TMVA, the Toolkit for Multivariate Data Analysis with ROOT

Helge Voss[*][†]
Max-Planck-Institut für Kernphysik, Heidelberg
E-mail: `Helge.Voss@cern.ch`

Andreas Höcker
CERN, Switzerland
E-mail: `Andreas.Hocker@cern.ch`

Jörg Stelzer
CERN, Switzerland
E-mail: `Joerg.Stelzer@cern.ch`

Fredrik Tegenfeldt
Iowa State U., USA
E-mail: `Fredrik.Tegenfeldt@cern.ch`

Multivariate classification methods based on machine learning techniques play a fundamental role in today's high-energy physics analyses dealing with ever smaller signal in ever larger data sets. TMVA is a toolkit integrated in the ROOT framework which implements a large variety of multivariate classification algorithms ranging from simple rectangular cut optimisation and likelihood estimators, over linear and non-linear discriminants to more recent developments like boosted decision trees, rule fitting and support vector machines. All classifiers can be trained, tested and evaluated simultaneously. They all see the same training are then afterwards also tested on the same independent test data allowing meaningful comparisons between the methods for a particular use case. Here, an overview about the package and the classifiers currently implemented is presented.

XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research
April 23-27 2007
Amsterdam, the Netherlands

[*]Speaker.
[†]contributors given in arXiv physics/0703039

1. Introduction

The Toolkit for Multivariate Analysis (TMVA) provides a ROOT-integrated environment for the processing and parallel evaluation of sophisticated multivariate classification techniques. The classification is done in terms of two event categories, e.g. signal and background. While TMVA is specifically designed for the needs of high-energy physics (HEP) applications it is not necessarily restricted to those. The idea behind this software package is to provide a large variety of powerful multivariate classifiers in one common environment, allowing for easy use and comparison of the different classification techniques for any give problem. TMVA provides convenient preprocessing possibilities for the data prior to feeding them into any of the classifiers. Many auxiliary information is provided about the data like the correlation matrix, variable ranking and separation power and finally a full efficiency versus background rejection curve of the trained classifiers. With this information the user is in the position to quickly find the optimal data selection procedure for his selection problem. The package currently includes:

- Rectangular cut optimisation;
- Projective likelihood estimation;
- Multi-dimensional likelihood estimation (PDE range-search and k-NN);
- Linear and nonlinear discriminant analysis (H-Matrix, Fisher, FDA);
- Artificial neural networks (three different implementations);
- Support Vector Machine;
- Boosted/bagged decision trees;
- Predictive learning via rule ensembles.

The TMVA software package consists of object-oriented implementations in C++/ROOT for each of these discrimination techniques and auxiliary tools such as parameter fitting and variable transformations. It provides training, testing and performance evaluation algorithms and visualisation scripts. A short description of the various classifiers is given in Sec. 4 while a detailed description including the options available for tuning the individual classifiers is given in the TMVA manual [1].

TMVA is an open source product and distributed either via the ROOT package [2] and separately via sourceforge [3]. Several similar combined multivariate classification ("machine learning") packages exist with rising importance in most fields of science and industry. In the HEP community there is also *StatPatternRecognition* [4, 5]. The idea of parallel training and evaluation of MVA-based classification in HEP has been pioneered by the *Cornelius* package, developed by the Tagging Group of the BABAR Collaboration [6].

2. Data Preprocessing, Training and Testing

Training and testing of the classifiers is performed with the use of user-supplied data sets with known event classification. These data are supplied in form of either ROOT trees or ASCII text files[1]. Individual event weights may be attributed when specified in the data set, allowing the use of Monte Carlo that include weighted events [2]. One is able to select any subset of variables provided in the data set, as well as variable combinations and formulas, just as they are available for the `Draw` command of a ROOT tree. All classifiers see the same training data and are afterwards tested all on the same test data, which is an independet data set from the training data. The evaluation prescriptions are the same for all classifiers. A *Factory* class organises the interaction between the user and the TMVA analysis steps including preanalysis and preprocessing of the training data to assess basic properties of the chosen variables as input to the classifiers.

