*T*MVA
A Toolkit for (Parallel) MultiVariate Data Analysis

Andreas Höcker (ATLAS), Helge Voss (LHCb), Kai Voss (ex. ATLAS)

Joerg Stelzer (ATLAS), (Peter Speckmayer (ATLAS))

- **apply a (large) variety of sophisticated data selection algorithms to your data sample**
- **have them all trained and tested**
- **choose the best one for your selection problem and “go”**

http://tmva.sourceforge.net/

Outline

- Introduction: MVAs, what / where / why
- the MVA methods available in *T*MVA
- toy examples
- (real experiences)
- Summary/Outlook
- some “look and feel”

Introduction to MVA

- **At the beginning of each physics analysis:**
  - **select your event sample, discriminating against background**
  - **event tagging**
- **Or even earlier:**
  - **find e, π , K .. candidates (← uses Likelihood in classification and RICH pattern recognition)**
  - **remember the 2D cut in log(Pt) vs impact parameter in old L1-Trigger ?**

- **MVA -- MulitVariate Analysis:**
  - **nice name, means nothing but:**

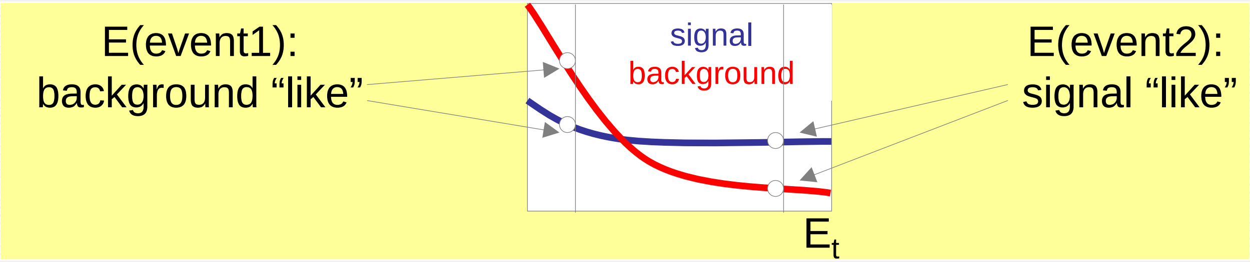
Use several observables from your events to form ONE combined variable and use this in order to discriminate between "signal" or "background"

Introduction to MVAs

- **e.g. sequence of cuts (well that's not really a "combination" into one variable)**
  - **probably still most used and accepted selection method**
  - **<u>advantage:</u> easy to understand and interprete**
  - **<u>disadvantage:</u> often inefficient!! e.g. a very signal like event that has just ONE observable that misses the cut is still rejected.**

several observables ➔ One selection criterium

- **Likelihood selection**
  - **probably the second most used and well accepted selection method**
  - **calculate for each observable in an event a probability that its value belongs to a "signal" or a "background" event using reference distributions for signal and background.**



  - **Then cut on the combined likelihood of all these probabilities**

MVA Experience

- MVAs certainly made it's way through HEP -- e.g. LEP Higgs analyses
- Often people use custom tools, without much comparison
  - MVAs tedious to implement, therefore few true comparisons between methods !
  - most accepted: cuts
  - also widely understood and accepted: likelihood (probability density estimators – PDE)
  - often disliked, but more and more used (LEP, BABAR): Artificial Neural Networks
  - much used in BABAR and Belle: Fisher discriminants
  - introduced by D0 (for electron id): H-Matrix
  - used by MiniBooNE and recently by BABAR: Boosted Decision Trees
- All interesting methods … but how to dissipate the widespread skepticism ?
  - black boxes !
  - how to interpret the selection ?
  - what if the training samples incorrectly describe the data ?
  - how can one evaluate systematics ?
  - very interesting, but which MVA should I use and where to find the time to code it?

MVA Experience

- **All interesting methods … but how to dissipate the widespread skepticism ?**

- black boxes !
- how to interpret the selection?

Certainly, cuts are more transparent, so
- if cuts are competitive (rarely the case) → use them
- in presence of correlations, cuts loose transparency

- what if the training samples incorrectly de-scribe the data ?

Not good, but not necessarily a huge problem:
- performance on real data will be worse than training results
- however: bad training does not create a bias !
- only if the training efficiencies are used in data analysis → bias
- optimized cuts are <u>not</u> in general less vulnerable to systematics (perhaps on the contrary !)

- how can one evaluate systematics ?

There is no principle difference in systematics evaluation between single variables and MVAs
- need control sample for MVA output (not necessarily for each input variable)

- nice and interesting but which MVA and how to code it ?

☺ : just look at *T*MVA !

What is *T*MVA

- Toolkit for Multivariate Analysis (*T*MVA): C++ , ROOT
  - 'parallel' processing / training and evaluation of various MVA techniques to discriminate *signal* from *background* samples.
- *T*MVA presently includes (ranked by complexity):
  - Rectangular cut optimisation (pure random or genetic optimisation algorithm)
  - Correlated likelihood estimator (PDE approach)
  - Multi-dimensional likelihood estimator (PDE - range-search approach)
  - Fisher (and Mahalanobis) discriminant
  - H-Matrix approach (χ^2 estimator)
  - Artificial Neural Network (two different implementations)
  - Boosted Decision Trees
- The *T*MVA analysis provides training, testing and evaluation of the MVAs
- The training results are written to specific weight files
- The weight files are read by dedicated reader class for actual MVA analysis
- *T*MVA supports training of multiple MVAs in (detector) regions as a function of up to two variables (*e.g.*, η, p_T)

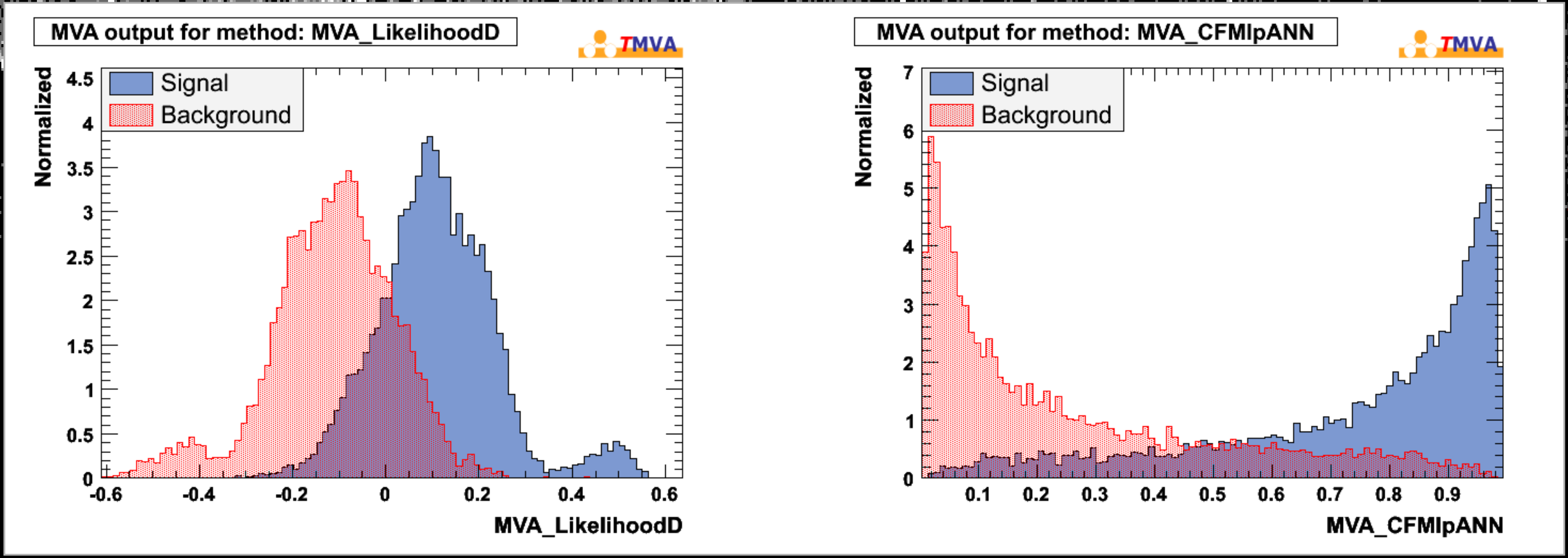
*T*MVA Technicalities

- ***T*MVA is available as:**
  - **sourceforge (SF) package to accommodate world-wide access**
    - **code can be downloaded as *tar* file, or via anonymous cvs access**
    - **home page:** [**http://tmva.sourceforge.net/**](http://tmva.sourceforge.net/)
    - **SF project page:** [**http://sourceforge.net/projects/tmva**](http://sourceforge.net/projects/tmva)
    - **view CVS:** [**http://cvs.sourceforge.net/viewcvs.py/tmva/TMVA/**](http://cvs.sourceforge.net/viewcvs.py/tmva/TMVA/)
    - **mailing lists:** [**http://sourceforge.net/mail/?group_id=152074**](http://sourceforge.net/mail/?group_id=152074)
  - **part of the ROOT package (Development release 5.11/06)**

