StatPatternRecognition in Analysis of HEP and Astrophysics Data

I. Narsky
California Institute of Technology, Pasadena, CA, USA

Abstract
StatPatternRecognition (SPR) is a C++ package for supervised machine learning. Introduced in 2005, it has been used by several HEP and astrophysics collaborations, as well as non-academic researchers, for analysis of complex multivariate data. The package implements powerful classification algorithms such as boosting (three flavors), arc-x4, bagging, random forest, neural networks, decision trees (two flavors), bump hunter (PRIM), multi-class learner, logistic regression, linear and quadratic discriminant analysis, combiner of classifiers, and others. It also offers a suite of tools for data analysis: estimation of variable importance, bootstrap, cross-validation, computation of data moments, multivariate goodness-of-fit estimation, and others. SPR is a standalone package with an optional dependency on Root for data input/output. The user can access major SPR methods from an interactive Root session by loading the SPR shared library. The package is under active development and shows a growing number of users in the HEP community and elsewhere. The latest source release of the package can be obtained under General Public License from Sourceforge [1]. A full list of notes and talks about the package can be found on the author's web page [2].

1 Introduction

For several decades the HEP community has been using various classification methods to separate signal from background. Among these methods, only binary decision splits, also known as "cuts" in physics jargon, Fisher discriminant [3], and neural networks [4] have become truly popular. Stimulated by discussion at the Phystat workshops and related publications in physics journals and web archives, the community is now exploring new powerful classifiers recently introduced in the statistics literature. In particular, boosted decision trees [5] and random forest [6, 7] are becoming increasingly popular.

One cannot apply these advanced methods to physics data without software. In the past, physicists used to adapt packages from other communities or write their own implementations of desired algorithms. This practice is still ongoing. Both approaches require a substantial investment of manpower and often involve replication of effort. A package that can be used for physics analysis off the shelf should reduce this waste of effort to minimum.

What are the code requirements for such a package? First and foremost, it must be written in C++. It is by far the most popular choice among HEP researchers and the base for software frameworks maintained by large HEP collaborations. A package for multivariate classification must implement various methods and provide tools for comparison of their performance on the same dataset. Such a package should offer methods particularly useful for physics analysis, for example, optimization and monitoring of HEP-specific figures of merit (FOM's) such as the signal significance $S/\sqrt(S + B)$, a 90% upper limit and others. One of the distinctive features of HEP analysis is the enormous amount of experimental data available. Thus, the package should perform well on big datasets in many dimensions. This package needs to be interfaced to Root [8], a widely accepted framework for data storage and access within the HEP and astrophysics communities. Prior to SPR, such a package was not available.

2 Distribution

SPR is distributed as source code off Sourceforge [1] under General Public License. Installation instructions are included in INSTALL. Users reported successful builds on 32-bit (Scientific Linux 3 and 4, RedHat Enterprise 4, and Solaris 9) and 64-bit Unix platforms (Fedora and Debian). Enthusiasts have adapted the package to Windows and MacOS. SPR has been included as an extra package in Fedora Core 6 and later versions.

I am committed to supporting two versions of the package: a standalone version that uses ASCII text for input and output of data, and a Root-dependent version. The user can choose between the two versions during installation by setting an appropriate parameter of the configuration script. The ASCII version found consumers outside of the HEP community, while the Root-dependent version is more popular among physicists. No graphical tools are offered for the ASCII version of the package; however, one can go through the full analysis chain using ASCII output from SPR executables, as long as one can tolerate digesting information in the form of text tables instead of plots.

From day one it has been my goal to deliver a package ideal for CPU-intensive long-running job batch submissions. A typical HEP user trains boosted decision trees or random forest on datasets with up to millions of events in up to hundreds of dimensions. The package includes two dozen executables, one for each implemented classifier plus other analysis methods. For the most part, graphical tools have not been in the focus of development. However, recently I introduced SprRootAdapter — this class wraps SPR functionality in a shared library that can be loaded in Root. Now the user has access to major SPR methods from an interactive Root session. Scripts for making Root plots of various SPR quantities are also provided. For Root users, this should make the graphical analysis of SPR data much easier.