As preanalysis, the linear correlation coefficients of the input variables are calculated and displayed, and a preliminary ranking is derived (which is later superseded by classifier-specific variable rankings).

Preprocessing of the data set includes the application of conventional preselection cuts that are common for all classifiers. In addition there is the possibility for two variables transformations, decorrelation via the square-root of the covariance matrix and via a principal component decomposition. The latter transformations can be individually chosen for any particular classifier. Each classifier writes *its* transformation into its weight file once the training has converged. The weight files contain all information for later application of the trained classifier. For testing and application of a classifier, the transformation is read from the weight file and a corresponding transformation is then applied during the application of the trained classifier to event data provided in the same format as the original training data.

Removing linear correlations from the data sample may be useful for classifiers that intrinsically do not take into account variable correlations as for example the projective likelihood. Because in most realistic use cases correlations are present, this results in a loss of performance. Also other classifiers, such as rectangular cuts or decision trees, and even multidimensional likelihood approaches underperform in presence of variable correlations.

A demonstration of the decorrelation procedure is shown in Fig. 1. This 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, the classifiers are subjected to testing and evaluation in order to assess their performances. 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 ease the choice TMVA computes and displays a number of benchmark quantities in order to compare the classifiers using the independent test sample. These quantities are

- The *signal efficiency at three representative background efficiencies* (the efficiency is equal to $1-$rejection) obtained from a cut on the classifier output. Also given is the area of the back-

[1]While training and testing is performed on statistically independent data samples to ensure an unbiased testing, a third data sample would be needed for a fully unbiased performance estimate in case parameter tuning of the individual classifiers is involved in the procedure.

[2]Some limitations when using negative event weights may be encountered.

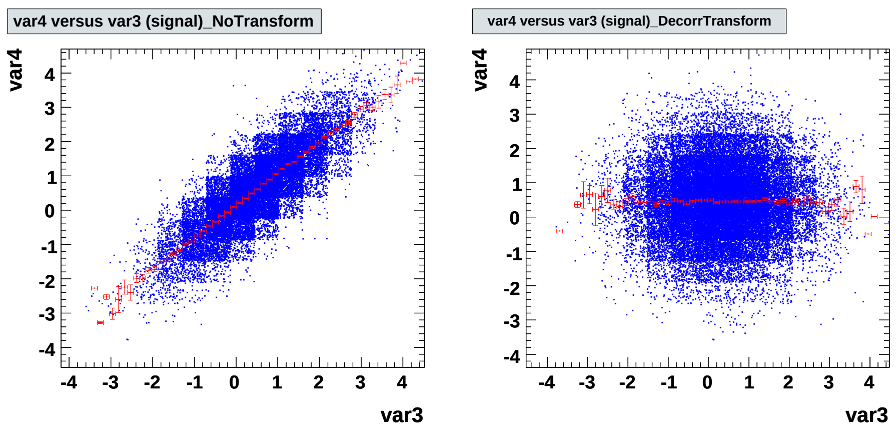

Figure 1: Correlation between input variables. Left: correlations between var3 and var4 for signal training events from a toy Monte Carlo. Right: the same after applying a linear decorrelation transformation.

ground rejection versus signal efficiency function (the larger the area the better the overall performance).

- The *separation* $\langle S^2 \rangle$of a classifier y, defined by the integral [6]

$$\langle S^2 \rangle = \frac{1}{2} \int \frac{(\hat{y}_S(y) - \hat{y}_B(y))^2}{\hat{y}_S(y) + \hat{y}_B(y)} dy\,, \tag{2.1}$$

  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.

In addition, smooth background rejection/efficiency versus signal efficiency curves are written to the target ROOT file. This and many other results and control plots like the MVA-output distributions (Fig. 2), variable distributions, correlation matrices and scatter plots as well as classifier specific information like the neural network architecture are conveniently plotted using custom made ROOT macros executed via a graphical user interface that comes with the TMVA distribution.

The TMVA training/testing job runs alternatively as a ROOT script, as a standalone executable, where TMVA shared library is linked, or as a python script via the PyROOT interface. Each classifier trained in one of these applications writes its configuration and training results in a result ("weight") file.

3. Classifier Application

The application of the trained classifiers to select 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.

