TMVA, the Toolkit for Multivariate Data Analysis with ROOT

Andreas Höcker[1], *Peter Speckmayer*[1], *Jörg Stelzer*[1] *Fredrik Tegenfeldt*[2] *and Helge Voss*[3]
[1]CERN, Switzerland
[2]Iowa State U., USA
[3]Max-Planck-Institut für Kernphysik, Heidelberg

Abstract
Multivariate classification methods based on machine learning techniques have become a fundamental ingredient to most physics analyses. The classification techniques themselves have also significantly evolved in recent years. Statisticians have found new ways to tune and to combine classifiers to further gain in performance. Integrated into the analysis framework ROOT, TMVA is a toolkit offering a large variety of multivariate classification algorithms. TMVA manages the simultaneous training, testing and performance evaluation of all the classifiers with a user-friendly interface, and also steers the application of the trained classifiers to data.

1 Introduction

The Toolkit for Multivariate Data Analysis (TMVA) provides a ROOT-integrated framework for the processing and parallel evaluation of many different multivariate classification techniques. The classification is done in terms of two event categories, e.g. signal and background. The idea of TMVA is to integrate a large variety of powerful multivariate classifiers in one common environment with a single interface allowing the user to compare all classification techniques for any given problem. TMVA offers convenient preprocessing possibilities for the data prior to feeding them into any of the classifiers. Auxiliary information about the data is provided such as the correlations between the input variables, their separation power and ranking, various classifier specific validations and finally efficiency versus background rejection curves for all trained classifiers. These criteria allow the user to choose the optimal classifier for the given problem. The package currently includes implementations of:

- Multi-dimensional rectangular cut optimisation using a genetic algorithm or Monte Carlo sampling;
- Projective likelihood estimation;
- Multi-dimensional likelihood estimation (k-nearest neighbour (k-NN) and probability density estimator range-search (PDERS));
- Linear and nonlinear discriminant analysis (Fisher, H-Matrix, Functional Discriminant Analysis);
- Artificial neural networks (three different multilayer percpetron implementations);
- Support Vector Machine;
- Boosted/bagged decision trees with pruning;
- Predictive learning via rule ensembles.

A detailed description of the individual classifiers including the configuration parameters available for their tuning is given in the TMVA Users Guide [1]. TMVA provides training, testing and performance evaluation algorithms, visualisation scripts and auxiliary tools such as parameter fitting and variable transformations.

2 Data Preprocessing, Training and Testing

Training and testing of the classifiers is performed with user-supplied data sets with known event classification. This data is given in form of either ROOT trees or ASCII text files. The data sets are divided into

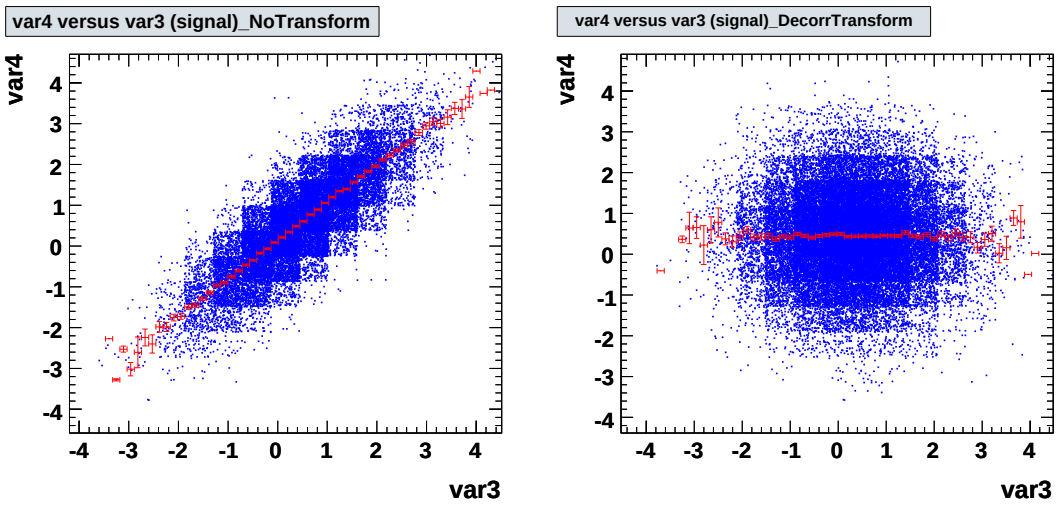

Fig. 1: Correlation between input variables. Left: correlations between var3 and var4 for signal training events from a Gaussian toy Monte Carlo. Right: the same after applying a linear decorrelation transformation.

statistically independent samples of training and testing data, omitting here an independent validation sample. Individual event weights may be attributed when specified in the data set. All classifiers see the same data sets and use the same prescription for the evaluation allowing for an objective comparison between them. A *Factory* class organises the interaction between the user and the TMVA analysis steps including preanalysis and preprocessing of the training data.