- ***T*MVA is modular**
  - **training, testing and evaluation factory iterates over all available (and selected) methods**
  - **though the current release is stable, we think that the improvement and extension of the methods is a continuous process**
  - **each method has specific options that can be set by the user for optimisation**
  - **ROOT scripts are provided for all relevant performance analysis**
  - **small "reader" class for simple "application" of the selection**

- ***We enthusiastically welcome new users, testers and developers ☺***

TMVA Methods



Cut Optimisation

- **Simplest method: cut in rectangular volume using N_{var} input variables**

$$x_{\text{cut},i_{\text{event}}} \in (0,1) = \bigcap_{v \in \{\text{variables}\}} \left\{ x_{v,i_{\text{event}}} \subset \left[x_{v,\min}, x_{v,\max} \right] \right\}$$

- **Usually training files in *T*MVA do not contain *realistic* signal and background abundance → cannot optimize for best significance (S/√(S+B))**
  - **scan in signal efficiency [0 →1] and maximise background rejection**

- **Technical problem: how to perform maximisation**
  - **Minuit fit (SIMPLEX/MIGRAD) found to be not reliable enough**
  - **use random sampling or**
  - ***Genetics Algorithm* for maximisation (→ CMS)**

- **Huge speed improvement by sorting training events in N_{var}*-dim. Binary Trees***
  - **for 4 variables: 41 times faster than simple volume cut**

- **Future improvement (not yet in release): cut in de-correlated variable space**

Projected Likelihood Estimator (PDE Appr.)

- **Combine probability density distributions to likelihood estimator**

Likelihood ratio for event i_{event}

PDFs

discriminating variables

$$x_{\text{PDE},i_{\text{event}}} = \frac{\prod_{v\in\{\text{variables}\}} p_v^{\text{signal}}(x_{v,i_{\text{event}}})}{\sum_{S\in\{\text{species}\}} \left(\prod_{v\in\{\text{variables}\}} p_v^{S}(x_{v,i_{\text{event}}}) \right)}$$

Species: signal, background types

- **Assumes uncorrelated input variables**
  - **optimal MVA approach if *true*, since containing *all* the information**
  - **usually *not true* → development of different methods!**

- **Technical problem: how to implement reference PDFs**
  - **3 ways: function fitting, parametric fitting (splines, kernel est.), counting**

difficult to automate

easy to automate, can create artefacts

automatic,unbiased, but suboptimal

"De-correlated" Likelihood Estimator

- Remove linear correlations by rotating variable space of PDEs
- Determine *square-root* C' of correlation matrix C, *i.e.*, $C = C'C'$
  - compute C' by diagonalising C: $D = S^T C S \;\Rightarrow\; C' = S\sqrt{D}S^T$
  - transformation from original (x) in de-correlated variable space (x′) by: $x' = C'^{-1}x$
- Separate transformation for signal and background
- Note that this "de-correlation" is only complete, if:
  - input variables are Gaussians
  - correlations linear only
  - in practise: gain form de-correlation often rather modest

---

- Output of likelihood estimators often strongly peaked at 0, 1 → *T*MVA applies inverse Fermi transformation to facilitate parameterisation:

$$x_{\mathrm{PDE},i_{\mathrm{event}}} \to x'_{\mathrm{PDE},i_{\mathrm{event}}} = -\tau^{-1}\ln\left(x^{-1}_{\mathrm{PDE},i_{\mathrm{event}}} - 1\right)$$

Multidimensional Likelihood Estimator

- **Generalisation of 1D PDE approach to N_{var} dimensions**

- **Optimal method – in theory – since full information is used**

- **Practical challenges:**
  - **parameterisation of multi-dimensional phase space needs <u>huge</u> training samples**
  - **implementation of N_{var}-dim. reference PDF with kernel estimates or counting ("curse of dimensionality")**

- ***T*MVA implementation following *Range-Search* method**
  - **count number of signal and background events in "vicinity" of a data event**
  - **"vicinity" defined by *fixed* or *adaptive* N_{var}-dim. volume size**
  - ***adaptive*: rescale volume size to achieve constant number of reference events**
  - **speed up range search by sorting training events in Binary Trees**

Carli-Koblitz, NIM A501, 576 (2003)

Kernel estimators: Non parametric estimators (no model function used) solely driven by the data

Fisher Discriminant (and H-Matrix)

- **Well-known, simple and elegant MVA method: event selection is performed in a transformed variable space with zero linear correlations, by distinguishing the mean values of the signal and background distributions**

- **Instead of equations, words:**

> An axis is determined in the (correlated) hyperspace of the input variables such that, when projecting the output classes (signal and background) upon this axis, they 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 method is reflected in the metric with which "far apart" and "close vicinity" are determined: the covariance matrix of the discriminant variable space.

  - **optimal for linearly correlated Gaussians with equal RMS' and different means**
  - **no separation if equal means and different RMS (shapes)**

- **Computation of Fisher MVA couldn't be simpler:**

$$x_{\mathrm{Fisher},i_{\mathrm{event}}} \propto \sum_{v\in\{\mathrm{variables}\}} \left\{ x_{v,i_{\mathrm{event}}} \cdot F_v \right\}$$

"Fisher coefficents"

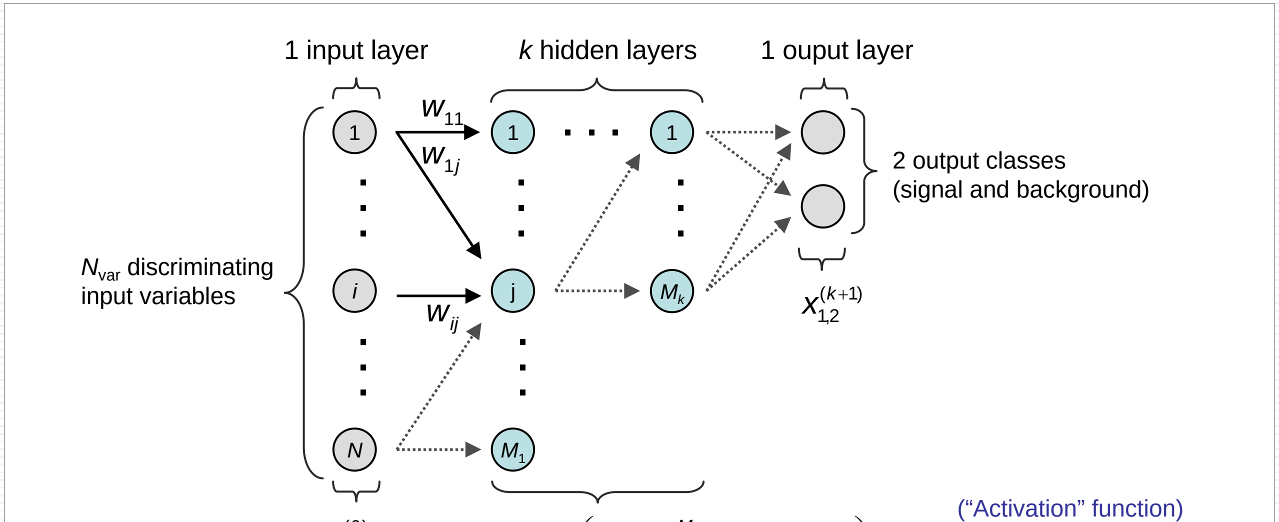
- **H-Matrix estimator: correlated χ^2 : poor man's variation of Fisher discriminant**

Artificial Neural Network (ANN)

- **ANNs are non-linear discriminants: Fisher = ANN without hidden layer**
  - **ANNs are now extensively used in HEP due to their performance and robustness**
  - **they seem to be better adapted to realistic use cases than Fisher and Likelihood**

- ***T*MVA has two different ANN implementations – both are *Multilayer Perceptrons***
  1. **Clermont-Ferrand ANN:** **used for ALEPH Higgs analysis; translated from FORTRAN**
  2. **TMultiLayerPerceptron interface:** **ANN implemented in ROOT**

Feed-forward Multilayer Perceptron



$$x_j^{(k)} = A\left(w_{0j}^{(k)} + \sum_{i=1}^{M_{k-1}} w_{ij}^{(k)} \cdot x_i^{(k-1)}\right)$$