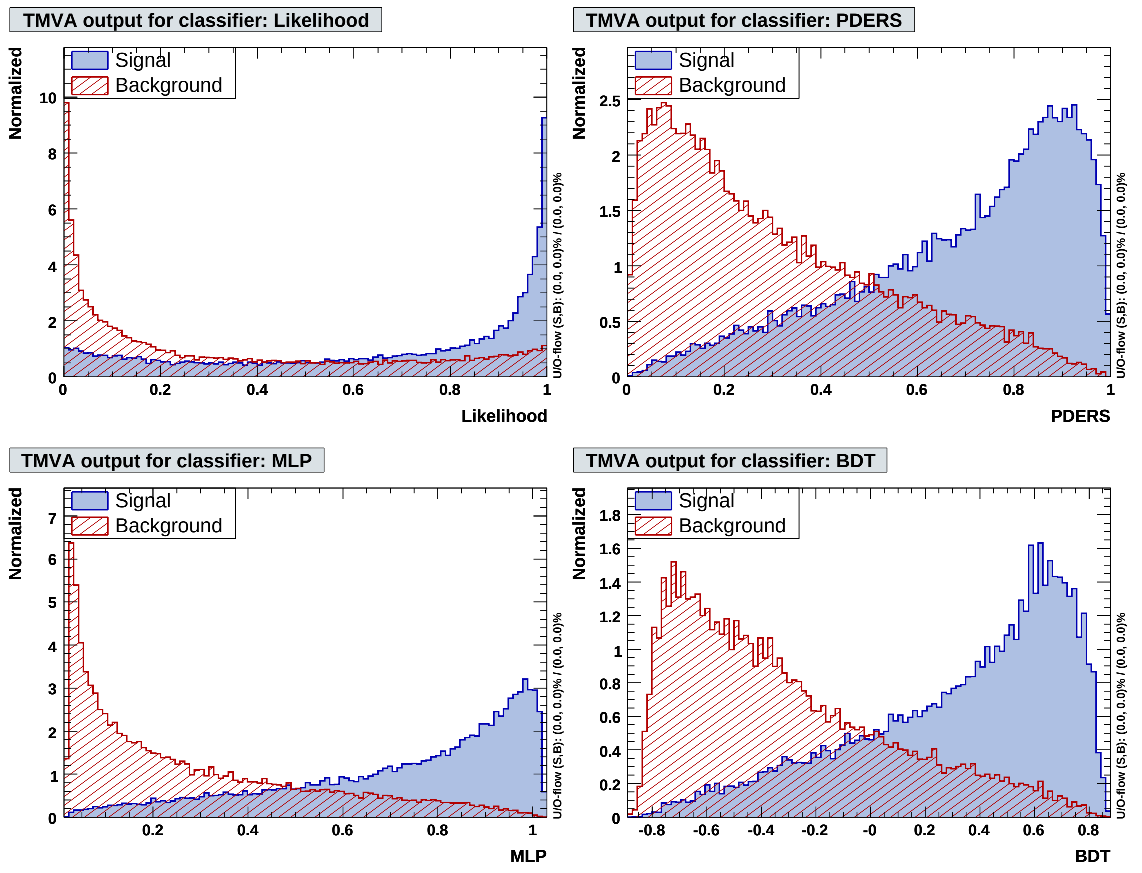

Figure 2: Example plots for classifier output distributions for signal and background events from the academic test sample. Shown are likelihood (upper left), PDE range search (upper right), MLP (lower left) and boosted decision trees.

For standalone use of the trained classifiers, TMVA also generates lightweight C++ response classes, which contain the encoded information from the weight files such that these are not required anymore. These classes do not depend on TMVA or ROOT, neither on any other external library.

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].