Documentation is supplied in the README file distributed with the package. All implemented methods and executables are described in sufficient detail.

3 Methods

It is impossible to fit a description of all SPR algorithms into a 4-page note. It is not necessary either because these algorithms have been described in many books on machine learning, advertised in several recent publications by physicists, and discussed at numerous seminars and workshops. Below is a brief summary of what is available.

The full analysis chain for a classifier of choice consists of training and testing. At the training stage, the user creates a trainable classifier and trains it for a specified number of cycles, either by supplying parameters for this classifier to one of the SPR executables or by using the SprRootAdapter interface from an interactive Root session. For a classifier that requires more than one training cycle, the user can monitor classification error computed for validation data. After the training is completed, the user can save the trained classifier configuration into a file. The saved configuration contains full information about the classifier. At any time the user can read the saved configuration from the file back into memory and either continue accumulating more training cycles or apply the trained classifier to test data. SPR allows to read configurations from several files and apply them to test data simultaneously turning the classifier comparison into an easy task.

SPR decision trees [9, 10] come in two flavors — a “regular” decision tree and a top-down tree. They use the same algorithm for training but store their configurations in different formats. A “regular” tree stores its terminal nodes as rectangular regions. If the number of nodes is small, this tree can be easily interpreted by a human. A top-down tree stores its configuration as a full path from the root of the tree. A top-down tree is faster because the lookup time grows as $\log(N)$ versus the number of nodes N while for the “regular” tree the lookup time grows linearly.

The bump hunter algorithm [11] finds one rectangular region in a multidimensional space by optimizing a chosen FOM. Both the bump hunter and the decision tree can optimize FOM’s widely used in the machine-learning research such as Gini index or cross-entropy, as well as FOM’s suited for physics analysis such as $S/\sqrt{S+B}$, a 90% upper limit and others.

Boosting [5] works by adding many weak classifiers sequentially and increasing weights of misclassified events at each step. By focusing on events that are misclassified most of the time, boosting typically achieves a very good predictive power. SPR implements three flavors of boosting: Discrete AdaBoost, Real AdaBoost, and ϵ-Boost. Although boosted decision trees have lately become popular in HEP analysis, one can successfully boost other classifiers as well. Boosted binary splits and boosted neural networks are two other typical applications of boosting found in the machine-learning literature. SPR allows the user to boost an arbitrary sequence of classifiers.

The bagging (bootstrap aggregation) algorithm [6, 12] averages over many weak classifiers built on bootstrap replicas of a training set. SPR allows the user to bag an arbitrary sequence of classifiers. Bagged decision trees have been often used in machine-learning research and are now being applied to physics analysis, in particular, for particle identification at *BABAR*. Another popular method, bagged neural networks, will hopefully find its way into physics analysis as well.

Random forest [7], typically used in conjunction with bagging, represents a set of decision trees. Each tree is built using randomly selected input variables for each decision split. Random sampling of input variables reduces correlation among the decision trees and improves the overall classification power. This method has been applied with success to several *BABAR* physics analyses.

SPR implements a feedforward backpropagation neural net with a logistic activation function [4] well known to physicists.

SPR implements a tool for combining several powerful classifiers. One can train several classifiers on subsets of input variables and then train a global classifier in the space of their outputs. The user needs to specify how the classifiers are to be combined through a configuration file.

All algorithms described above can be only used for separation of two classes, signal and background. A multi-class method [13] reduces a problem with an arbitrary number of classes to a set of two-class problems and then converts the solutions to these binary problems into an overall multi-category classification label.

SPR offers various methods for estimation of variable importance. For decision trees, the importance of an input variable can be estimated by adding changes in the optimized FOM due to decision splits on this variable. For any classifier, the importance of an input variable can be estimated by randomly permuting class labels across this variable and estimating an increase in the overall classification error due to this permutation.

SPR implements other tools for data analysis that will not be described here due to limited space.