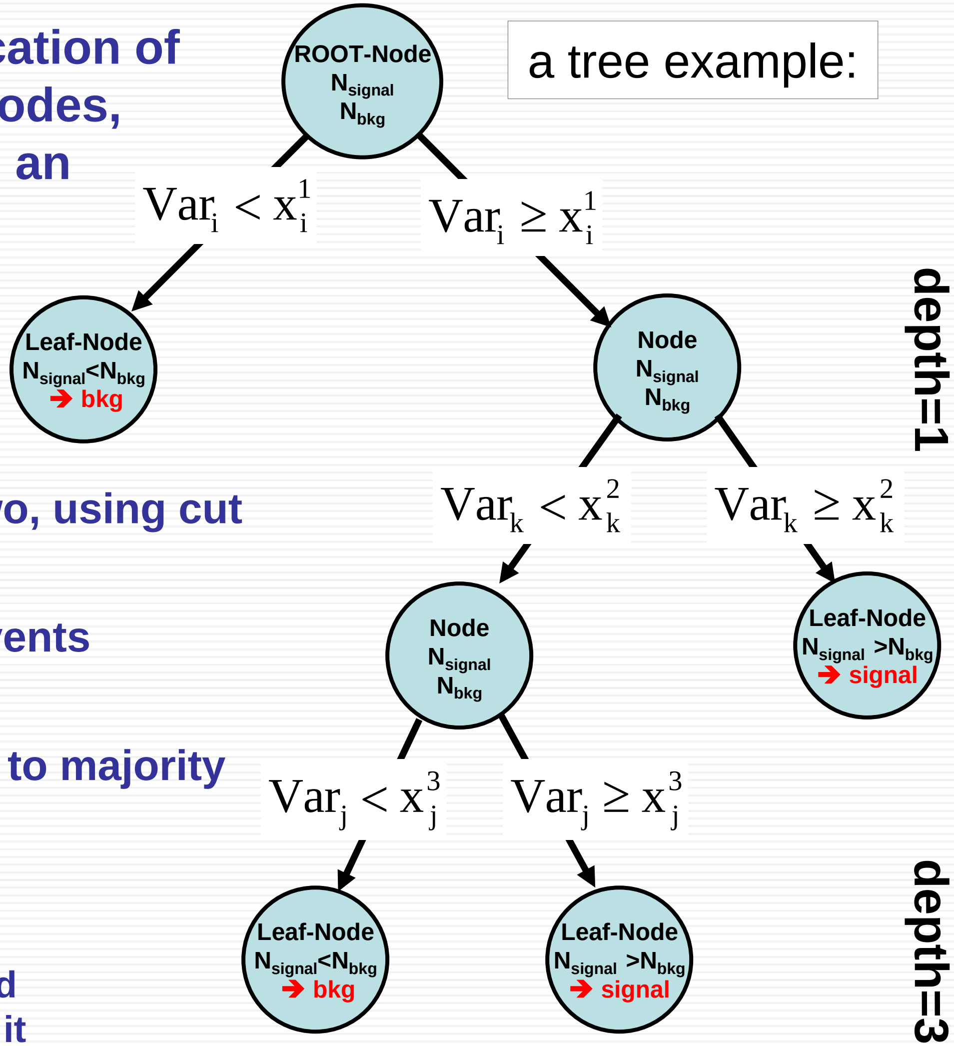
with: $A(x) = \left(1 + e^{-x}\right)^{-1}$

Decision Trees

- **Decision Trees: a sequential application of "cuts" which splits the data into nodes, and the final nodes (leaf) classifies an event as signal or background**

- **Training:**
  - start with the root-node
  - split training sample at node into two, using cut in variable that gives best separation
  - continue splitting until: minimal #events reached, or max purity reached
  - leaf-nodes classifie (S/B) according to majority of events / or give a S/B probablility

- **Testing/Application:**
  - a test event is "filled" at the root-node and classified according to the leaf-node where it ends up after the "cut"-sequence



Boosted Decision Trees

- <u>Decision Trees:</u> used since a long time in general "data-mining" applications, less known in HEP (but very similar to "simple Cuts")
- Advantages:
  - easy to interpret: independently of N_{var}, can always be visualised in a 2D tree
  - independent of monotone variable transformation: rather immune against outliers
  - immune against addition of weak variables
- Disadvatages:
  - instability: small changes in training sample can give large changes in tree structure

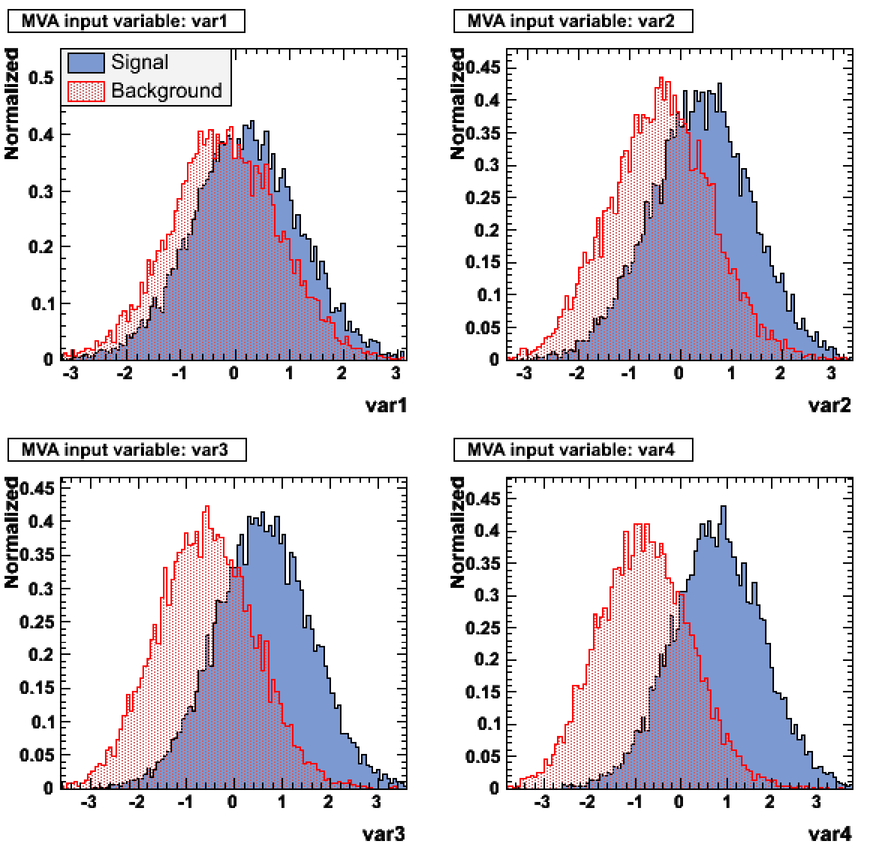
- Boosted Decision Trees (1996): combining several decision trees (forest) derived from one training sample via the application of event weights into ONE mulitvariate event classifier by performing "majority vote"
  - *e.g*. AdaBoost: wrong classified training events are given a larger weight
  - bagging: random weights (re-sampling)

Academic Examples (I)

- **Simple toy to illustrate the strength of the de-correlation technique**
  - **4 linearly corr. Gaussians, with equal RMS and shifted means between S and B**

***T*MVA output :**

Distribution of variables:



Correlation matrix:

--- TMVA_Factory: correlation matrix (signal):

---		var1	var2	var3	var4
---	var1:	+1.000	+0.336	+0.393	+0.447
---	var2:	+0.336	+1.000	+0.613	+0.668
---	var3:	+0.393	+0.613	+1.000	+0.907
---	var4:	+0.447	+0.668	+0.907	+1.000

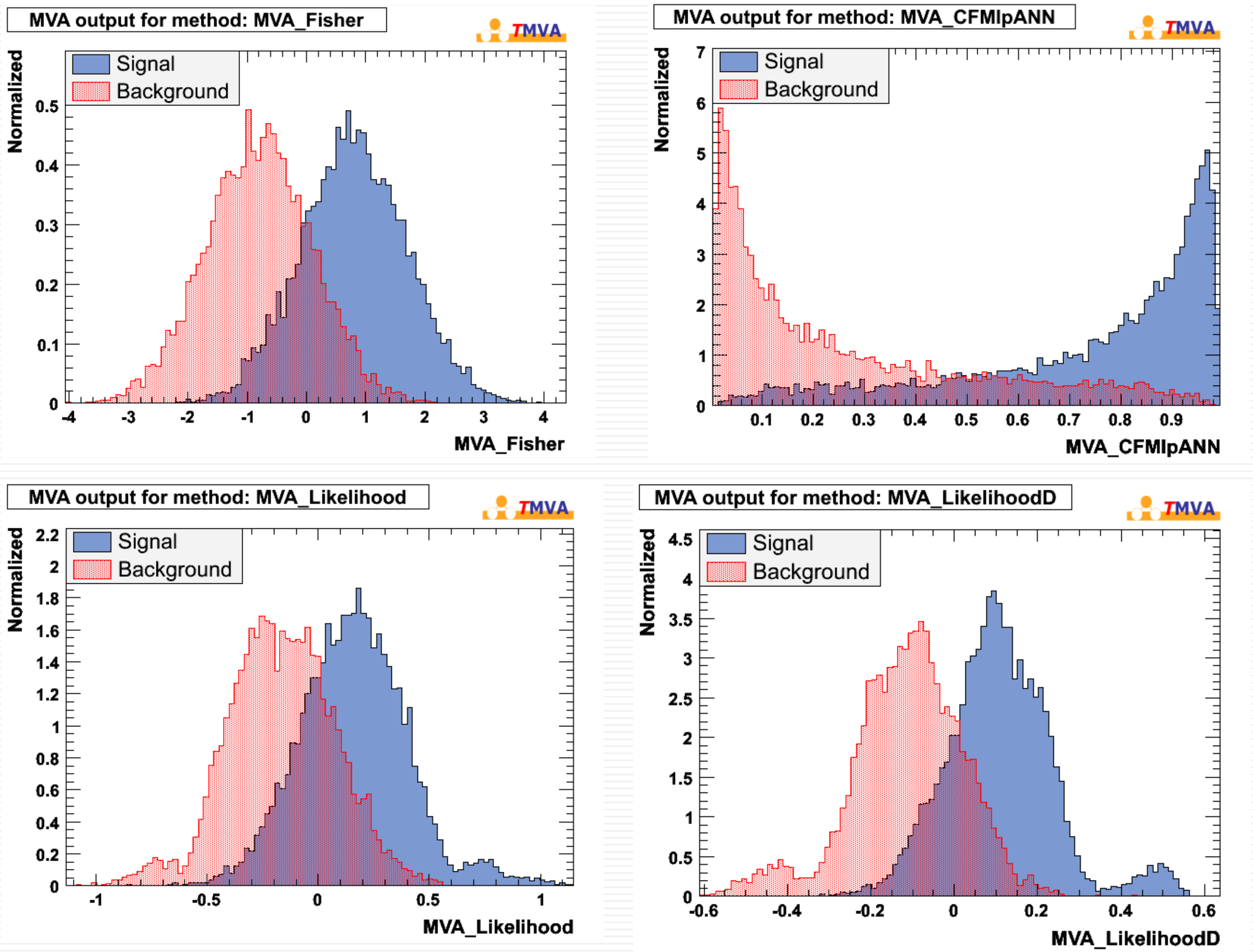
Variable ranking: (currently Fisher only)

