Journal of Machine Learning Research 9 (2008) 993-996 Submitted 10/07; Revised 3/08; Published 6/08

Shark

Christian Igel CHRISTIAN.IGEL@NEUROINFORMATIK.RUB.DE
Verena Heidrich-Meisner VERENA.HEIDRICH-MEISNER@NEUROINFORMATIK.RUB.DE
Tobias Glasmachers TOBIAS.GLASMACHERS@NEUROINFORMATIK.RUB.DE
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

Editor: Soeren Sonnenburg

Abstract

SHARK is an object-oriented library for the design of adaptive systems. It comprises methods for single- and multi-objective optimization (e.g., evolutionary and gradient-based algorithms) as well as kernel-based methods, neural networks, and other machine learning techniques.

Keywords: machine learning software, neural networks, kernel-methods, evolutionary algorithms, optimization, multi-objective-optimization

1. Overview

SHARK is a modular C++ library for the design and optimization of adaptive systems. It serves as a toolbox for real world applications and basic research in computational intelligence and machine learning. The library provides methods for single- and multi-objective optimization, in particular evolutionary and gradient-based algorithms, kernel-based learning methods, neural networks, and many other machine learning techniques. Its main design criteria are flexibility and speed. Here we restrict the description of SHARK to its core components, albeit the library contains plenty of additional functionality. Further information can be obtained from the HTML documentation and tutorials. More than 60 illustrative example programs serve as starting points for using SHARK.

2. Basic Tools—Rng, Array, and LinAlg

The library provides general auxiliary functions and data structures for the development of machine learning algorithms. The Rng module generates reproducible and platform independent sequences of pseudo random numbers, which can be drawn from 14 predefined discrete and continuous parametric distributions. The Array class provides dynamical array templates of arbitrary type and dimension as well as basic operations acting on these templates. LinAlg implements linear algebra algorithms such as matrix inversion and singular value decomposition.

3. ReClaM—Regression and Classification Methods

The goal of the ReClaM module is to provide machine learning algorithms for supervised classification and regression in a unified, modular framework. It is built like a construction kit, where the main building blocks are adaptive data processing models, error functions, and optimization

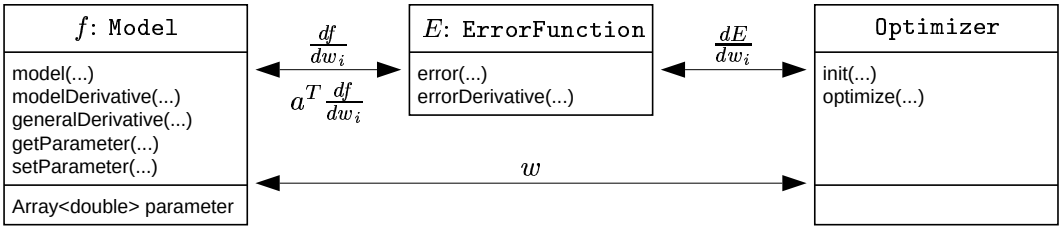

Figure 1: Almost all ReClaM objects are inherited from one of the three base classes `Model`, `ErrorFunction`, and `Optimizer`. The optimizer has access to the parameter vector w of the model $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$, $(x, w) \mapsto f_w(x)$, to minimize a scalar error function E. For gradient-based optimization, the error function provides the derivative dE/dw based on df/dw. In many cases we can speed up the computation of dE/dw by a factor of m by using $a^T df/dw$, where a is a vector of coefficients dependent on the error function.

algorithms (see Figure 1). The superclasses representing these components communicate through fixed interfaces. Problem definition and solution are clearly separated. A problem is defined by a model defining a parametric family of candidate hypotheses, and a possibly regularized error function to minimize (and, of course, sample data). It is usually solved with an (iterative) optimization algorithm, which adapts the model parameters in order to minimize the error function evaluated on the given data set. Additional error functions and data sets can then be used to test the resulting performance. This clear structure makes ReClaM easy to use and extend.

ReClaM focuses on kernel methods and neural networks. It offers a variety of predefined network models including feed-forward and recurrent multi-layer perceptron networks, radial basis function networks, and CMACs. Several gradient-based optimization algorithms are available for network training and general purpose optimization including the conjugate gradient method, the BFGS algorithm, and improved Rprop (Igel and Hüsken, 2003).