During the preanalysis, a preliminary ranking of the input variables is provided and their linear correlation coefficients are displayed. The variable ranking is later superseded by the ranking provided for each of the classifiers.

Preprocessing of the data set includes the application of conventional preselection cuts that are common for all classifiers. In addition one can apply two different variable transformations, decorrelation via the square-root of the covariance matrix and via a principal component decomposition. These transformations can be individually chosen for any particular classifier. Removing linear correlations from the data sample may be useful for classifiers that intrinsically do not take into account variable correlations as for example rectangular cuts or projective likelihood. A demonstration of the decorrelation procedure is shown in Fig. 1. It shows the decorrelation applied to a toy Monte Carlo with linearly correlated and Gaussian distributed variables that is supplied together with the TMVA package.

After the training, each classifier writes the entire information needed for its later application to weight files[1]. The classifiers are then tested and evaluated to assess their performance. The optimal classifier to be used for a specific analysis strongly depends on the problem at hand and no general recommendations can be given. To simplify the choice, TMVA computes and displays for each classifier a number of benchmark quantities such as:

- The *signal efficiency and background rejection* obtained from cuts on the classifier output. The area of the background rejection versus signal efficiency function is used for ranking the different classifiers.
- The *separation* $\langle S^2 \rangle$ of a classifier y, defined by the integral [2]

$$\langle S^2 \rangle = \frac{1}{2} \int \frac{\left(\hat{y}_S(y) - \hat{y}_B(y)\right)^2}{\hat{y}_S(y) + \hat{y}_B(y)} dy \,, \tag{1}$$

[1] A stand alone C++ code of the trained classifier which is independent of the TMVA libraries is also provided.

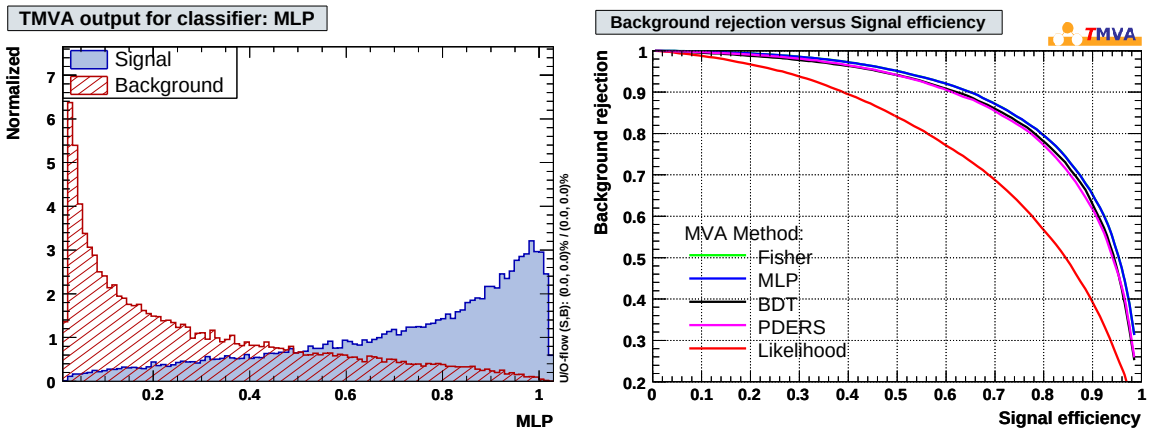

Fig. 2: The top left shows and example plot for classifier output distributions for signal and background events from the Neural Network (MLP) analysis on a toy Monte Carlo data sample. The background rejection versus signal efficiency obtained by cutting on the classifier output for the events of the test sample is shown at the top right.

where $\hat{y}_S$ and $\hat{y}_B$ are the signal and background PDFs of y, respectively. The separation is zero for identical signal and background shapes, and it is one for shapes with no overlap.

– The discrimination *significance* of a classifier, defined by the difference between the classifier means for signal and background divided by the quadratic sum of their root-mean-squares.

A cut placed on the classifier's output value y is typically used to classify an event as either signal or background. Upon user request TMVA also provides the classifier's signal and background PDFs, $\hat{y}_{S(B)}$. The PDFs can be used to derive classification probabilities for individual events. It is also used to compute the *Rarity* transformation.

