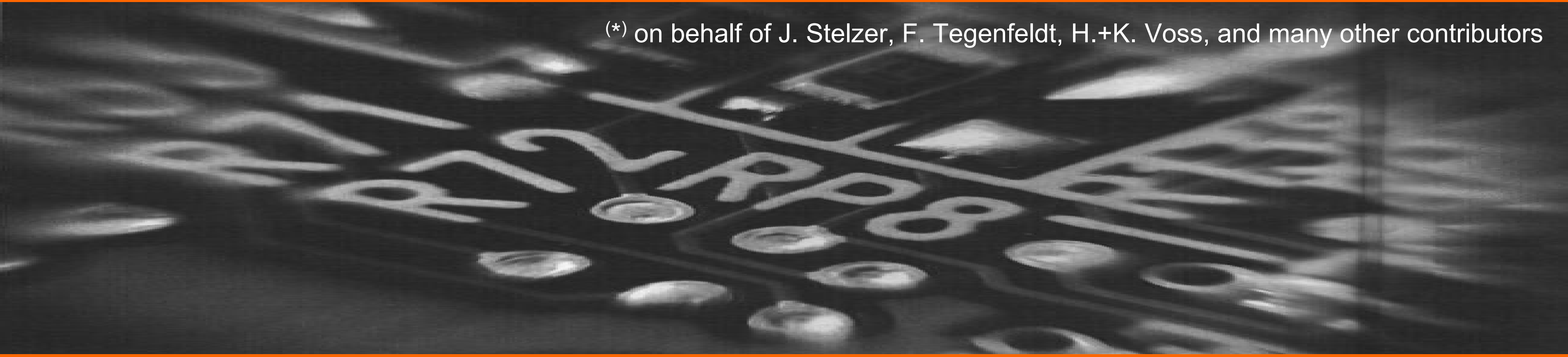
HEP Data Mining with *T*MVA

— ToolKit for Multivariate Analysis with ROOT —

Andreas Hoecker[*] (CERN)