4. The Classifiers

All TMVA classifying methods inherit from a common base class, which implements basic functionality common to all classifiers. Options common for all classifiers that can be specified upon booking include the possibility to normalise the input variables, variable transformations as

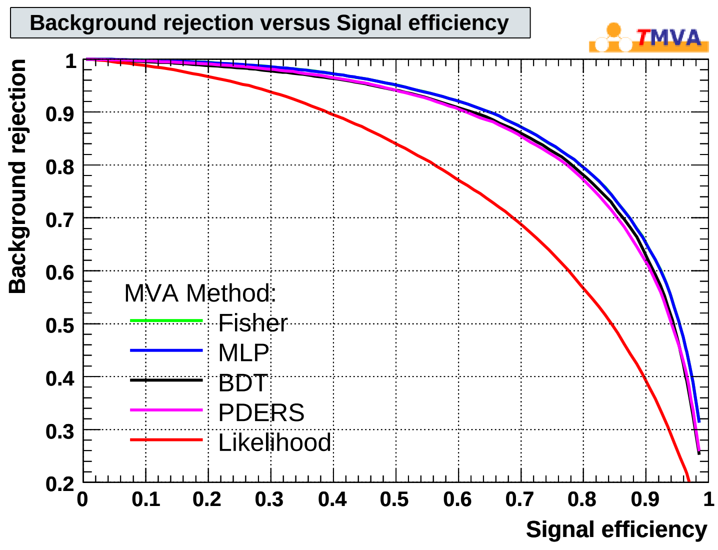

Figure 3: Example for the background rejection versus signal efficiency obtained by cutting on the classifier outputs for the events of the test sample.

preprocessing, the possibility to create probability density functions from the resulting MVA-output distributions and the level of printout generated during application and testing.

4.1 Rectangular Cut optimisation

Although not a real multivariate classifier, the simplest and most common technique for selecting signal events from a mixed sample of signal and background events is the application of an ensemble of rectangular cuts on discriminating variables. Unlike all other classifiers in TMVA, the cut classifier only returns a binary response (signal *or* background).

The optimisation of cuts performed by TMVA maximises the background rejection at given signal efficiency, and scans over the full range of the latter quantity. Dedicated analysis optimisation for which, e.g., the signal *significance* is maximised rather than the background rejection requires the expected signal and background yields to be known before applying the cuts. This is not the case for a multi-purpose discrimination and hence not used by TMVA. However, the cut ensemble leading to maximum significance corresponds to a particular working point on the efficiency curve, and can hence be easily derived after the cut optimisation scan has converged.[3]

TMVA cut optimisation is performed with the use of multivariate parameter fits. Satisfactory results for the fitting are obtained with either Monte Carlo sampling or a Genetic Algorithm, both

[3] Assuming a large enough number of events so that Gaussian statistics is applicable, the significance for a signal is given by $\mathscr{S} = \varepsilon_{\mathscr{S}}\mathscr{N}_{\mathscr{S}}/\sqrt{\varepsilon_{\mathscr{S}}\mathscr{N}_{\mathscr{S}}+\varepsilon_{\mathscr{B}}(\varepsilon_{\mathscr{S}})\mathscr{N}_{\mathscr{S}}}$, where $\varepsilon_{S(B)}$ and $N_{S(B)}$ are the signal and background efficiencies for a cut ensemble and the event yields before applying the cuts, respectively. The background efficiency ε_B is expressed as a function of ε_S using the TMVA evaluation curve obtained form the test data sample. The maximum significance is then found at the root of the derivative

$$\frac{d\mathscr{S}}{d\varepsilon_S} = N_S \frac{2\varepsilon_B(\varepsilon_S)N_B + \varepsilon_S\left(N_S - \frac{d\varepsilon_B(\varepsilon_S)}{d\varepsilon_S}N_B\right)}{2\left(\varepsilon_S N_S + \varepsilon_B(\varepsilon_S)N_B\right)^{3/2}} = 0\,, \tag{4.1}$$

which depends on the problem.

implemented in TMVA.

The training events are sorted in *binary trees* prior to the optimisation, which significantly reduces the computing time required to determine the number of events passing a given cut ensemble.

4.2 Projective likelihood estimation

The maximum likelihood classification is based on building a model of one dimensional probability density functions (PDFs) from the training data that reproduces the input variables for signal and background. For a given event, the likelihood for being of signal type is obtained by multiplying the signal probability densities of all input variables, and normalising this by the sum of the signal and background likelihoods. Correlations among the variables are ignored.