--- TMVA_MethodFisher: ranked output (top
--- variable is best ranked)

--- Variable	:	Coefficient:	Discr. power:
--- var4	:	+8.077	0.3888
--- var3	:	–3.417	0.2629
--- var2	:	–0.982	0.1394
--- var1	:	–0.812	0.0391

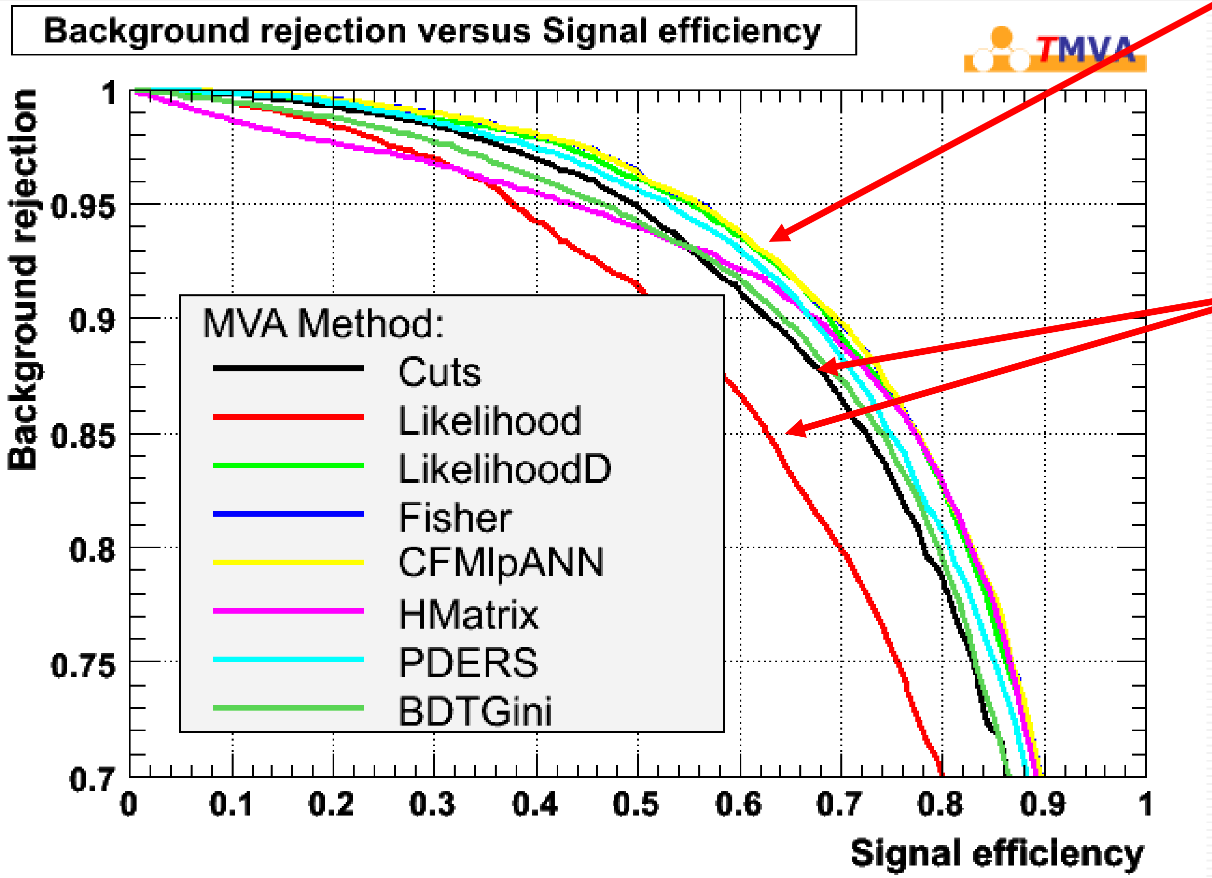
Academic examples (I) …*continued*

distributions for Fisher, (CF)ANN, Likelihood and de-corr. Likelihood



Academic examples (I) …*continued*

… resulting in the following "performance"

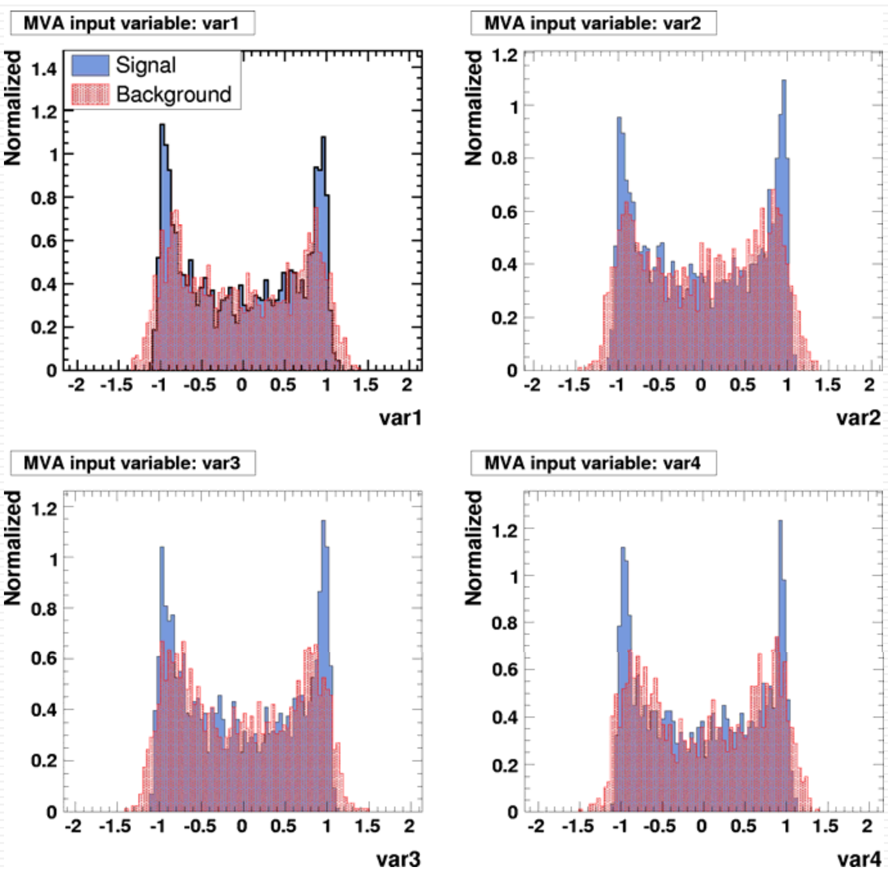


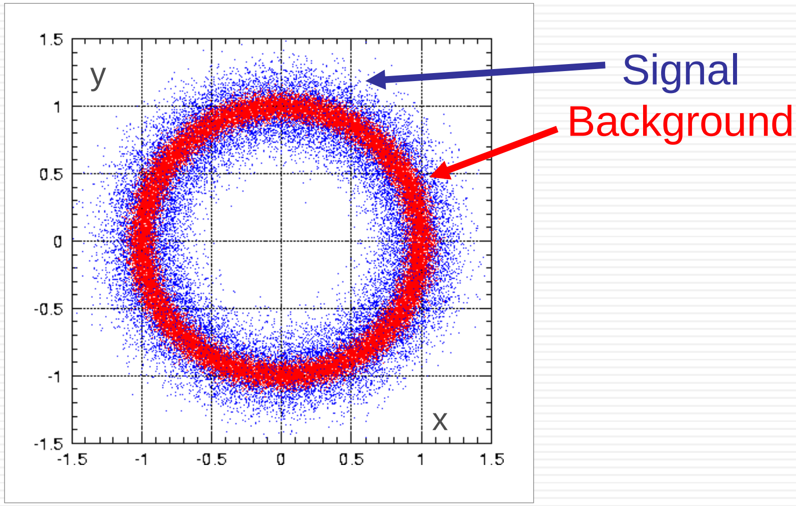
For this case: Fisher discriminant provides the theoretically 'best' possible method