Seminar, IFJ – Krakow, Feb 27, 2007

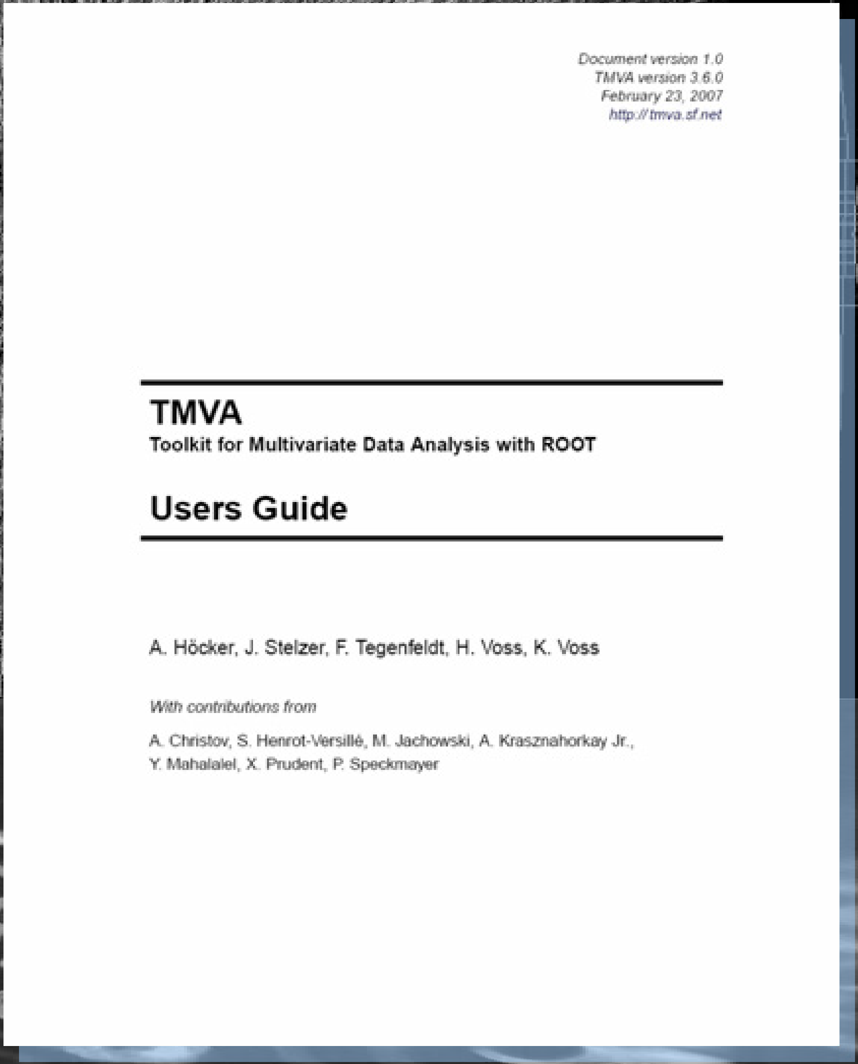


[*] on behalf of J. Stelzer, F. Tegenfeldt, H.+K. Voss, and many other contributors

a d v e r t i s e m e n t

We (finally) have a Users Guide !

Please check on tmva.sf.net for its imminent release



Document version 1.0
TMVA version 3.6.0
February 23, 2007
http://tmva.sf.net

TMVA
Toolkit for Multivariate Data Analysis with ROOT

Users Guide

A. Höcker, J. Stelzer, F. Tegenfeldt, H. Voss, K. Voss

With contributions from

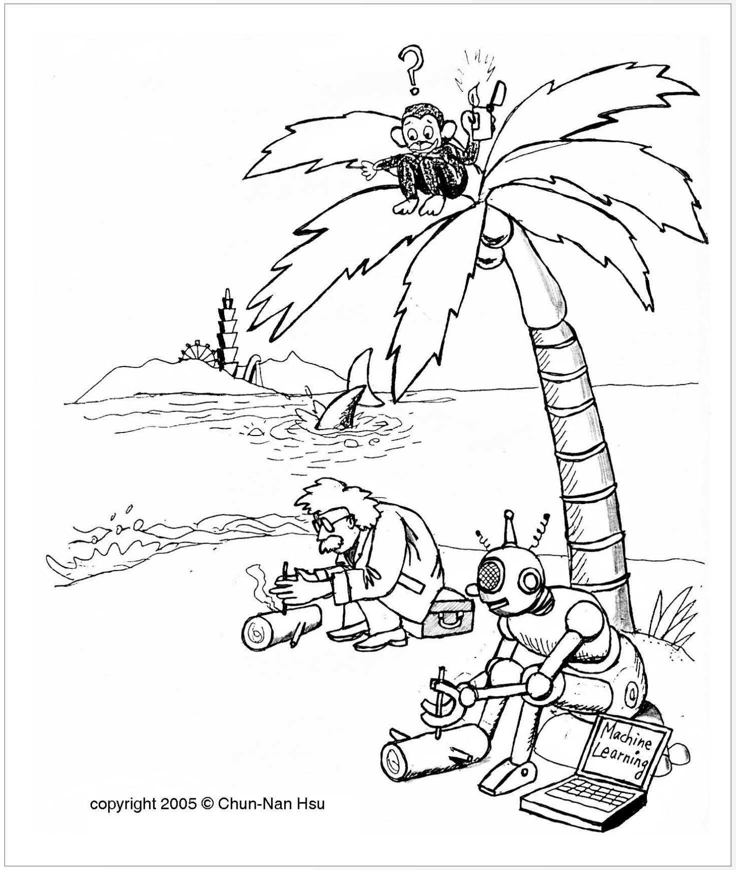
A. Christov, S. Henrot-Versillé, M. Jachowski, A. Krasznahorkay Jr., Y. Mahalalel, X. Prudent, P. Speckmayer