The probability density functions are constructed either by histogramming the variable distributions in the training data with subsequent spline smoothing or using an unbinned kernel density estimator.

4.3 Multi-dimensional likelihood estimation (k-NN and PDE range-search)

These are generalisations of the projective likelihood classifier to n_{var} dimensions, where n_{var} is the number of input variables used. If the multidimensional PDF for signal and background were known, the multi-dimensional likelihood would exploit the full information contained in the input variables, and would hence be optimal. In practice however, huge training samples are necessary to sufficiently populate the multidimensional phase space for a good estimate of the underlying PDF.[4] Kernel estimation or event counting methods are typically used to approximate the shape of the PDF for finite training statistics.

The k-nearest neighbour (k-NN) method compares an observed (test) event to reference events from a training data set. It searches for a fixed number of closest events in the training sample. The fraction of signal events in the neighbourhood is then used as a probability density estimator (PDE) for the test event's phase space point. To enhance the sensitivity within the volume, instead of pure counting of the events, a polynomial Kernel estimate which weighs events according to their distance from the test event is implemented as well.

A variant of the k-NN classifier is the *PDE range search*, or *PDERS*, that has been suggested in Ref. [7]. Here the PDE for a given test event is obtained by counting the (normalised) number of signal and background (training) events that occur in a fixed volume around the test event. The classification of the test event may then be conducted on the basis of the majority of the nearest training events. In the TMVA implementation, the n_{var}-dimensional volume that encloses the "vicinity" is user-defined and can also be adaptive to ensure a certain number of events in the volume. Again, a number of kernel functions may be used to weight the reference events according to their distance from the test event. The mulit-dimensional likelihood estimators use sorted binary trees to reduce the computing time for the range search.

4.4 Linear and nonlinear discriminant analysis (H-Matrix, Fisher, FDA)

The linear discriminant analysis generally finds a hyperplane in the variable phase space that (best) separates signal from background. The standard method of Fisher discriminants [8] works

[4] Due to correlations between the input variables, only a sub-space of the full phase space may be populated.

in a linear transformed variable space where projected onto the axis defining the separating hyperplane, the events of different classes are pushed as far as possible away from each other, while events of a same class are confined in a close vicinity. The linearity property of this classifier is reflected in the metric with which "far apart" and "close vicinity" are determined: the covariance matrix of the discriminating variable space.

The H-Matrix method is a simple estimator built on the relative difference of the two multivariate χ^2-values of the event being compatible with either signal (χ^2_S) or background (χ^2_B). The respective $\chi^2_{S(B)}$-values are calculated using the sample means for signal and background and their respective covariance matrices obtained from the training data.

The function discriminant analysis (FDA) lets the user choose any parametrised function as discrimination boundary and FDA fits the parameters to it, requiring the signal (background) function value to be as close as possible to 1 (0). Its advantage over the more involved and automatic nonlinear discriminators is the simplicity and transparency of the discrimination expression. A shortcoming is that FDA will underperform for involved problems with complicated, phase space dependent nonlinear correlations, where it will be difficult to guess a discriminating function for the decision boundary.

4.5 Artificial neural networks

Artificial neural networks (ANNs) are classifiers that feed the weighted input variables of a test event into a set of nodes also called neurons. Each node generates an output as response of the input according to its (non linear) activation function which can either be fed as input into consecutive nodes or used as an output, i.e. the final response of the network. With each node acting as a basis function, an appropriate choice of arrangement and number of nodes and their interconnecting weights in principle allows to approximate any decision boundary. For a given node arrangement (architecture) the training of a neural network consists of finding the interconnecting weights such that the separation between background and signal event is optimised.

There are three different implementations of ANNs in TMVA, all being feed-forward networks. This means that the nodes are arranged in an array with connections only in one direction, i.e. forward, from the input nodes (through the hidden layers) towards the output nodes without cycles or loops.

The CFANN was adapted from a FORTRAN code developed at the Université Blaise Pascal in Clermont-Ferrand and which uses random Monte Carlo sampling for the weight fitting during training. The other two networks both use standard back-propagation during the weight optimisation. One is an interface to the network already previously implemented in ROOT and the other is our own development offering additional flexibility concerning the choice of different activation functions.