4 Examples of Use

In 2005 the SPR implementation of the random forest algorithm was applied to muon identification at *BABAR*, and a significant improvement over the traditional neural-net approach was achieved. This exercise, presented at the CHEP 2006 and ACAT 2007 conferences, instigated development of an SPR muon selector. At present the *BABAR* PID group is working on electron, proton and kaon SPR selectors as well. An SPR-based K_L selector is also available. These selectors outperform likelihood-, cut- and neural-net-based selectors that are still used in *BABAR*. An SPR-based electron selector has been recently introduced at CMS.

Several *BABAR* physics analyses use SPR methods for background suppression. A search for $B^+ \to K^+\nu\nu$ and a measurement of the branching fractions for the decays $B \to \rho\gamma$ and $B \to \omega\gamma$ exploit the SPR bump hunter algorithm to find the optimal combination of orthogonal cuts and apply the random forest method with dozens or hundreds of input variables to achieve the ultimate classification power. SPR boosted decision trees are used in a search for $B^+ \to \tau^+\nu$ and a measurement of exclusive $b \to s\gamma$ modes.

There are several published results as well. SPR boosted decision trees and random forest were applied to identify supernovae [14]. 32 input variables were used to separate signal modeled as fake supernovae inserted into real sky images from background. The decision tree methods reduce background by 1-2 orders of magnitude in a broad range of the true positive identification rate compared to the traditional threshold-cut approach and a support vector machine classifier.

Another study [15] used SPR boosted decision trees for tagging of b-jets. $W \to l\nu q\bar{q}$ events were generated by simulating the environment of the LHC collider with $p\bar{p}$ collisions at a center-of-mass energy of 14 TeV. The training sample consisted of 50k b-jet events and 50k u-jet events with 7 input variables. For moderate b-tagging efficiencies, the boosted decision trees improve the u-jet rejection by several dozen percent compared to the multi-layer perceptron implemented in Root.

This is by far an incomplete list of analyses using SPR. I have little or no knowledge of how the package is used by collaborations other than BaBar or CMS, and even less so by analysts outside the HEP and astrophysics communities.

5 Acknowledgments

Many people have contributed to SPR development by submitting code, offering advice, providing feedback about the package installation and use, and by presenting the package at conferences on my behalf. A full list of code contributors can be found in the AUTHORS file included in the package. I would like to thank the organizers of Phystat07 for giving me an opportunity to present this work at the conference.

References

[1] `http://sourceforge.net/projects/statpatrec/`.

[2] `http://www.hep.caltech.edu/~narsky/spr.html`.

[3] R.A. Fisher, *The use of multiple measurements in taxonomic problems*, Annals of Eugenics **7**, 179-188 (1936).

[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[5] Y. Freund and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, J. of Computer and System Sciences **55**, 119-139 (1997); J. Friedman, T. Hastie and R. Tibshirani, *Additive Logistic Regression: a Statistical View of Boosting*, Annals of Statistics **28(2)**, 337-407 (2000).

[6] L. Breiman, *Bagging Predictors*, Machine Learning **26**, 123-140 (1996).

[7] L. Breiman, *Random Forests*, Machine Learning **45**, 5-32 (2001).

[8] `http://root.cern.ch/`.

[9] L. Breiman et al., *Classification and Regression Trees*, Waldsworth International, 1984.

[10] I. Narsky, *StatPatternRecognition: A C++ Package for Statistical Analysis of High Energy Physics Data*, physics/0507143 (2005).

[11] J. Friedman and N. Fisher, *Bump hunting in high dimensional data*, Statistics and Computing **9**, 123-143 (1999).

[12] I. Narsky, *Optimization of Signal Significance by Bagging Decision Trees*, physics/0507157 (2005).

[13] E.L. Allwein, R.E. Schapire and Y. Singer, *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers*, J. of Machine Learning Research **1**, pp. 113–141 (2000).

[14] S. Bailey et al., *How to Find More Supernovae with Less Work*, The Astrophysical Journal **665**, 1246-1253 (2007); arXiv:0705.0493 [astro-ph].

[15] J. Bastos, *Tagging heavy flavours with boosted decision trees*, arXiv:physics/0702041; J. Bastos and Y. Liu, *A Multivariate approach to heavy flavour tagging with cascade training*, arXiv:0704.3706 [physics.data-an].