Cuts, Decision Trees and Likelihood w/o de-correlation are inferior

Academic Examples (II)

- **Simple toy to illustrate the shortcomings of the de-correlation technique**
  - **2x2 variables with circular correlations: each set, equal means and different RMS**





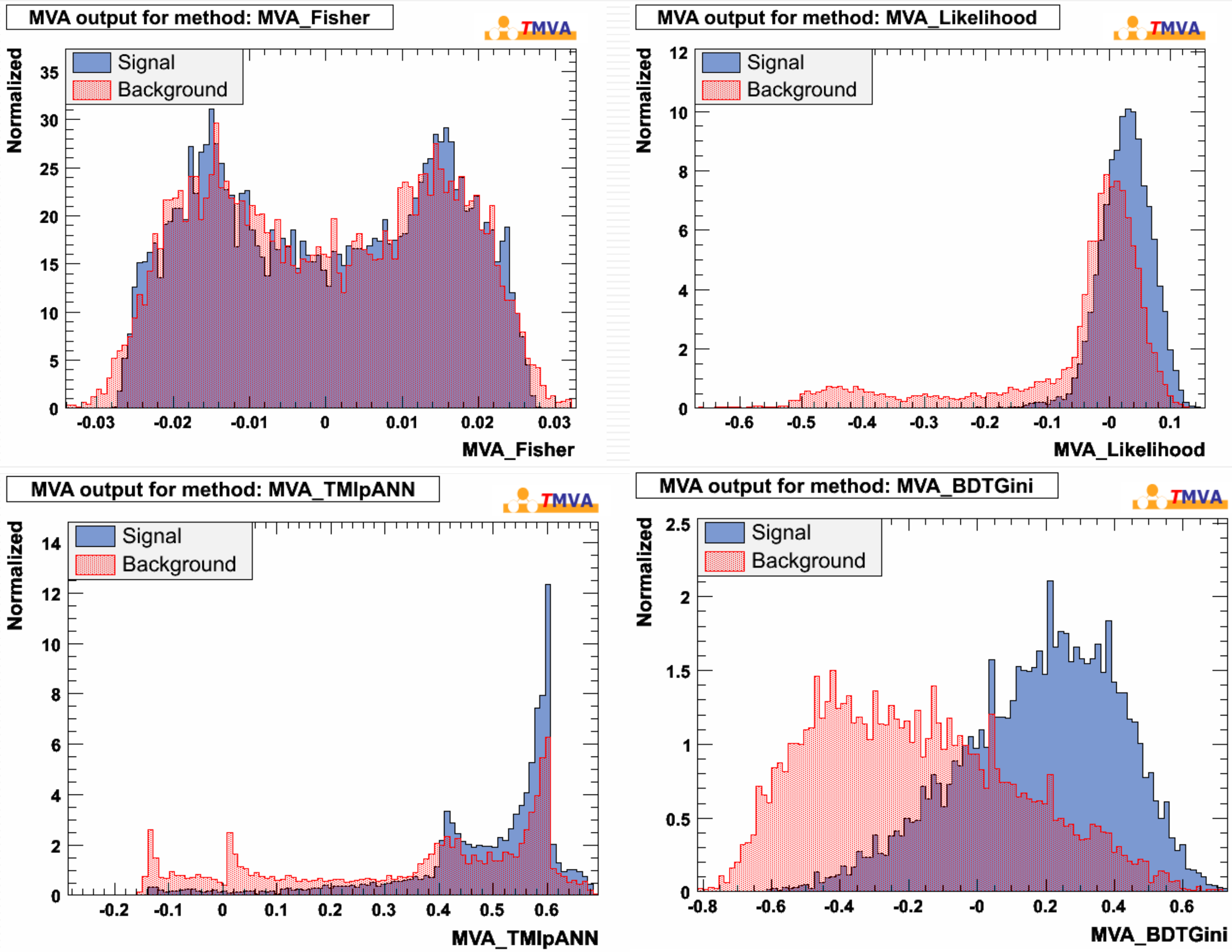
TMVA output :

--- TMVA_Factory: correlation matrix (signal):

---		var1	var2	var3	var4
---	var1:	+1.000	+0.001	–0.004	–0.012
---	var2:	+0.001	+1.000	–0.020	+0.001
---	var3:	–0.004	–0.020	+1.000	+0.012
---	var4:	–0.012	+0.001	+0.012	+1.000

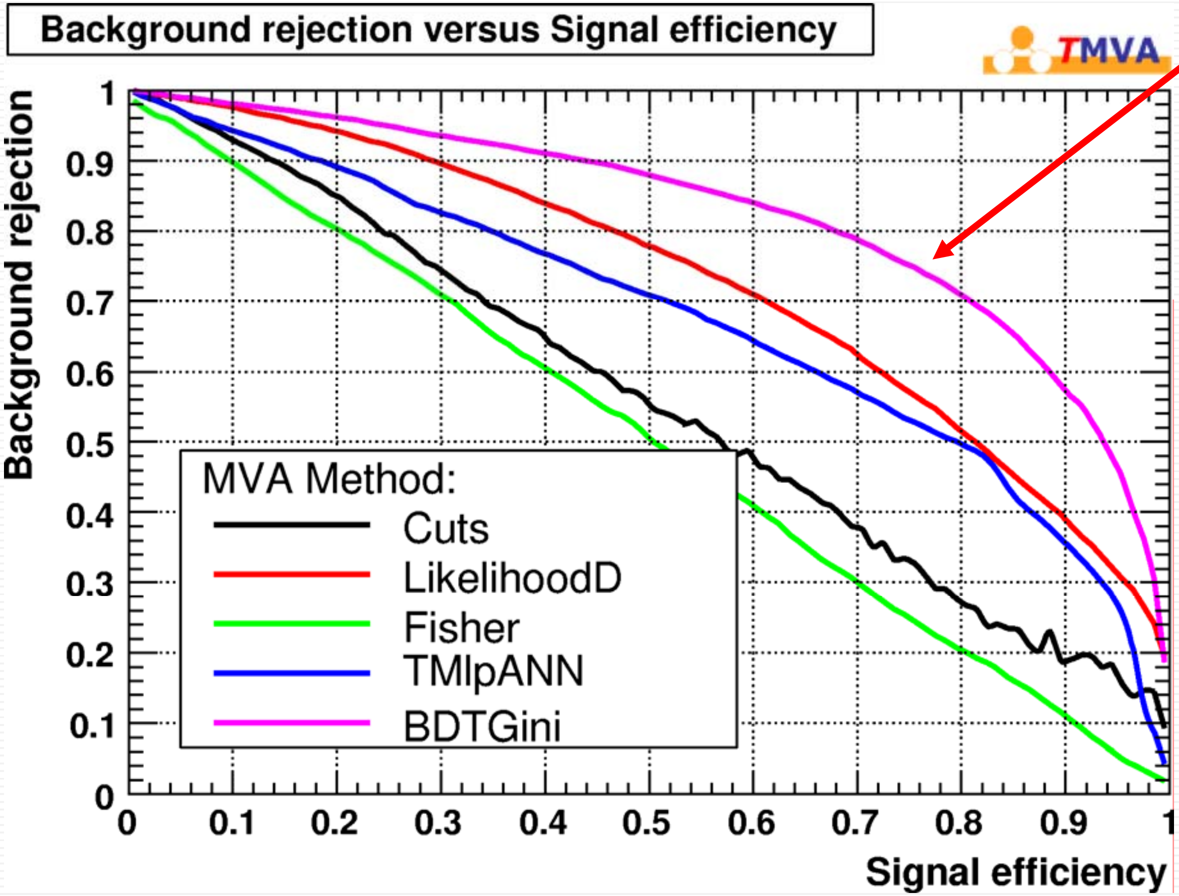
Academic examples (II) …*continued*

MVA output: distributions for Fisher, Likelihood, (ROOT)ANN, Boosted DT



Academic examples (II) …*continued*

… resulting in the following "performance"