4.6 Support Vector Machine

Support Vector Machines are non-linear discrimination algorithms following an idea from [9, 10] to separate signal and background by a simple linear hyperplane in a non-linear transformed variable space. The hyperplane is completely defined by the events that are closest to the separating plane. These events are also called the *support vectors*. The actual variable space transformation allowing for the linear hyperplane separation of signal and background is never actually specified or calculated but replaced by the choice of so called kernel functions that define a (non linear) metric in the variable space. Polynomial, Sigmoidal and Gaussian kernel functions are available in TMVA. Their parameters (i.e. the with of the Gaussian) are to be chosen in order to adapt to the size and structure of the non-linear features that need to be captured in the decision boundary of the data.

4.7 Boosted/bagged decision trees

A decision tree is a classifier that is structured as a binary tree. For each test event, repeated left/right (yes/no) decisions are performed on a single variable at a time until the event reaches a so called leaf node which classifies it as being either signal or background. The collection of leaf nodes split the phase space into many disjunct regions that are classified as being either signal or background like. The tree structure is defined during the training (tree building) phase, when starting from the whole training sample, consecutive binary splits are determined using the variable and cut value that allows maximum separation between signal and background at the time. When the splitting is finished, the node is classified as either signal or background depending on the majority of training events that end up in it.

In TMVA, the stop criteria for the splitting during the training phase is given by the minimum number of events which is demanded for a leaf node. Small numbers of events in leaf nodes are able to capture small features in the phase space discriminating signal from background. However this may easily result in over-training, i.e. the capture of statistial fluctuations in the training sample rather than genuine features of the underlying PDFs. A pruning algorithm with adjustable prune strength is applied after the tree building to remove statistically insignificant nodes from the decision tree.

Boosted decision trees represent an extension to a single decision tree. The classification of individuals of an ensemble of decision trees are combined to form a classifier which is given by a (weighted) majority vote of the classificaion from the individual decision trees. The individual trees are derived from the same training sample by reweighting the events. In TMVA, the standard AdaBoost [11] algorithm is implemented, which calculates the boost weight used in the next tree building depending on the number of misclassified training events in the previously trained tree.

Rather than performing boosting in order to create an ensemble of decision trees, bagging as defined in [12] uses randomly drawn events taken from the original training sample with replacement to construct different training samples. An ensemble of decision trees is then constructed from the collection of training samples derived by this re-sampling. A variant of this idea implemented in TMVA uses random event weights to create different training samples from which the decision trees in the ensemble are constructed.

Both bagging and boosting stabilise the response of the decision trees with respect to fluctuations in the training sample.

4.8 Predictive learning via rule ensembles

This classifier is a TMVA implementation of Friedman-Popscus' RuleFit method described in [13]. Its idea is to use an ensemble of so-called *rules* to create a scoring function with good classification power. Each rule is defined by a simple sequence of cuts which either selects signal or background. The easiest way to create an ensemble of rules is to extract it from an ensemble of decision trees where every node in a tree (except the root node) corresponds to a sequence of cuts required to reach the node from the root node, and can be regarded as a rule. The rules give a response value of 1 if an events satisfies the respective cuts and 0 otherwise and are regarded as basis function for the final classifier. The latter is constructed as linear combinations of the rules in the ensemble resulting in a scoring function as response of the RuleFit classifier.

The coefficients (rule weights) of the linear combination that maximise the separation power between signal and background events are calculated using a regularised minimisation procedure [14].

4.9 Classifier Discussion

There is obviously no general answer to the question which classifier should be used. To guide the user, we have attempted an assessment of various relevant classifier properties in Table 1. Simplicity is a virtue, but only if it is not at the expense of discrimination power. Robustness with respect to overtraining could become an issue when the training sample is scarce. Some methods require more attention than others in this regard. For example, boosted decision trees are particularly vulnerable to overtraining if used without care. To circumvent overtraining a problem-specific adjustment of the pruning strength parameter is required.

