TMVA - Toolkit for Multivariate Data Analysis

Jan Therhaag[1]

University of Bonn, Physikalisches Institut, Nußallee 12, 53115 Bonn

Abstract. The Toolkit for Multivariate Analysis (TMVA) provides a ROOT-integrated [2] environment for the processing, parallel evaluation and application of multivariate classification and - since version 4.0.0 - regression techniques which is specifically designed for the needs of high-energy physics (HEP) applications. In TMVA, all MVA methods are embedded in a common framework capable of handling the pre-processing of the data as well as the evaluation of the output, thus allowing a simple and convenient use and assessment of multivariate techniques.

Keywords: TMVA, Multivariate Data Analysis, Neural Networks, Boosted Decision Trees
PACS: 02.50.Sk

INTRODUCTION

In high-energy physics, with the search for ever smaller signals in ever larger data sets, it has become essential to extract the maximum of the available information from the data. Multivariate analysis techniques have proven indispensable in tackling this challenge. The toolkit for multivariate data analysis, TMVA, provides a large set of advanced multivariate analysis techniques for both classification and regression problems including

- Rectangular cut optimisation (binary splits),
- Projective likelihood estimation,
- Multi-dimensional likelihood estimation (PDE range-search, PDE-Foam[3] and k-NN),
- Linear and nonlinear discriminant analysis (H-Matrix, Fisher, LD, FDA),
- Artificial neural networks (three different multilayer perceptron implementations),
- Support vector machine,
- Boosted/bagged decision trees (supporting both AdaBoost and Gradient Boost),
- Predictive learning via rule ensembles (RuleFit),
- A generic boost classifier, allowing one to boost any of the above classifiers,
- A category classifier, allowing for the use of different methods in different phase space regions.

Aside from a convenient assesment of different multivariate techniques, TMVA provides the user with a variety of data preprocessing capabilities and auxiliary information about the data, such as correlations between input variables and a ranking of their discrimination power. Finally, a large range of evaluation and comparison methods assist the user in choosing the best MVA method for any given analysis.

DATA ANALYSIS WITH TMVA

A complete TMVA analysis consists of two phases: The training phase, where the multivariate methods are trained through supervised learning, and the application phase, where the chosen methods are applied to the classification or regression problem in question. An overview of the typical code flow for these two phases is sketched in Fig. 1.

---

[1] on behalf of the TMVA core developer team: A. Hoecker, P. Speckmayer, J. Stelzer, E. von Toerne, H. Voss