– *Classification probability*: The probability for event i to be of signal type is given by,

$$P_S(i) = \frac{f_S \cdot \hat{y}_S(i)}{f_S \cdot \hat{y}_S(i) + (1 - f_S) \cdot \hat{y}_B(i)} \,, \tag{2}$$

where $f_S = N_S/(N_S + N_B)$ is the expected signal fraction, and $N_{S(B)}$ is the expected number of signal (background) events (default is $f_S = 0.5$).

– *Rarity*: The Rarity $\mathcal{R}(y)$ of a classifier y is given by the integral [3]

$$\mathcal{R}(y) = \int_{-\infty}^{y} \hat{y}_B(y')\, dy' \,, \tag{3}$$

which is defined such that $\mathcal{R}(y_B)$ for background events is uniformly distributed between 0 and 1, while signal events cluster towards 1. The signal distributions can thus be directly compared among the various classifiers. The stronger the peak towards 1, the better is the discrimination. Another useful aspect of the Rarity is the possibility to directly visualise deviations of a test background (which could be physics data) from the training sample, by exhibition of non-uniformness.

In addition, the variable distributions, correlation matrices and scatter plots, overtraining validation plots, as well as classifier specific information such as likelihood reference distributions, the neural network architecture and decision trees are conveniently plotted using ROOT macros executed via a graphical user interface that comes with TMVA. An example of the output is given in Fig. 2.

3 Classifier Application

The application of the trained classifiers to the selection of events from a data sample with unknown signal and background composition is handled via a light-weight *Reader* object. It reads and interprets the weight files of the chosen classifier and can be included in any C++ executable, ROOT macro or python analysis job.

For standalone use of the trained classifiers, TMVA also generates stand alone C++ response classes for most classifiers, which contain the encoded information from the weight files and the classifier's functionality. These classes do not depend on TMVA or ROOT, neither on any other external library.

4 Summary

TMVA is a toolkit that unifies highly customisable sophisticated multivariate classification algorithms in a single framework thus ensuring convenient use and an objective performance assessment since all classifiers see the same training and test data, and are evaluated following the same prescription.

Emphasis has been put on the clarity and functionality of the *Factory* and *Reader* interfaces to the user applications, which will hardly exceed a few lines of code. All classifiers run with reasonable default configurations and should have satisfying performance for average applications. It is stressed however that, to solve a concrete problem, all classifiers require at least some specific tuning to deploy their maximum classification capability. Individual optimisation and customisation of the classifiers is achieved via configuration strings that are detailed in [1].

TMVA is an open source project. The newest TMVA development version can be downloaded from Sourceforge.net at `http://tmva.sourceforge.net`. It is also part of the standard ROOT distribution kit (v5.14 and higher).

Acknowledgements

The fast growth of TMVA would not have been possible without the contributions from many developers listed as co-authors in the Users Guide [1]) and the crucial feedback from the user community. We thank in particular the CERN summer students Matt Jachowski (Stanford U.) for the implementation of TMVA's MLP neural network and Yair Mahalalel (Tel Aviv U.) for a significant improvement of PDERS. The Support Vector Machine has been contributed to TMVA by Andrzej Zemla and Marcin Wolter (IFJ PAN Krakow), and the k-NN classifier has been written by Rustem Ospanov (Texas U.). We also thank René Brun and the ROOT team for their support.

References

[1] A. Höcker, P. Speckmayer, J. Stelzer, F. Tegenfeldt, H. Voss, K. Voss, A. Christov, S. Henrot-Versillé, M. Jachowski, A. Krasznahorkay Jr., Y. Mahalalel, R. Ospanov, X. Prudent, M. Wolter, A. Zemla arXiv:physics/0703039 (2007).

[2] The BABAR Physics Book, BABAR Collaboration (P.F. Harrison and H. Quinn (editors) *et al.*), SLAC-R-0504 (1998); S. Versillé, PhD Thesis at LPNHE, `http://lpnhe-babar.in2p3.fr/theses/these_SophieVersille.ps.gz` (1998).

[3] To our information, the *Rarity* has been originally defined by F. Le Diberder in an unpublished Mark II internal note. In a single dimension, as defined in Eq. (3), it is equivalent to the μ-transform developed in: M. Pivk, "*Etude de la violation de CP dans la désintégration* $B^0 \to h^+h^-$ $(h = \pi, K)$ *auprès du détecteur BABAR à SLAC*", PhD thesis (in French), `http://tel.archives-ouvertes.fr/documents/archives0/00/00/29/91/index_fr.html` (2003).