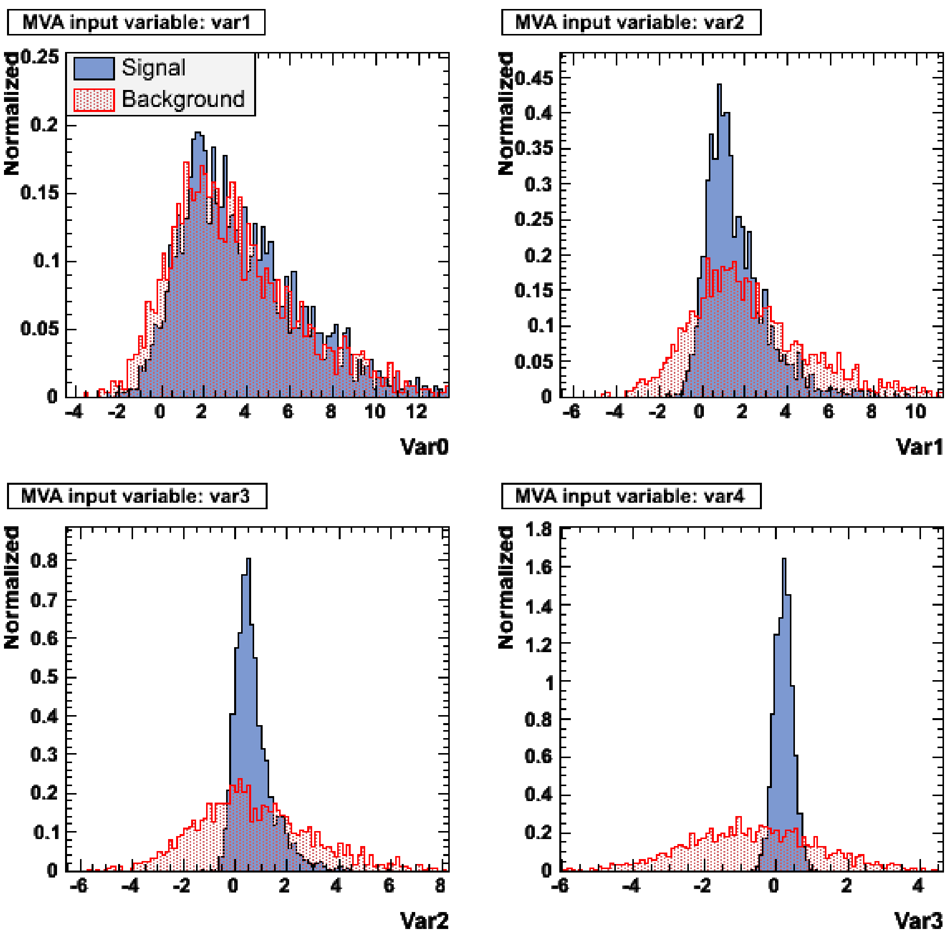


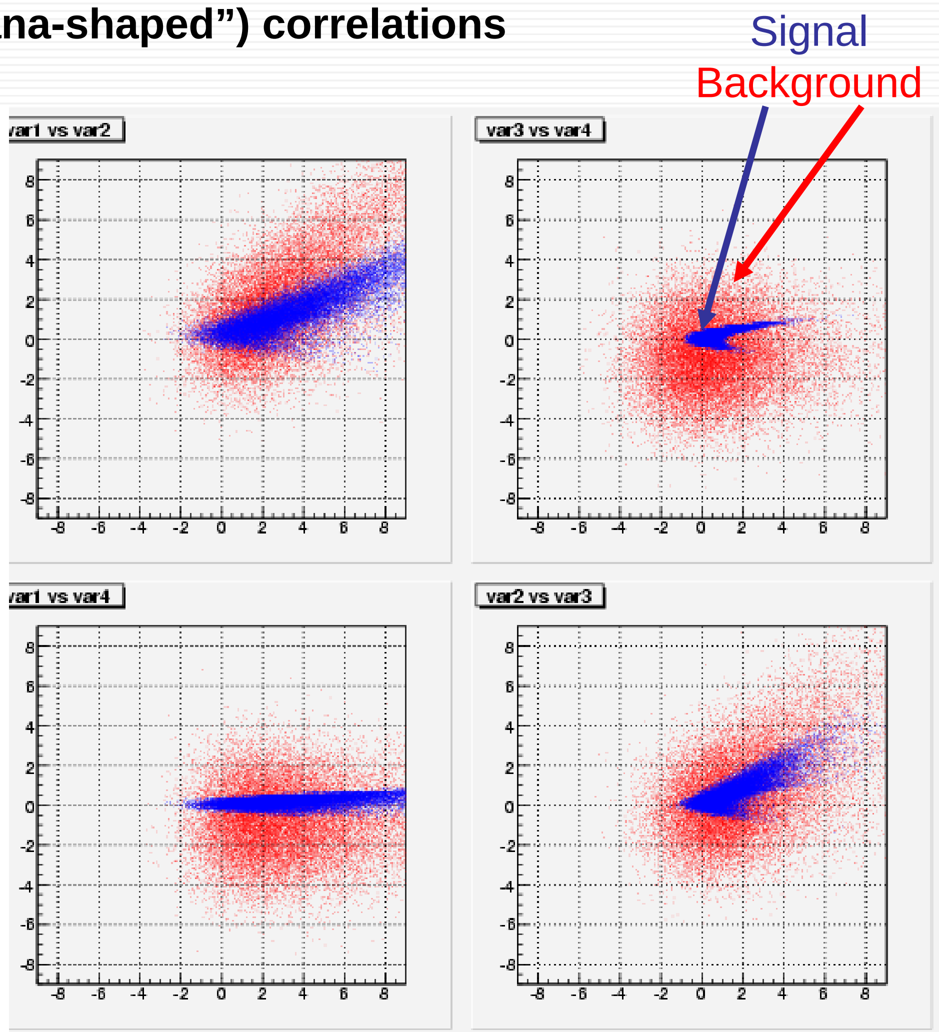
highly nonlinear correlations:

➔ Decision Trees outperform other methods by far

Academic Examples (III)

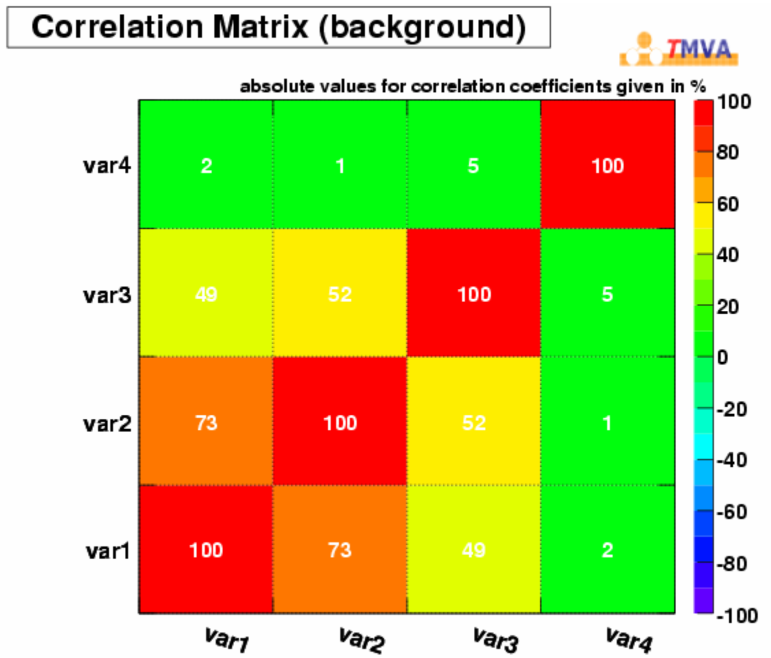
- **Simple toy, perhaps a bit more realistic than the circular correlations:**
  - **2x2 variables with non-linear ("banana-shaped") correlations**





Academic Examples (III) …*continued*

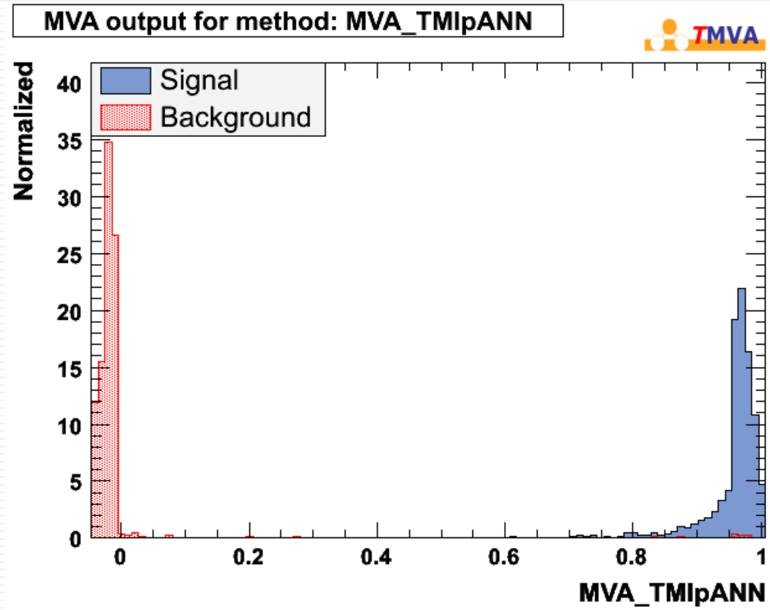
TMVA output :