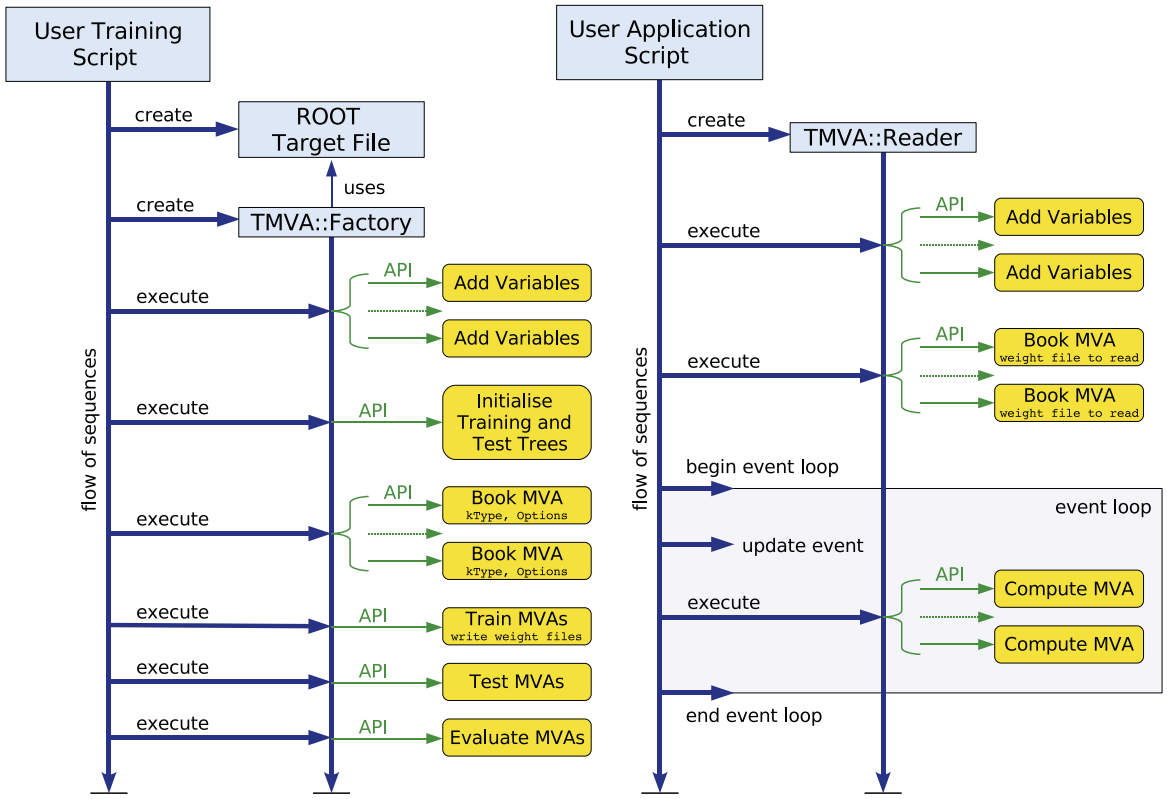

FIGURE 1. Left: Flow (top to bottom) of a typical TMVA training application. The user script can be a ROOT macro, C++ executable, python script or similar. After creation by the user, the Factory organises the user's interaction with the TMVA modules. First the discriminating variables and - in case of regression - the target variables are registered. Then, selected MVA methods are booked and configured via a simple option string. The TMVA analysis proceeds by consecutively calling the training, testing and performance evaluation methods of the Factory. The training results are then written to custom weight files in XML format and the evaluation histograms are stored in the output file.
Right: Flow (top to bottom) of a typical TMVA analysis application. The selected MVA methods are now used to classify data of unknown signal and background composition or to predict a regression target. First, a `Reader` class object is created, which serves as the interface to the method's response. Then the discriminating variables and references to locally declared memory placeholders are registered with the Reader. Finally, the selected MVA methods are booked and fully configured through the weight files produced during the training phase.

Training, testing and evaluation

The TMVA Factory

In order to allow for an unbiased performance assessment and comparison between different MVA methods in the training phase, TMVA works in a transparant factory mode. All interactions between the user and the MVA methods take place via a `Factory` object, which guarantees that all methods see the same training and testing data with the same preprocessing applied. Thus every TMVA analysis begins with the instantiation of a `Factory` object, which then steers the training, testing and evaluation of the booked methods.

Specification of the input data and the desired MVA methods

TMVA supports both ROOT trees and text files as input data sets for training and testing. The `Factory` interface handling the input data is highly flexible - signal and background events can be stored in the same tree or in different trees, precuts can be applied and individual event weights are as well supported as overall weights for entire trees.

Once the input collections are registered with the `Factory`, the input variables that will be used by the MVA methods can be declared. Variable combinations and formulas may be used. The new version of TMVA also supports the use of "spectator variables", which do not take part in the analysis, but may be used for further evaluation and plotting.

All events that are handed to the TMVA `Factory` are internally copied and split into one *training* and one *test* tree to assure a statistically independent evaluation of the MVA methods based on the test sample. The creation of these trees can be preceded by a cut based event selection.

Just like the input variables, all MVA methods have to be registered with the `Factory`. This is done by specifying the method's type, a unique identifier and a specific option string which configures the method and selects possible variable transformations to be applied. TMVA offers a great variety of MVA methods which are highly configurable. Please consult the official TMVA Users Guide [1] for a detailed description.

Transforming the input variables

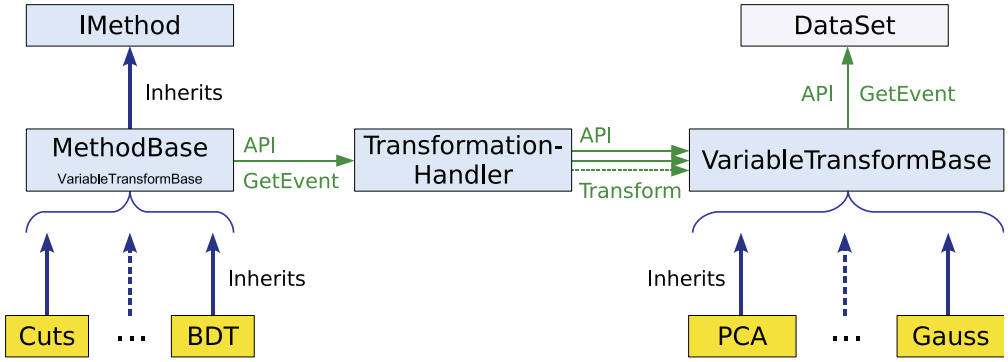

FIGURE 2. Each MVA method inherits from `MethodBase`, which holds a protected member object of type `TransformationHandler`. The `TransformationHandler` consists of a list of objects derived from `VariableTransformBase` which are the implementations of the particular variable transformations available in TMVA.

