StatPatternRecognition: A C++ Package for Multivariate Classification of HEP Data

I. Narsky, California Institute of Technology, Pasadena, CA, USA[*]

Abstract

Modern analysis of HEP data needs advanced statistical tools to separate signal from background. A C++ package has been implemented to provide such tools for the HEP community. The package includes linear and quadratic discriminant analysis, decision trees, bump hunting (PRIM), boosting (AdaBoost and arc-x4), bagging and random forest algorithms, a multi-class learner, and interfaces to the standard backpropagation neural net and radial basis function neural net implemented in the Stuttgart Neural Network Simulator. Supplemental tools such as bootstrap, estimation of data moments, a test of zero correlation between two variables with a joint elliptical distribution, and a multivariate goodness-of-fit method are also provided. The package offers a convenient set of tools for imposing requirements on input data, storing output into Root or Hbook, and handling multi-class data. Integrated in the *BABAR* computing environment, the package maintains a minimal set of external dependencies and can be easily adapted to any other HEP environment. It has been tested on many idealistic and realistic examples.

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 [1], and neural networks [2] have become truly popular. Stimulated by discussion at 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. Two such classifiers, boosted decision trees and random forest, are briefly described in this note.

Application of these advanced methods to physics data, of course, requires software. Surprisingly, despite the great abundance of free software packages, finding one suitable for HEP analysis is not easy. C++ is the natural choice for code that needs to be integrated in the HEP computing environment. Besides boosted decision trees and random forest, the package would have to implement linear discriminant analysis and a bump hunting algorithm for optimization of rectangular cuts because these two methods are often used by HEP analysts. Implementations of the bump hunter and decision trees would have to include optimization of non-standard figures of merit (FOM's) suited for physics analysis such as, for example, the signal significance $S/\sqrt{S+B}$, where S and B are signal and background, respectively, expected in the signal region. Prior to StatPatternRecognition (SPR), such a package was not available.

The package is described in greater detail in two notes posted at the physics archive [3, 4]. Links to all publications and talks about the package can be found on my home page [5].

METHODS

Cut-based analysis of physics data can be carried out using the PRIM algorithm [6]. This algorithm finds a user-specified number of rectangular regions (bumps) with excess of signal events above background. Optimization of each bump is done in two stages — shrinkage and expansion. At the shrinkage stage, the bump hunter gradually reduces the size of the signal box by imposing binary splits. The most optimal split is found at each iteration by searching through all possible splits in all input variables. The rate of shrinkage is controlled by a "peel" parameter, the maximal fraction of events that can be peeled off the signal box with one binary split. This parameter is supplied by the user and can be used to adjust the level of conservatism and the rate of convergence for the algorithm. If the bump hunter cannot find a binary split to improve the FOM, shrinkage is stopped. At the expansion stage, the hunter attempts to relax the bounds of the signal box to optimize the FOM further. After the signal box has been found, the hunter removes points located inside this box from the original data set and starts a new search from scratch.

The implementation of the bump hunter in SPR is capable of optimizing an arbitrary FOM plugged through an abstract C++ interface. In most physics applications, the user needs to find one rectangular region by optimizing a FOM suited for physics analysis such as the signal significance $S/\sqrt{S+B}$, signal purity $S/(S+B)$, potential for discovery $2(\sqrt{S+B}-\sqrt{B})$ [7], Punzi's criterion [8], 90% Bayesian upper limit [9] and others.

Unlike the bump hunter, a decision tree recursively splits training data into rectangular regions (nodes). For each node, the tree examines all possible binary splits in each dimension and selects the one with the highest FOM. This procedure is repeated until a stopping criterion, the minimal number of events per tree node, is satisfied. The tree continues making new nodes until it is composed of leaves only — nodes that cannot be split without a decrease in the FOM and nodes that cannot be split be-

