Multivariate Data Analysis with TMVA4

Jan Therhaag(*) (University of Bonn)
ICCMSE 2009 29.09.09

(*) On behalf of the present core developer team: A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss

On the web: http://tmva.sourceforge.net/
Classification and Regression Tasks in HEP

Classification of signal/background:
How to find the best decision boundary?

Regression:
How to determine the correct model and its parameters?
Classification and Regression Tasks in HEP

Classification of signal/background: How to find the best decision boundary?

Regression: How to determine the correct model and its parameters?

Can a machine learn that for you? Yes, with TMVA!
What is TMVA?

- TMVA provides methods for multivariate analysis…

…based on supervised learning…
What is TMVA?

- TMVA provides methods for multivariate analysis...
  Multivariate classifiers/regressors condense (correlated) multi-variable input information into scalar output variables
  ...based on supervised learning...
  Supervised learning means learning by example: the program extracts patterns from training data
What is TMVA?

- **TMVA** provides methods for **multivariate analysis**…

Multivariate classifiers/regressors condense (correlated) multi-variable input information into scalar output variables

…based on **supervised learning**…

Supervised learning means learning by example: the program extracts patterns from training data

…and so much more:

- a common interface for all MVA techniques
- a common interface for classification and regression
- easy training and testing of all methods on the same datasets
  - consistent evaluation and comparison
  - common data preprocessing
- a complete user analysis framework and examples
- embedding in ROOT
- creation of standalone C++ classes (ROOT independent)
- an **understandable** Users Guide
Overview

- TMVA Classifiers & Metaclassifiers
- The TMVA Workflow
The TMVA Classifiers

Cuts, Fisher, Likelihood, Functional Discriminant, kNN, Neural Networks, Support Vector Machine, Boosted Decision Trees, Rule Fit…
Linear Discriminant (Fisher)

- Fisher’s discriminant is a linear model which projects the data on the (hyper)plane of best separation

- Advantages:
  - easy to understand
  - robust, fast

- Disadvantages:
  - not very flexible
  - poor performance in complex settings

<table>
<thead>
<tr>
<th>Performance</th>
<th>Speed</th>
<th>Robustness</th>
<th>Curse of Dim.</th>
<th>Transparency</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>No/linear correlations</td>
<td>Nonlinear correlations</td>
<td>Training</td>
<td>Response</td>
<td>Overtraining</td>
<td>Weak input vars</td>
</tr>
<tr>
<td>😊</td>
<td>😕</td>
<td>😊</td>
<td>😊</td>
<td>😞</td>
<td>😊</td>
</tr>
</tbody>
</table>
K-Nearest-Neighbour / Multidimensional PDE

- Idea: Estimate the multidimensional probability densities by counting the events of each class in a predefined or adaptive volume

- Advantages:
  - reconstruction of PDFs possible

- Disadvantages:
  - „curse of dimensionality“

<table>
<thead>
<tr>
<th>Performance</th>
<th>Speed</th>
<th>Robustness</th>
<th>Curse of Dim.</th>
<th>Transparency</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>No/linear correlations</td>
<td>Nonlinear correlations</td>
<td>Training</td>
<td>Response</td>
<td>Overtraining</td>
<td>Weak input vars</td>
</tr>
<tr>
<td>🌡️</td>
<td>🌡️</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
</tbody>
</table>
Artificial Neural Networks

- Modelling of arbitrary nonlinear functions as a nonlinear combination of simple „neuron activation functions“

- **Advantages:**
  - very flexible, no assumption about the function necessary

- **Disadvantages:**
  - „black box“
  - needs tuning
  - seed dependent

---

### Artificial Neural Networks

#### Feed-forward Multilayer Perceptron

![Feed-forward Multilayer Perceptron Diagram](image)

- **Performance**
  - No/linear correlations
  - Nonlinear correlations

- **Speed**
  - Training
  - Response

- **Robustness**
  - Overtraining
  - Weak input vars

- **Curse of Dim.**
  - Weak input vars

- **Transparency**
  - 1D

- **Regression**
  - multi D

---

Rhodos, 29.09.09

J. Therhaag — Multivariate Data Analysis with TMVA4
Boosted Decision Trees (AdaBoost, Bagging, GradBoost)

- A decision tree is a set of binary splits of the input space
- Grow a forest of decision trees and determine the event class/target by majority vote