TMVA offers an efficient interface for performing variable transformations "on the fly", when an event is requested from the central `DataSet` class (cf. Fig. 2). Variable transformations are booked through the MVA option string, which means that each method can have its own transformations. Currently TMVA supports:

- variable normalization,
- decorrelation via the square-root of the covariance matrix or via principal component decomposition,
- transformation of the variables into Gaussian distributions ("Gaussianisation").

Removing linear correlations from the data is most beneficial for simple MVA methods which do not take into account variable correlations, for example rectangular cuts or projective likelihood. Complicated non-linear correlations on the other hand call for more sophisticated classifiers.

Evaluation of the MVA methods

As soon as the training has finished, all MVA methods write their current status information to custom weight files for later application. They are then tested and evaluated to asses their performance. To guide the user in the selection process, TMVA computes a variety of benchmark quantities for each method which are either printed to screen or can be accessed via a graphical user interface. Examples are shown in Fig. 3.

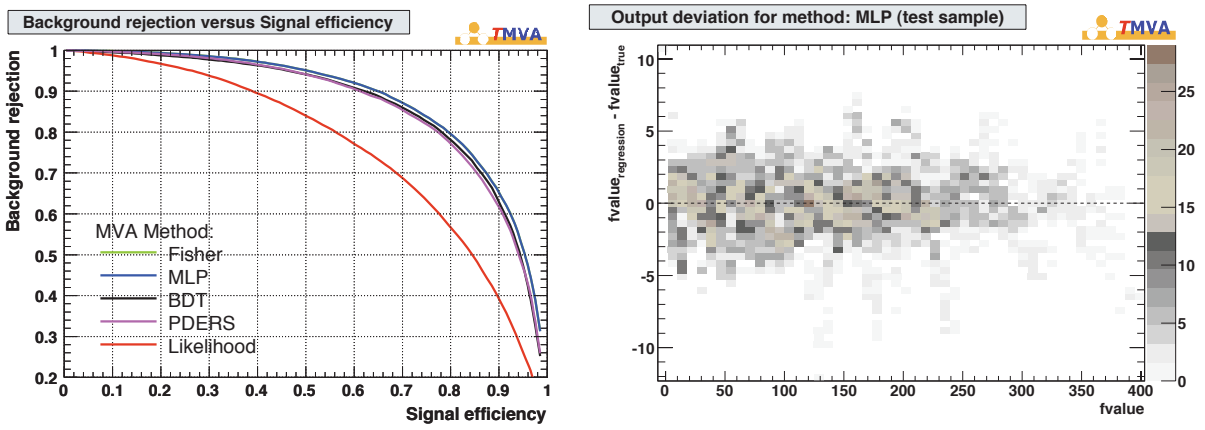

FIGURE 3. Left: Example for the background rejection versus signal efficiency obtained by cutting on the classifier outputs. Right: Example plot for the deviation between regression output and target value for MLP.

Application to data

In the application phase, the most performing MVA methods can be selected by the user to classify events in data samples of unknown composition ot to predict the values of a regression target. TMVA supports the application of trained methods via a `Reader` class or via standalone C++ response classes. The `Reader` is the analogue to the `Factory` and is used in a very similar way: Datasets, variables and methods are registered with the `Reader` which then takes care of possible variable transformations that were applied during the training phase and steers the processing of the data. All information required to configure the MVA methods is automatically retrieved from the weight files after the methods have been registered.
The C++ response classes generated by TMVA are intended for standalone use, they contain the entire information encoded in the weight files but do neither depend on TMVA or ROOT, nor on any other non-standard library.

SUMMARY

TMVA unifies highly customizable multivariate methods for both classification and regression, powerful data preprocessing and convenient evaluation in a single framework. The user interface comprised of the `Factory` and the `Reader` places emphasis on clarity and functionality and will hardly exceed a few lines of code in most applications. TMVA is evolving quickly, the current framework is already compatible with upcoming extensions including classifiers which support multiple classes and automated classifier tuning via cross valiadation.
TMVA is an open source project, the newest version can be obtained from `http://tmva.sourceforge.net`.

ACKNOWLEDGMENTS

The fast growth of TMVA would not have been possible without the contributions from many developers and users listed in the Users Guide [1].

REFERENCES

1. A. Hoecker et al. *TMVA - Toolkit for Multivariate Data Analysis*, (*Preprint* arXiv:physics/0703039), updated 2009
2. R. Brun and F. Rademakers *ROOT - an object oriented data analysis framework*, Nucl. Inst. Meth. in Phys. Res., A 389, 81, 1997
3. D. Dannheim, T.Carli, A. Voigt, K. J. Grahn, P. Speckmayer *PDE-FOAM – a probability-density estimation method based on self-adapting phase-space binning*, Nucl. Instrum. Methods Phys. Res., A 606, 717, 2008