[*] narsky@hep.caltech.edu

cause they have too few events. A conventional decision tree [10] uses Gini index, $Q(p,q) = -2pq$, or cross-entropy, $Q(p,q) = p\log p + q\log q$, for split optimization, where p and $q = 1-p$ are fractions of correctly classified and misclassified events in a given node. If a parent node with the total event weight W is split in two daughter nodes with weights W_1 and $W_2 = W - W_1$, the best decision split is chosen to maximize $Q_{\text{split}} = (W_1Q_1 + W_2Q_2)/W$, where Q_1 and Q_2 are FOM's computed for the two daughter nodes. A conventional decision tree therefore treats the two categories, signal and background, symmetrically. In HEP analysis, one usually wishes to optimize an asymmetric FOM. SPR offers a modified splitting algorithm for this purpose. The best decision split is now chosen to maximize $Q_{\text{split}} = \max(Q_1, Q_2)$, where Q_1 and Q_2 can be asymmetric figures of merit for the daughter nodes, as listed in the previous paragraph. After the tree is grown, the terminal nodes are merged to optimize the overall asymmetric FOM. The merging algorithm sorts all terminal nodes by signal purity in descending order and computes the overall FOM for the n first nodes in the sorted list with n taking consecutive values from one to the full length of the list. The optimal combination of the terminal nodes is given by the highest FOM computed in this manner.

Boosting [11], bagging [12], random forest [13], and the arc-x4 algorithm [14] are powerful methods for multivariate classification. These methods combine many weak classifiers, e.g., decision trees, into an ensemble with a high overall predictive power. At each optimization step, a boosting algorithm enhances weights of misclassified events, reduces weights of correctly classified events and trains a new classifier on the reweighted sample. In contrast, a bagging algorithm trains new classifiers on bootstrap replicas [17] of the training set, without changing event weights. After training is completed, events are classified by the majority vote of the trained classifiers. The votes of individual weak classifiers can be weighted (boosting) or uniform (bagging). Random forest [13], typically used in conjunction with bagging, is a technique that randomly selects a subset of input variables for each decision split. This method can make individual trees less correlated with each other and increase the overall predictive power.

Although SPR is capable of boosting or bagging any weak classifier with a complying interface, in practice boosted (or arced) decision trees and random forest are the two most significant methods. Their performance is most spectacular in many dimensions because, unlike neural nets and distance-based classifiers, decision trees do not suffer from the curse of dimensionality. The training instability of a single decision tree is compensated by other participants in the ensemble, thus giving an overall robustness to these algorithms. Boosted trees and random forest typically perform much better than neural nets in dozens or hundreds of dimensions.

Note that boosted trees and random forest work in very different regimes. Random forest typically benefits from large trees with fine leaves. Boosting usually works better with relatively small trees. It is hardly possible to predict which method of these two will perform better for a given problem. I found that the predictive power of the random forest is typically not worse and in many situations better than that of boosted trees. In practice, however, this statement needs to be verified for each particular problem.

A multi-class learning algorithm [15] reduces the classification problem with an arbitrary number of input categories to a set of binary problems with only two categories. This reduction is achieved using a $K \times L$ matrix for K classes and L binary classifiers. Every binary classifier is trained to separate a subset of input categories from a different subset of input categories. One popular strategy, "one against one", is to treat one class as signal, another one as background and ignore all others. In this case, if the base binary classifier treats signal and background symmetrically, one needs $L = K(K-1)/2$ classifiers. Another popular strategy, "one against all", requires $L = K$ binary classifiers with each of them treating one class as signal and all other classes as background. The user can implement any strategy by supplying any matrix $Y_{K\times L}$: Each matrix column must show 1 for classes treated as signal, -1 for classes treated as background and 0 for classes ignored by this classifier. Response of each binary classfier must range from -1 to 1 with negative values used for background-like events and positive values used for signal-like events. After all binary classifiers are trained, events are categorized by selecting the matrix row with minimal categorization error. For example, for the quadratic classification error one obtains

$$\mathcal{E}_k = \frac{1}{L}\sum_{l=1}^{L} (y_{kl} - f_l(x))^2 \ , \qquad (1)$$

where $\mathcal{E}_k$ is the categorization error for class k, y_{kl} is an element of matrix Y, and $f_l(x)$ is the response of the l-th binary classifier for an event with coordinates x. The event is assigned to class k if the categorization error $\mathcal{E}_k$ is minimal in the set $\{\mathcal{E}_k\}_{k=1}^{K}$.

Other methods implemented in SPR are only briefly mentioned here due to limited space. They include linear and quadratic discriminant analysis [1, 16], cross-validation tools [16], bootstrap [17], evaluation of consistency of a correlation coefficient with zero [18], a method for estimation of a multivariate goodness-of-fit [19], and interfaces to the SNNS feedforward backpropagation neural net and radial basis function neural net [20]. The SNNS interfaces can be used only to read saved neural net configurations and apply them to data.

INSTALLATION AND USE

SPR was written for Unix systems with HEP libraries installed. External dependencies include CLHEP [21] for matrix operations and CERNLIB [22] for random number generation and calculation of χ^2 probabilities. Input data can be read either from ascii or Root [23] files. Input data and classifier output can be stored in ascii, Root or

Hbook [24] formats.