To assess whether a linear discriminant analysis (LDA) could be sufficient for a classification problem, the user is advised to analyse the correlations among the discriminating variables by inspecting scatter and profile plots (it is not enough to print the correlation coefficients, which by definition are linear only). Using an LDA greatly reduces the number of parameters to be adjusted and hence allow smaller training samples. It usually is robust with respect to generalisation to larger data samples. For intermediate problems, the function discriminant analysis (FDA) with some selected nonlinearity may be found sufficient. It is always useful to cross-check its performance against several of the sophisticated nonlinear classifiers to see how much can be gained over the use of the simple and very transparent FDA.

For problems that require a high degree of optimisation and allow to form a large number of input variables, complex nonlinear methods like neural networks, the support vector machine, boosted decision trees and/or RuleFit are more appropriate.

Very involved multi-dimensional variable correlations with strong nonlinearities are usually best mapped by the multidimensional probability density estimators such as PDERS and k-NN.

For RuleFit we emphasize that the TMVA implementation differs from Friedman-Popescu's original code [13], with (yet) slightly better robustness and out-of-the-box performance for the latter version. In particular, the behaviour of the original code with respect to nonlinear correlations and the curse of dimensionality would have merited two stars.[5] We also point out that the excellent performance for by majority linearly correlated input variables is achieved somewhat artificially by adding a Fisher-like term to the RuleFit classifier (this is true for both implementations).

[5]An interface to Friedman-Popescu's original code has now been implemented in TMVA.

		Classifiers									
Criteria		Cuts	Likeli-hood	PDE-RS	k-NN	H-Matrix	Fisher	ANN	BDT	Rule-Fit	SVM
Perfor-mance	No or linear correlations	⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
	Nonlinear correlations	○	○	⋆⋆	⋆⋆	○	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆
Speed	Training	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆	○	⋆	○
	Response	⋆⋆	⋆⋆	○	⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
Robust-ness	Overtraining	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	○	⋆	⋆⋆
	Weak variables	⋆⋆	⋆	○	○	⋆⋆	⋆⋆	⋆	⋆⋆	⋆	⋆
Curse of dimensionality		○	⋆⋆	○	○	⋆⋆	⋆⋆	⋆	⋆	⋆	
Transparency		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

Table 1: Assessment of classifier properties. The symbols stand for the attributes "good" (⋆⋆), "fair" (⋆) and "bad" (○). "Curse of dimensionality" refers to the "burden" of required increase in training statistics and processing time when adding more input variables. See also comments in text. The FDA classifier is not represented here since its properties depend on the chosen function.

5. Conclusion

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

Source code and library of TMVA-v.3.5.0 and higher versions are part of the standard ROOT distribution kit (v5.14 and higher). The newest TMVA development version can be downloaded from Sourceforge.net at http://tmva.sf.net.

Acknowledgements

The fast growth of TMVA would not have been possible without the contribution and feedback from many developers mentionned as co-authors of the users guide [1]) as well as René Brun and the ROOT team. 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.).

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] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/.

[3] http://tmva.sourceforge.net

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

[5] The following web pages give information on available statistical tools in HEP and other areas of science: https://plone4.fnal.gov:4430/P0/phystat/, http://astrostatistics.psu.edu/statcodes/.

[6] 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).*

[7] T. Carli and B. Koblitz, Nucl. Instrum. Meth. **A501**, *576 (2003) [hep-ex/0211019].*

[8] R.A. Fisher, Annals Eugenics **7**, *179 (1936).*

[9] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method*", Automation and Remote Control, 24, 774 (1963).*

[10] V. Vapnik and A. Chervonenkis, "A note on one class of perceptrons*", Automation and Remote Control, 25 (1964).*

[11] Y. Freund and R.E. Schapire, J. of Computer and System Science **55**, *119 (1997).*

[12] L. Breiman, J. Friedman, R. Olshen and C. Stone, "Classification and Regression Trees*", Wadsworth (1984).*

[13] J. Friedman and B.E. Popescu, "Predictive Learning via Rule Ensembles*", Technical Report, Statistics Department, Stanford University, 2004.*

[14] J. Friedman and B.E. Popescu, "Gradient Directed Regularization for Linear Regression and Classification*", Technical Report, Statistics Department, Stanford University, 2003.*