Jens Keilwagen[*] JENS.KEILWAGEN@IPK-GATERSLEBEN.DE
Leibniz Institute of Plant Genetics and Crop Plant Research (IPK)
Corrensstraße 3
D-06466 Stadt Seeland, OT Gatersleben, Germany

André Gohr ANDRE.GOHR@INFORMATIK.UNI-HALLE.DE
Institute of Computer Science
Martin Luther University Halle–Wittenberg
D-06099 Halle (Saale), Germany

Berit Haldemann BERIT.HALDEMANN@INFORMATIK.HU-BERLIN.DE
Department of Computer Science
Humboldt University of Berlin
Unter den Linden 6, D-10099 Berlin, Germany

Stefan Posch STEFAN.POSCH@INFORMATIK.UNI-HALLE.DE
Ivo Grosse IVO.GROSSE@INFORMATIK.UNI-HALLE.DE
Institute of Computer Science
Martin Luther University Halle–Wittenberg
D-06099 Halle (Saale), Germany

Editor: Antti Honkela

Abstract

Jstacs is an object-oriented Java library for analysing and classifying sequence data, which emerged from the need for a standardized implementation of statistical models, learning principles, classifiers, and performance measures. In Jstacs, these components can be used, combined, and extended easily, which allows for a direct comparison of different approaches and fosters the development of new components. Jstacs is especially tailored to biological sequence data, but is also applicable to general discrete and continuous data. Jstacs is freely available at `http://www.jstacs.de` under the GNU GPL license including an API documentation, a cookbook, and code examples.

Keywords: machine learning, statistical models, Java, bioinformatics, classification

1. Introduction

During the last years, machine learning techniques have gained an increasing importance in many fields of science including bioinformatics and computational biology. A plethora of new or improved statistical models, learning principles, and classification approaches has evolved.

∗. Both authors contributed equally.

One critical step in assessing their relevance is the comparison to existing methods. Such direct comparisons are hampered, if the approaches compared are implemented in stand-alone applications or web-servers or if different performance measures are being used. In addition, the same building blocks, such as statistical models or evaluation of performance measures, are implemented repeatedly, which induces an unnecessary implementation overhead slowing down scientific progress.

These observations lead to the development of Jstacs as an object-oriented open-source library. In contrast to other libraries like JavaML (Abeel et al., 2009) or Shogun (Sonnenburg et al., 2010), Jstacs focuses on statistical models and statistical learning principles. Similar to JavaML, Jstacs is mainly targeted at developers who want to use the library in their own code.

In a typical Jstacs application for sequence classification, a user first chooses appropriate statistical models for the data of the different classes. One then combines these models to a classifier, chooses a learning principle for learning the parameters of this classifier, and learns this classifier on training data. Finally, one uses the classifier for predicting class labels for previously unseen data. For assessing the performance of the classifier, one can choose an evaluation schema like cross-validation and choose different performance measures.

At each of these steps, one may use a statistical model, classifier, learning principle, evaluation schema, or performance measure existing in Jstacs, or implement and use new ones. At each level, Jstacs defines interfaces and abstract classes to standardize and ease development, and to achieve modularity. Thus, a replacement of one component does not require a modification of other parts.

Jstacs has been applied to diverse biological problems such as prediction of transcription factor binding sites and splice sites, de-novo motif discovery, analysis of gene expression and Array-CGH data, and classification based on flow cytometry data.

In the following section, we describe the general structure, essential interfaces, and abstract classes of Jstacs. In a case study, we show how these can be used for building a problem-specific application.

2. The Jstacs Library

In Jstacs, data representation is organized at three levels: alphabets, sequences, and data sets. The most prevalent alphabet in Jstacs is the `DNAAlphabet`, while more general implementations can be used for instance to define a three-letter amino acid alphabet. `Sequence`s are defined using such alphabets, while `DataSet`s comprise a collection of sequences over the same alphabet. `DataSet`s are constructed either from an existing array of sequences or from a file. The latter is the standard way of loading data into Jstacs. In addition, `DataSet`s can be sampled from statistical models.