For BaBar members, no installation is necessary since the package is included in standard releases. A user outside BaBar must provide a Makefile that resolves references to CLHEP, CERNLIB, and, if desired, Root or Hbook libraries. An example of such a Makefile is posted at my home page [5]. For users outside BaBar, instructions describing how to request a copy of the package are available on my home page as well. A detailed README file is included in the package.

SPR provides no GUI support. Several executables, one for each major method, are included in the package. Each of them offers a flexible set of command-line options. The user can choose classification parameters, select optimization criteria, group input classes in two categories (signal and background) in an arbitrary way, cross-validate, reweight signal events, postpone and resume training, save classifier configurations into ascii files, and read back saved configurations. Advanced users may wish to write their own C++ code, taking the supplied executables as examples, and thus use the package to its full capacity.

EXAMPLES

The package has been tested on physics data with up to one million events in up to 200 dimensions. Below I include two examples in which random forest has shown a substantial improvement over other methods. The performance of each classifier has been optimized using a training set and, if necessary, cross-checked using a validation set. An independent test set has been then used to obtain an unbiased estimate of the predictive power.

Search for the Rare Decay $B^+ \to K^+\nu\nu$ at BaBar

BaBar recently searched for the rare leptonic decay $B^+ \to K^+\nu\nu$ and published an upper limit of 5.2×10^{-5} at 90% confidence level [25]. The expected branching fraction for this decay in the Standard Model is 3.8×10^{-6} [26]. Because right-handed flavor-changing neutral currents can increase this decay rate by an order of magnitude, an observation of this mode at the present level of the experimental sensitivity would signify physics beyond the Standard Model.

The quoted BaBar result, obtained using 82 fb^{-1}, is a combined limit from two separate analyses with semileptonic and hadronic tags, comparable in experimental sensitivity. We re-analyzed the semileptonically tagged dataset using two classifiers implemented in SPR: the bump hunter and random forest. After the training dataset has been reduced to several hundred thousand events by imposing simple cuts, we train these classifiers using 60 input variables. The rectangular signal region found by the bump hunting algorithm using Punzi's FOM [8] can be directly compared with a set of cuts used in the original analysis [25]. As shown in Fig. 1, the sensitivity of the bump hunter is very close to that of the original analysis; the nine variables selected by the algorithm show some overlap with those used previously. To apply the random forest technology, we train 50 decision trees with at least 5 events per leaf randomly selecting (with replacement) 30 input variables as candidates for each decision split. Random forest gives a substantial improvement over the sensitivity of the rectangular cuts reducing the expected background for the same amount of signal by a factor of $\sim$3. It has been shown that all 60 variables contribute to the background suppression for the random forest algorithm.

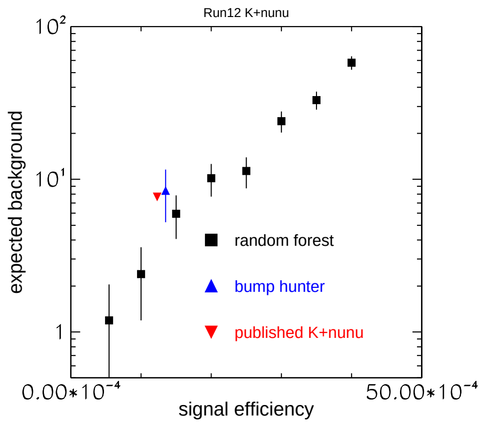

Figure 1: Background expected in 82 fb^{-1} of data versus signal efficiency obtained by various classification methods. These estimates were obtained using generic background Monte Carlo and a large sample of signal Monte Carlo events.

Muon Identification at BaBar

Several selectors are used at BaBar for muon identification. At present the most powerful muon selector uses a two-stage neural net approach. At first, ten variables measured by the electromagnetic calorimeter (EMC) are used as inputs to a neural net with one hidden layer. Then seven variables measured by the instrumented flux return (IFR) and the output of the EMC neural net are used as inputs to a global neural net, also with one hidden layer. Events without EMC or IFR information are rejected, i.e., effectively classified as “not muons”.