In the remainder of this section we present the realization of kernel-based learning in more detail. The library offers kernelized versions of several learning machines from nearest neighbor classifiers and simple Gaussian processes to different flavors of support vector machines. These algorithms operate on general kernel objects and users can supply new kernel functions easily. At the time of writing, ReClaM provides the fastest support vector machine (SVM) implementation for dense large-scale learning problems. The SVM training automatically switches between the most efficient SMO-like algorithms available depending on the current problem size (Fan et al., 2005; Glasmachers and Igel, 2006).

On top of these models, ReClaM defines meta-models for model selection of kernel and regularization parameters. It offers more objective functions and optimization methods for model selection than any other library. Objective functions include leave-one-out and cross validation errors, radius-margin quotient, kernel-target alignment, and the span bound (Chapelle et al., 2002; Glasmachers and Igel, 2005; Igel et al., 2007a). For optimization, nested grid-search and evolutionary kernel learning are supported, and efficient gradient-based optimization is available whenever possible. For both model training and model selection, we make use of ReClaM's superclass architecture to describe and solve the optimization problems. For example, a gradient-based optimization algorithm

may decrease a radius-margin quotient in order to adapt the hyperparameters of an SVM, where in each iteration an SVM model is trained by a special quadratic program optimizer to determine the margin.

To reduce the complexity of SVMs and Gaussian processes after training, algorithms for approximating the solutions in feature space are implemented (Romdhani et al., 2004; Suttorp and Igel, 2007).

4. EALib and MOO-EALib—Evolutionary Single- and Multi-objective Optimization

The evolutionary algorithms module (EALib) implements classes for stochastic direct optimization using evolutionary computing, in particular genetic algorithms and evolution strategies (ESs). Evolutionary algorithms (EAs) maintain populations (i.e., multi-sets) of candidate solutions. In the EALib structure, instances of the class `Population` contain instances of `Individual` consisting of one or more `Chromosome`s, which can have different types. Numerous variation (i.e., mutation and recombination) operators for different types of chromosomes, for example real-valued or binary vectors, are available. The user has the choice between many different deterministic and stochastic selection mechanisms operating on population level.

The MOO-EALib extends the EALib to evolutionary multi-objective (i.e., vector valued) optimization (EMO). The goal of EMO is usually to approximate the set of Pareto-optimal solutions, where a solution is Pareto-optimal if it cannot be improved in one objective without getting worse in another one. To our knowledge, the MOO-EALib module makes SHARK one of the most comprehensive libraries for EMO. The efficient implementation of measures for quantifying the quality of sets of candidate solutions is a strong argument for the MOO-EALib.

In SHARK we put an emphasis on variable-metric ESs for real-valued optimization. Thus, the most recent implementation of the covariance matrix adaptation ES (CMA-ES; Hansen et al., 2003) and its EMO counterpart (Igel et al., 2007b) are included. We do not know any C++ toolbox for EAs that comes close to the EALib in terms of flexibility and quality of algorithms for continuous optimization.

5. Availability and Requirements

The C++ source code is available from `http://shark-project.sourceforge.net` under GNU Public License and compiles under MS Windows, Linux, Solaris, and MacOS X. No third-party libraries are required, except Qt and Qwt for graphical examples.

Acknowledgments

The authors of this paper comprise the team responsible for a major revision and the maintenance of the SHARK library at the time of writing the article. The SHARK project was started by M. Kreutz, who wrote the basic components such as LinAlg, Array, and Rng as well as the EALib. Then B. Sendhoff joined the project, which was fused with C. Igel's ReClaM library. Afterwards, many people contributed to the package, in particular (in alphabetic order) R. Alberts, T. Bücher, A. W. Dietrich, who invented the name Shark, T. Glasmachers, who extended the ReClaM library, M. Hüsken, T. Okabe, who wrote the MOO-EALib, S. Roth, P. Stagge, T. Suttorp, M. Toussaint, and T. Voß. The SHARK project is supported by the Honda Research Institute Europe.

References

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Glasmachers and C. Igel. Gradient-based adaptation of general Gaussian kernels. *Neural Computation*, 17(10):2099–2105, 2005.

T. Glasmachers and C. Igel. Maximum-gain working set selection for support vector machines. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11 (1):1–18, 2003.

C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

C. Igel, T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):216–226, 2007a.

C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007b.

S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Efficient face detection by a cascaded support-vector machine expansion. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 460(2051):3283–3297, 2004.

T. Suttorp and C. Igel. Resilient simplification of kernel classifiers. In J. Marques de Sá et al., editors, *International Conference on Artificial Neural Networks (ICANN 2007)*, volume 4668 of *LNCS*, pages 139–148. Springer-Verlag, 2007.