On the algorithmic side, Jstacs is organized around two central types depicted in Figure 1: the abstract class `AbstractClassifier` and the interface `StatisticalModel`, including two subinterfaces `TrainableStatisticalModel` and `DifferentiableStatisticalModel` abbreviated as `TrainSM` and `DiffSM`, respectively. For these interfaces, Jstacs provides abstract classes with standard implementations of many of the specified methods to reduce implementation effort as well as factory classes enabling user-friendly creation of many standard models.

`TrainableStatisticalModel`s provide methods for training the parameters of the model from one data set. For example, this can be accomplished generatively by analytic parameter estimation. Current implementations include inhomogeneous and homogeneous Markov models, Bayesian networks, hidden Markov models, and mixture models accepting any `TrainableStatisticalModel` as mixture components.

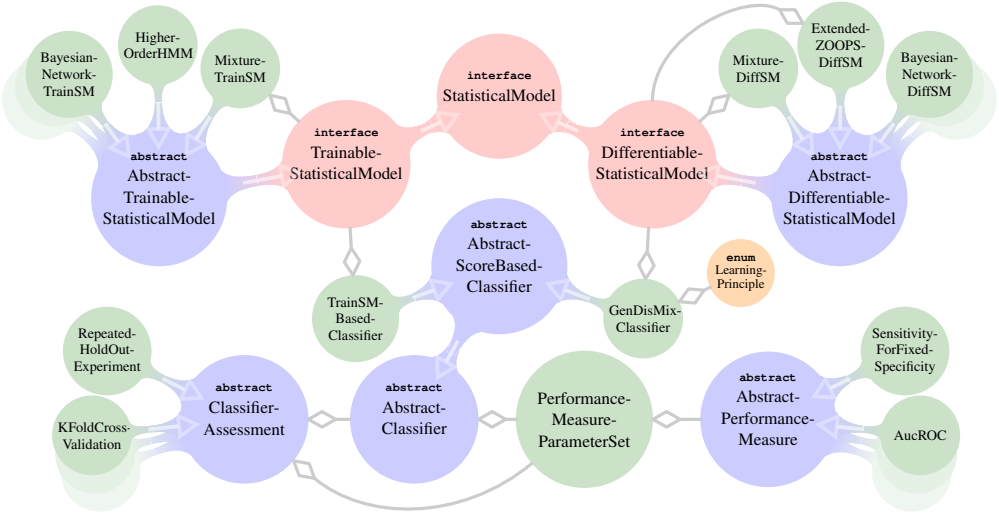

Figure 1: Part of the class structure of Jstacs. The `interfaces` are colored red, `abstract` classes blue, `enums` orange, and concrete classes green without preceeding modifier. Continuous transitions represent inheritance, where arrows indicate the direction of inheritance. Arrows with diamond heads represent usage of a type in the class at the arrow head.

In contrast, `DifferentiableStatisticalModels` provide methods tailored to numerical optimization like the computation of gradients with respect to their parameters. Jstacs provides several `DifferentiableStatisticalModels` including Markov models, Bayesian networks, and mixtures of `DifferentiableStatisticalModels`. In addition, a ZOOPS[1] model for de-novo motif discovery is implemented in `ExtendedZOOPSDiffSM`, which will be the topic of the case study.

`AbstractClassifiers` provide methods for learning internal `StatisticalModels` on training data from different classes and for classifying new input sequences. The `TrainSMBasedClassifier` trains each of the provided `TrainableStatisticalModels` separately on the data set for the corresponding class. The `GenDisMixClassifier` performs a simultaneous numerical parameter estimation for the enclosed `DifferentiableStatisticalModel`, for instance by maximum supervised posterior (MSP) (Grünwald et al., 2002; Cerquides and de Mántaras, 2005), or a unified learning principle (GenDisMix) (Keilwagen et al., 2010).

`ClassifierAssessments` can be used for assessing the performance of any `AbstractClassifier`, for example by k-fold cross validation or repeated holdout sampling. Here, the user may choose one or multiple performance measures such as sensitivity, precision, or the areas under the receiver operating characteristic and precision-recall curve.