To improve the muon purity, we train random forest using all 17 variables from both subsystems at once. We divide data in two bins of momentum ($0.5 < p < 2.0$ GeV$/c$ and $2.0 < p < 4.0$ GeV$/c$) and two bins of polar angle (barrel and endcap) with several dozen or hundred thousand events in a typical training set. We build 100 decision trees with minimal leaf size varying from 5 to 50 events and with the number of randomly selected (with replacement) variables for each decision split varying from 8 to 17. Events without EMC or IFR information are also included in the training set with missing quantities coded as negative values outside the physical range. Such treatment of missing values would be impossible in the neural net approach as it would lead to an instability in the training mechanism. Because decision trees can easily deal with mixed continuous and discrete values for the same input variable, random forest gives an excellent classification power in this example. A substantial improvement obtained by application of the random forest over the neural net result for the barrel is shown in Fig. 2.

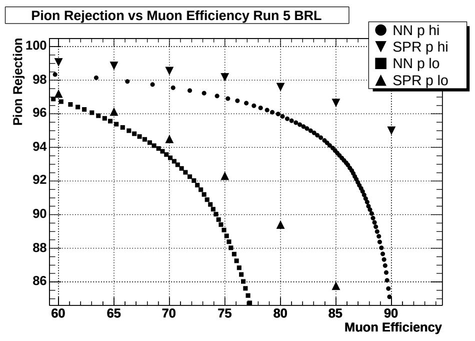

Figure 2: Pion rejection factor versus muon efficiency for the random forest (SPR) and the two-stage neural net algorithm described in the text (NN). These results have been obtained for two momentum ranges, $0.5 < p < 2.0$ GeV$/c$ (low momentum) and $2.0 < p < 4.0$ GeV$/c$ (high momentum), in the barrel part of the detector using BABAR data collected in Run 5.

SUMMARY

A C++ package that includes several powerful classification methods has been developed for the HEP community. Random forest shows a significant improvement over other methods in several analyses. To request a copy of the package, please visit my web page [5].

ACKNOWLEDGMENTS

More than a dozen people have contributed in various ways to development and testing of the package. I would like to thank Akram Khan for his initiative and support in submitting this paper to CHEP. I thank Kevin Flood for providing muon identification datasets and his help in the muon selector analysis. I also thank Julian Bunn for presenting this work at the conference. This work is partially supported by Department of Energy under Grant DE-FG03-92-ER40701.

REFERENCES

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

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

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

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

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

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

[7] S. I. Bityukov and N. V. Krasnikov, *Uncertainties and discovery potential in planned experiments*, hep-ph/0204326 (2002).

[8] G. Punzi, *Sensitivity of Searches for New Signals and Its Optimization*, Proceedings of Phystat2003, `http://www.slac.stanford.edu/econf/C030908/papers/MODT002.pdf`.

[9] O. Helene, *Upper Limit Of Peak Area*, Nucl. Instrum. Meth. **212**, 319 (1983); *Determination Of The Upper Limit Of A Peak Area*, Nucl. Instrum. Meth. A **300**, 132 (1991).

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

[11] 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).

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

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

[14] L. Breiman, *Arcing classifiers*, The Annals of Statistics **26**, 801-849 (1998).

[15] 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).

[16] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, 2001.

[17] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall/CRC, 1993.

[18] T.W. Anderson, *An Introduction to Multivariate Statistical Analysis*, John Wiley & Sons, Inc., 2003.

[19] J. Friedman, *On Multivariate Goodness-of-Fit and Two-Sample Testing*, Proceedings of Phystat2003, `http://www.slac.stanford.edu/econf/C030908/papers/THPD002.pdf`.

[20] University of Stuttgart and University of Tübingen, *Stuttgart Neural Network Simulator: User Manual, Version 4.2*, `http://www-ra.informatik.uni-tuebingen.de/SNNS/`.

[21] *CLHEP — A Class Library for High Energy Physics*, `http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep/index.html`.

[22] `http://wwwasd.web.cern.ch/wwwasd/cernlib/libraries.html`, `http://wwwasdoc.web.cern.ch/wwwasdoc/cernlib.html`.

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

[24] `http://wwwasdoc.web.cern.ch/wwwasdoc/hbook/HBOOKMAIN.html`.

[25] BABAR Collaboration (B. Aubert et al.), *Search for the Decay $B^+ \to K^+\nu\bar{\nu}$*, Phys. Rev. Lett. **94**, 101801 (2005).

[26] G. Buchalla, G. Hiller and G. Isidori, *Phenomenology of nonstandard Z couplings in exclusive semileptonic $b \to s$ transitions*, Phys. Rev. D **63**, 014015 (2001).