- Advantages:
  - ignores weak variables
  - works out of the box

- Disadvantages:
  - vulnerable to overtraining

---

<table>
<thead>
<tr>
<th>Performance</th>
<th>Speed</th>
<th>Robustness</th>
<th>Curse of Dim.</th>
<th>Transparency</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>No/linear correlations</td>
<td>Nonlinear correlations</td>
<td>Training</td>
<td>Response</td>
<td>Overtraining</td>
<td>Weak input vars</td>
</tr>
<tr>
<td>😊</td>
<td>😊</td>
<td>😞</td>
<td>😞</td>
<td>🙁</td>
<td>😊</td>
</tr>
</tbody>
</table>
Meta classifiers – Category Classifier and Boosting

• The category classifier is custom-made for HEP
  – Use different classifiers for different phase space regions and combine them into a single output

• TMVA supports boosting for all classifiers
  – Use a collection of “weak learners“ to improve their performance (boosted Fisher, boosted neural nets with few neurons each…)
Using TMVA
The TMVA Workflow – Training, Testing, Application

• Training:
  – Classification:
    Learn the features of the different event classes from a sample with known signal/background composition
  – Regression:
    Learn the functional dependence between input variables and targets

• Testing:
  – Evaluate the performance of the trained classifier/regressor on an independent test sample
  – Compare different methods

• Application:
  – Apply the classifier/regressor to real data
Selection of variables

- TMVA supports...
  - ... ROOT TTree or ASCII files as input
  - ... any combination or function of available input variables (including TMath functions)
  - ... independent signal/bg preselection cuts
  - ....any input variable as individual event weight
  - ... various methods to split data into training/test trees
Data Preprocessing

• Decorrelation may improve the performance of simple classifiers...

• Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

... but complex correlations call for more sophisticated classifiers
Data Preprocessing

• Decorrelation may improve the performance of simple classifiers…

• Note that in cases with non-Gaussian distributions and/or nonlinear correlations, decorrelation may do more harm than any good

… but complex correlations call for more sophisticated classifiers
A complete TMVA training/testing session

```cpp
void TMVAAnalysis() {
    TFile* outputFile = TFile::Open( "TMVA.root", "RECREATE" );

    TMVA::Factory *factory = new TMVA::Factory( "MVAnalysis", outputFile,"!V" );

    TFile *input = TFile::Open("tmva_example.root");

    factory->AddVariable("var1+var2", 'F');
    factory->AddVariable("var1+var2", 'F');  // factory->AddTarget("tarval", 'F');

    factory->AddSignalTree ( (TTree*)input->Get("TreeS"), 1.0 );
    factory->AddBackgroundTree ( (TTree*)input->Get("TreeB"), 1.0 );
    //factory->AddRegressionTree ( (TTree*)input->Get("regTree"), 1.0 );

    factory->BookMethod( TMVA::Types::kLikelihood, "Likelihood", "!V:!TransformOutput:Spline=2:NSmooth=5:NAvEvtPerBin=50" );
    factory->BookMethod( TMVA::Types::kMLP, "MLP", "!V:NCycles=200:HiddenLayers=N+1,N:TestRate=5" );

    factory->TrainAllMethods();  // factory->TrainAllMethodsForRegression();
    factory->TestAllMethods();
    factory->EvaluateAllMethods();

    outputFile->Close();
    delete factory;
}
```
Evaluation – The TMVA GUI

• TMVA is not only a collection of classifiers, but also a complete MVA framework!
  – a collection of evaluation macros can be accessed via the convenient GUI