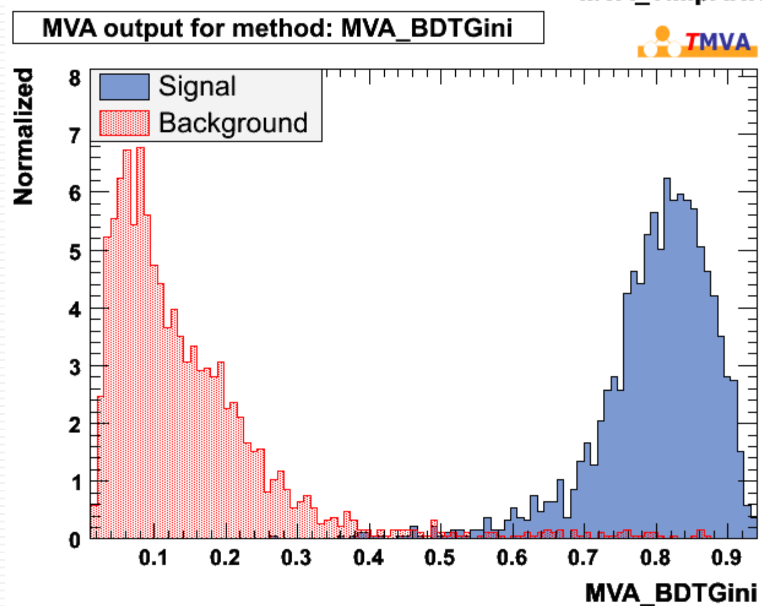


And the same thing in your logfile

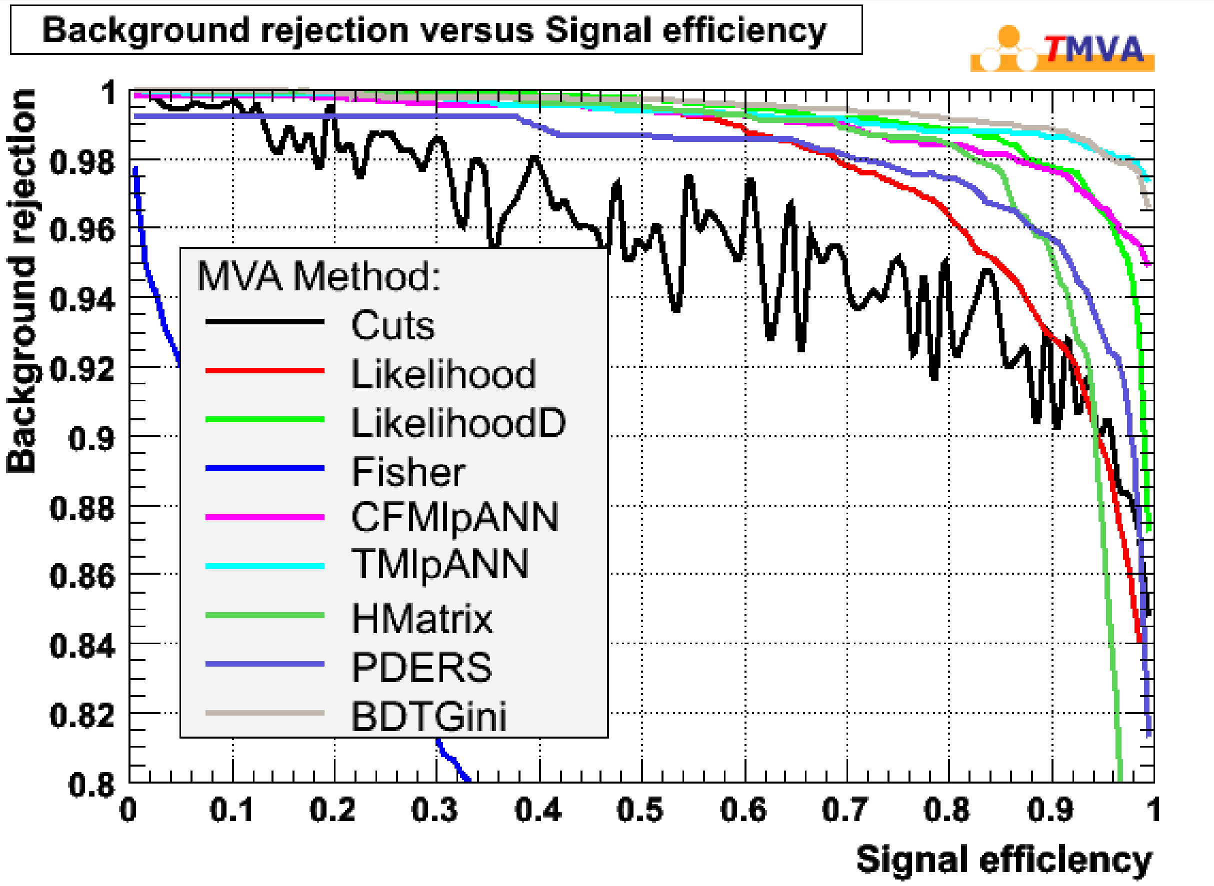
--- TMVA_Factory: correlation matrix (signal):

---		var1	var2	var3	var4
---	var1:	+1.000	+0.733	+0.491	+0.023
---	var2:	+0.733	+1.000	+0.524	+0.019
---	var3:	+0.491	+0.524	+1.000	+0.055
---	var4:	+0.023	+0.019	+0.055	+1.000





Academic Examples (III) …*continued*



also in this case:

"simple methods like Cuts, Fisher discriminant and Likelihood are inferior.

ANNs and Decision Trees perform 'best'

Some "publicity"

Someone from BaBar wrote:

Dear All,
I have been playing with TMVA, the package advertised by Vincent Tisserand in http://babar-hn.slac.stanford.edu:5090/HyperNews/get/physAnal.html to see if it can be useful for our analysis.

.
.
.
.

A few comments:

-The package seems really user-friendly as advertised. I manage to install and run it with no problems.

-Little work is necessary to move from the examples to one's own analysis.

-Spending some time to tailor it to our needs looks to me a very good investment.

-The performance of several different multivariate methods can be evaluated simoultanously! Substituting variables takes 0 work; adding variables, little more than 0.

Concluding Remarks

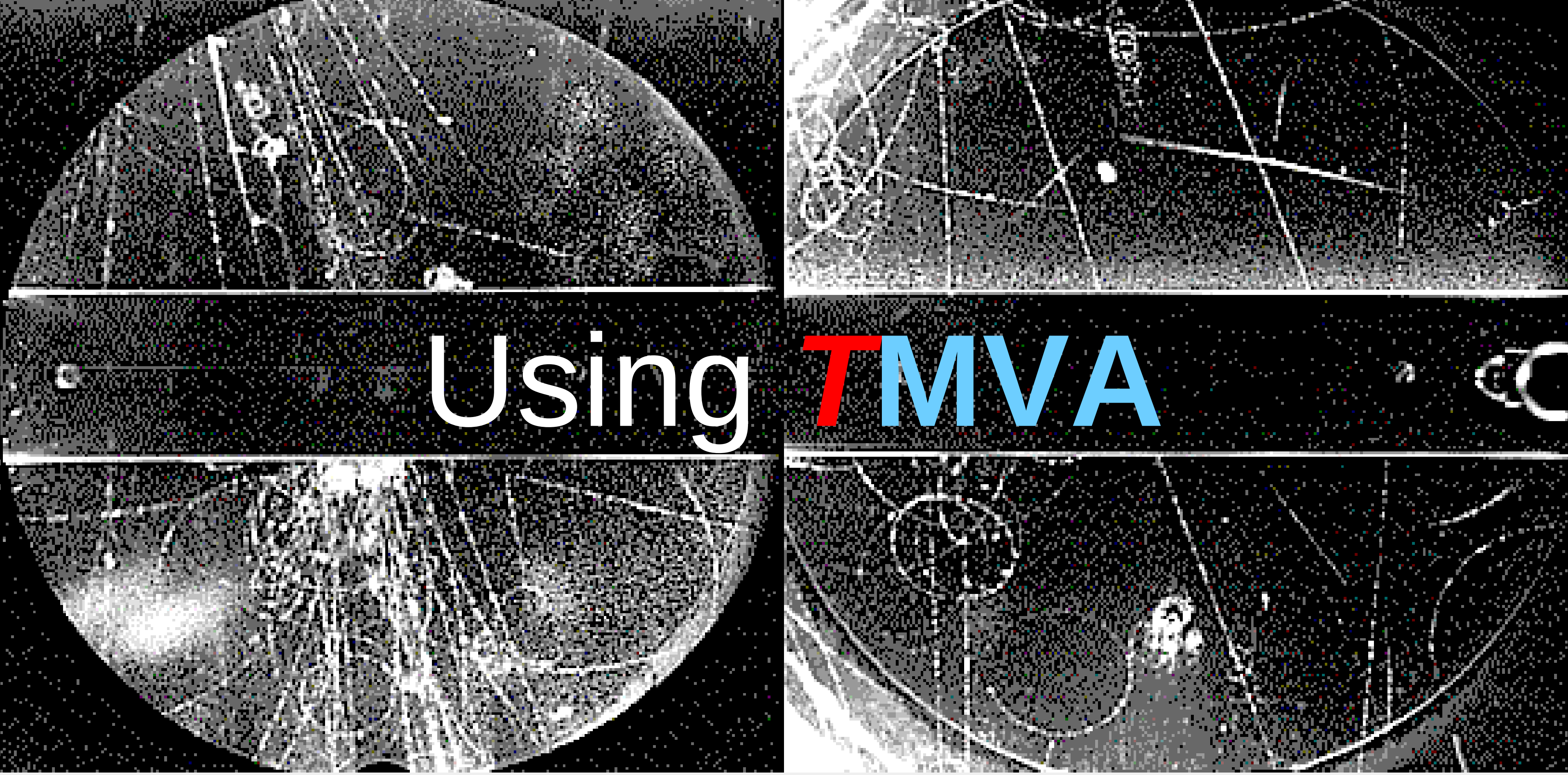
- First stable *T*MVA release available at sourceforge since March 8, 2006
- root integration: TMVA-ROOT package in development release 5.11/06 (May 31)