3. Case Study

In this section, we describe how we used Jstacs for developing Dispom, a new application for de-novo motif discovery (Keilwagen et al., 2011). Existing approaches for de-novo motif discovery

1. ZOOPS abbreviates *Zero or one occurrence per sequence*.

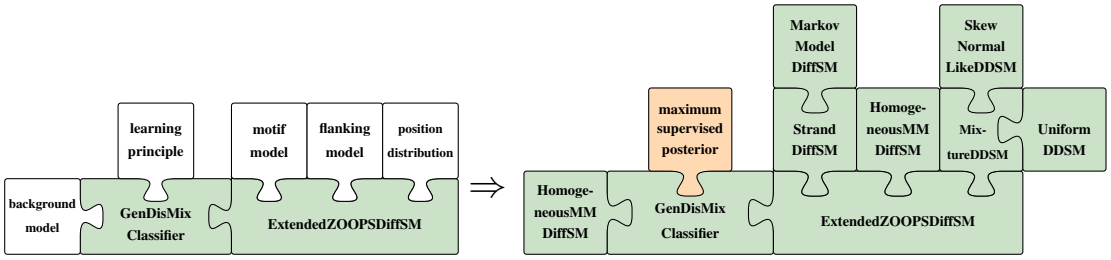

Figure 2: Structure of Dispom within Jstacs. The left side illustrates, how modules can be plugged into the core structure. The right side shows the concrete classes used in the application.

differ in the employed learning principle and in the capability of learning the positional preference of motif occurrences. However, prior to Dispom, no approach existed for learning the motif and the positional preference simultaneously using a discriminative learning principle. The general structure of Dispom in Jstacs is depicted on the left side of Figure 2, where each white piece represents a slot that can be filled with implementations of interfaces defined in Jstacs.

The motif, flanking, and background model are `DifferentiableStatisticalModel`s, the position distribution is a `DurationDiffSM`, and the learning principle is a value from an `enum` type. In the Dispom application illustrated on the right side of Figure 2, we use an inhomogeneous Markov model of order 0 with a mixture over the DNA-strands as motif model. We use homogeneous Markov models of order 0 for both the flanking and background model. All of these models existed before we started developing Dispom. We use a mixture of a skew normal and a uniform distribution as position distribution, and the discriminative MSP learning principle.

This modular structure allowed for an easy adaption to other problems like challenge 2 of DREAM5[2] on protein binding microarray data, where we simply increased the orders of the motif, flanking, and background model, and extended the learning principle to a weighted variant of the MSP principle. These minimal changes were all that was needed for developing a novel application for the analysis of protein binding microarrays and a successful performance in the challenge.

4. Author Contributions

JG, AG, SP and IG initiated the project and JG and AG were main developers. Later, JK joined as a main developer, and AG contributed occasionally. BH contributed to parts of Jstacs and to documentation. All authors contributed to writing and approved the final manuscript.

Acknowledgments

We thank Michael Scharfe and Michael Seifert for their contributions to Jstacs, and Ralf Eggeling, Martin Gleditzsch, Michaela Mohr, Birgit Möller, Martin Porsch, and Marc Strickert for valuable discussions. This work was supported by grant 0312706A by the German Ministry of Education and Research (BMBF) and grant XP3624HP/0606T by the Ministry of Culture of Saxony-Anhalt.

2. A description of the challenge is available at `http://wiki.c2b2.columbia.edu/dream/index.php/D5c2`.

References

Thomas Abeel, Yves Van de Peer, and Yvan Saeys. Java-ML: A machine learning library. *Journal of Machine Learning Research*, 10:931–934, June 2009.

Jesús Cerquides and Ramon López de Mántaras. Robust Bayesian linear classifier ensembles. In *Proceedings of the 16th European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2005.

Peter Grünwald, Petri Kontkanen, Petri Myllymäki, Teemu Roos, Henry Tirri, and Hannes Wettig. Supervised posterior distributions. Presented at the Seventh Valencia International Meeting on Bayesian Statistics, 2002. URL `http://homepages.cwi.nl/~pdg/presentations/Valencia2.pdf`.

Jens Keilwagen, Jan Grau, Stefan Posch, Marc Strickert, and Ivo Grosse. Unifying generative and discriminative learning principles. *BMC Bioinformatics*, 11(1):98, 2010.

Jens Keilwagen, Jan Grau, Ivan A. Paponov, Stefan Posch, Marc Strickert, and Ivo Grosse. De-novo discovery of differentially abundant transcription factor binding sites including their positional preference. *PLoS Computational Biology*, 7(2):e1001070, 02 2011.

Sören Sonnenburg, Gunnar Rätsch, 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, August 2010.