<table>
<thead>
<tr>
<th>TMVA Plotting Macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1a) Input Variables</td>
</tr>
<tr>
<td>(1b) Decorrelated Input Variables</td>
</tr>
<tr>
<td>(1c) PCA-transformed Input Variables</td>
</tr>
<tr>
<td>(2a) Input Variable Correlations (scatter profiles)</td>
</tr>
<tr>
<td>(2b) Decorrelated Input Variable Correlations (scatter profiles)</td>
</tr>
<tr>
<td>(2c) PCA-transformed Input Variable Correlations (scatter profiles)</td>
</tr>
<tr>
<td>(3) Input Variable Linear Correlation Coefficients</td>
</tr>
<tr>
<td>(4a) Classifier Output Distributions</td>
</tr>
<tr>
<td>(4b) Classifier Output Distributions for Training and Test Samples</td>
</tr>
<tr>
<td>(4c) Classifier Probability Distributions</td>
</tr>
<tr>
<td>(4d) Classifier Rarity Distributions</td>
</tr>
<tr>
<td>(5a) Classifier Cut Efficiencies</td>
</tr>
<tr>
<td>(5b) Classifier Background Rejection vs Signal Efficiency (ROC curve)</td>
</tr>
<tr>
<td>(6) Likelihood Reference Distributions</td>
</tr>
<tr>
<td>(7a) Network Architecture</td>
</tr>
<tr>
<td>(7b) Network Convergence Test</td>
</tr>
<tr>
<td>(8) Decision Trees</td>
</tr>
<tr>
<td>(9) PDFs of Classifiers</td>
</tr>
<tr>
<td>(10) Rule Ensemble Importance Plots</td>
</tr>
<tr>
<td>(11) Quit</td>
</tr>
</tbody>
</table>

Plot all signal (S) and background (B) input variables with and without pre-processing

Correlation scatters and linear coefficients for S & B

Classifier outputs (S & B) for test and training samples (spot overtraining)

Classifier significance with optimal cuts

B rejection versus S efficiency (ROC)

Classifier-specific plots:
• Likelihood reference distributions
• Classifier PDFs (for probability output and Rarity)
• Network architecture, weights and convergence
• Visualization of decision trees
Evaluation of classification

- Check for overtraining, compare classifier performance and select the optimal cut value for every classifier
Evaluation of classification

- Check for overtraining, compare classifier performance and select the optimal cut value for every classifier.
Evaluation of classification

- Check for overtraining, compare classifier performance and select the optimal cut value for every classifier.
Evaluation of regression

- Compare deviation from target for different methods
• Multiclass classification
• Automatic classifier tuning via cross validation
• Advanced method combination
• Bayesian classifiers
Conclusion

- TMVA is a powerful toolkit for both classification and regression in HEP
- All MVA methods can be accessed through a common interface, data preprocessing can be applied
- TMVA supports the user in the MVA selection process via a collection of evaluation scripts
- TMVA is open source software and can be downloaded from http://tmva.sourceforge.net

Acknowledgments: The fast development of TMVA would not have been possible without the contribution and feedback from many developers and users to whom we are indebted. A full list of contributors is available in the TMVA User’s Guide (http://tmva.sourceforge.net).
Backup
Data Preprocessing

- TMVA supports individual transformations for each method
  - transformations can be chained
  - transformations are calculated on the fly when the event is read
  - TMVA supports decorrelation, Gaussianization and normalization
Decorrelation

- Decorrelations can drastically improve the performance of the simple classifiers

- But complex correlations call for more sophisticated classifiers...
Gaussianisation

- Decorrelation may improve by pre-Gaussianisation of the variables
  - First step: obtain uniform distribution through Rarity transformation
    \[ x_k^{\text{flat}}(i_{\text{event}}) = \int_{-\infty}^{x_k(i_{\text{event}})} p_k(x'_k) \, dx'_k, \quad \forall k \in \{\text{variables}\} \]
    - Rarity transform of variable \(k\)
    - Measured value
    - PDF of variable \(k\)
  - Second step: make Gaussian via inverse error function
    \[ x_k^{\text{Gauss}}(i_{\text{event}}) = \sqrt{2} \cdot \text{erf}^{-1}(2x_k^{\text{flat}}(i_{\text{event}}) - 1), \quad \forall k \in \{\text{variables}\} \]
    \[ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt \]
Classifier specific evaluation

average no. of nodes before/after pruning: 4193 / 968