- *T*MVA provides the training and evaluation tools, but the decision which method is the best is heavily dependent on the use case → train several methods in parallel and see what is best for YOUR analysis
- Most methods can be improved over default by optimising the training options

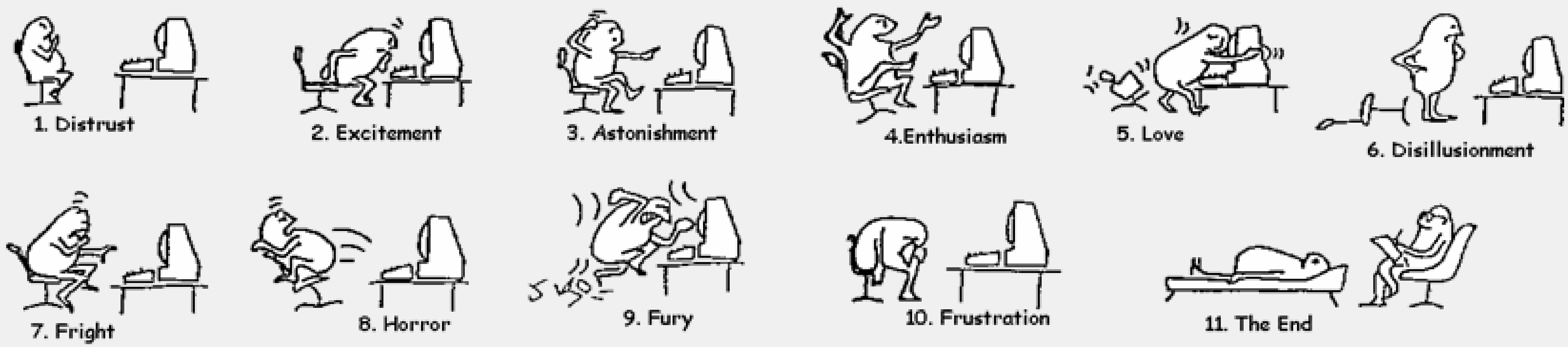
- Tools are developed, but now need to gain realistic experience with them !
  - several ongoing BaBar analyses!!! So how about Belle ?? LHCb ??

Outlook

- **Implementation of “RuleFit”**
- **Implementation of our “own” Neural Network**
- **provide possibility of using event weights in all methods**
- **include ranking of variables (Decision Trees)**



Using TMVA


1. Distrust
2. Excitement
3. Astonishment
4.Enthusiasm
5. Love
6. Disillusionment
7. Fright
8. Horror
9. Fury
10. Frustration
11. The End

Curious? Want to have a look at *T*MVA

<u>**Nothing easier than that:**</u>

- **Get ROOT Development version 5.11/06 (e.g. binaries)**
- **Get from ROOT version 5.11/06 also the source files**
  - **go to $ROOTSYS/tmva**
  - **start: root TMVAnalysis.C**

(toy samples are provided, look at the results/plots and then how you can implement your stuff in the code)

Web Presentation on Sourceforge.net

http://tmva.sourceforge.net/

TMVA toolkit for parallel multivariate data analysis

Quickstart Plots Project page Viewcvs Mailing lists Download

Two ways to download TMVA source code:

1. get tgz file by clicking on green button, or directly from here:
   http://sourceforge.net/project/showfiles.php?group_id=152074&package_id=168420&release_id=397782

1. via anonymous cvs:
   cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/tmva co -P TMVA

- H-Matrix (chi-squared) estimator
- Artificial Neural Network (two different implementations)
- Boosted Decision Trees

The TMVA package includes an implementation for each of these discrimination techniques, their training and testing (performance evaluation). In addition all these methods can be tested in parallel, and hence their performance on a particular data set may easily be compared. For the comparison of the efficiency and background rejection of all the different methods, the analysis job gives some tabulated efficiency / background values in the log file, as well as the efficiencye vs. background rejection curves along with

TMVA – Directory Structure

`src/`	the sources for the TMVA library
`lib/`	here you'll find the TMVA library once it is compiled (copy it to you prefered library directory or include this directory in your LD_LIBRARY_PATH as it is done by: source setup.(c)sh
`examples/`	example code of how to use the TMVA library, using input data from a Toy Monte Carlo
`examples/data`	the Toy Monte Carlo
`reader/`	here you find a single file (TMVA_Reader) which contains all the functionality to "apply" the multivariate analysis which had been trained before. Here you simply read the weight files created during the training, and apply the selection to your data set WITHOUT using the whole TMVA library. An example code is given in TMVApplication.cpp
`macros/`	handy root macros which read and display the results produced e.g. from the "examples/TMAnalysis"
`development/`	similar than what you find in examples, but this is our working and testing directory... have a look if you want to get some idea of how to use the TMVA library

*T*MVA – Compiling and Running

How to compile and run the code:

```
/home> cd TMVA
/home/TMVA> source setup.sh (or setup.csh) // include TMVA/lib in path
/home/TMVA> cd src
/home/TMVA/src> make // compile & build the library ../libTMVA.so
/home/TMVA/src> cd ../examples
/home/TMVA/examples> make
/home/TMVA/examples> TMVAnalysis "MyOutput.root" // run the code
/home/TMVA/examples> root ../macros/efficiencies.C\(\"MyOutput.root\"\)
                    (the cryptic way to give "command line arguments" to ROOT)

or:
/home/TMVA/examples> root -l
root [0] .L ../macros/efficiencies.C
root [1] efficiencies("MyOutput.root")
```

*T*MVA – Training and Testing

Code example for training and testing (TMVAnalysis.cpp): (1)

Create the factory

```
int main( int argc, char** argv )
{
  // ---- create the root output file
  TFile* target = TFile::Open( "TMVA.root", "RECREATE" );

  // create the factory object
  TMVA_Factory *factory = new TMVA_Factory( "TMVAnalysis", target, "" );

...
```

*T*MVA – Training and Testing

Code example for training and testing (TMVAnalysis.cpp): (2)

Read training and testing files, and define MVA variables

```
// load input trees (use toy MC sample with 4 variables from ascii file)
// alternatively: sig and bkg ROOT-Tree (separated or mixed)
factory->SetInputTrees("toy_sig.dat", "toy_bkg.dat")

// this is the variable vector, defining what's used in the MVA
vector<TString>* inputVars = new vector<TString>;
inputVars->push_back("var1");
inputVars->push_back("var2");
inputVars->push_back("var3");
inputVars->push_back("var4");

factory->SetInputVariables( inputVars );
```

*T*MVA – Training and Testing

Code example for training and testing (TMVAnalysis.cpp): (3)

Book MVA methods

Training options: specific for each method

```
factory->BookMethod( "MethodCuts",         “MC:1000000:AllFSmart" );
factory->BookMethod( "MethodLikelihood",   "Spline2:3"      );
factory->BookMethod( "MethodLikelihood",   "Spline2:10:25:D");
factory->BookMethod( "MethodFisher",       "Fisher" );
factory->BookMethod( "MethodCFMlpANN",     "5000:N:N"  );
factory->BookMethod( "MethodTMlpANN",      "200:N+1:N"  );
factory->BookMethod( "MethodHMatrix" );
factory->BookMethod( "MethodPDERS",        "Adaptive:50:100:50:0.99" );
factory->BookMethod( "MethodBDT",          "200:AdaBoost:GiniIndex:10:0:20" );
```

TMVA – Training and Testing

Code example for training and testing (TMVAnalysis.cpp): (4)

Training and testing

```
factory->TrainAllMethods(); // train all MVA methods
factory->TestAllMethods();  // test all MVA methods

// performance evaluation
factory->EvaluateAllVariables(); // for each input variable used in MVAs
factory->EvaluateAllMethods();   // for all MVAs

// close output file and cleanup
target->Close();
delete factory;
}
```

```
#include "TMVA_reader.h"
using TMVApp::TMVA_Reader;
Void MyAnalysis() {

vector<string> inputVars;
 inputVars.push_back( "var1" );
 inputVars.push_back( "var2" );
 inputVars.push_back( "var3" );
 inputVars.push_back( "var4" );

TMVA_Reader *tmva = new TMVA_Reader( inputVars );

tmva->BookMVA(TMVA_Reader::Fisher, "TMVAnalysis_Fisher.weights");

vector<double> varValues;
varValues.push_back( var1 );
varValues.push_back( var2 );
varValues.push_back( var3 );
varValues.push_back( var4 );

double mvaFi  = tmva->EvaluateMVA( varValues, TMVA_Reader::Fisher   );
delete tmva;}
```

[1] Include the reader class

[2] Create an array of the input variables names (here 4var)

[3] Create the reader class

[4] Read the weights and build the MV tool

[5] Create an array with the input variables values

[6] Compute the value of the MV, it is the value you will cut on