TMVA Users Guide
68pp, incl. code examples
to be submitted to arXiv:physics

Preliminary Remarks

- One example for “Machine Learning”:
  - It is not so useful to let the machine learn what we know well, but let it do where we’re bad:

Human	Set the goal (define “signal” and “background”) Determine discriminating variables Find appropriate control samples Develop machine learning algorithms
Machine	Train the algorithm Independently evaluate the algorithm Apply the algorithm to unknown data

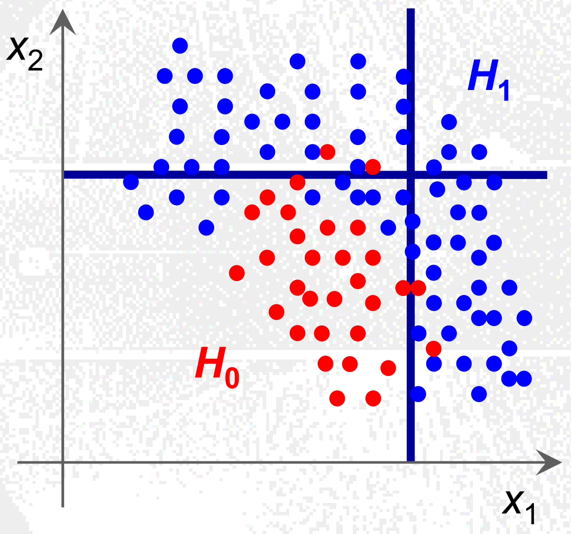


- It is good to understand what the machine does – but it is not mandatory !

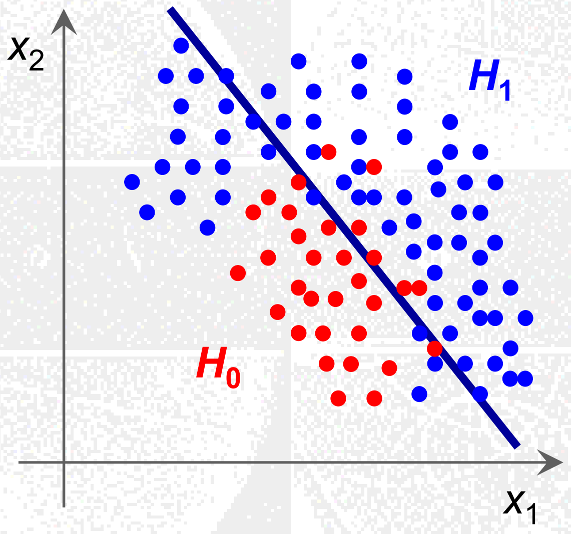
Event Classification

- Suppose data sample with two types of events: H_0, H_1
  - We have found discriminating input variables x_1, x_2, …
  - What decision boundary should we use to select events of type H_1 ?

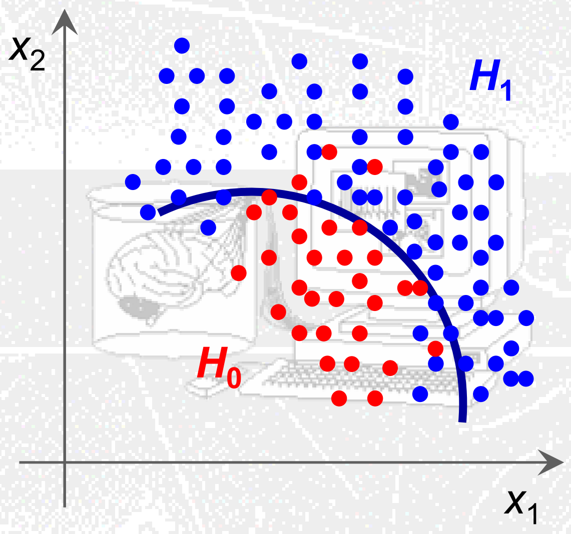
Rectangular cuts?



A linear boundary?



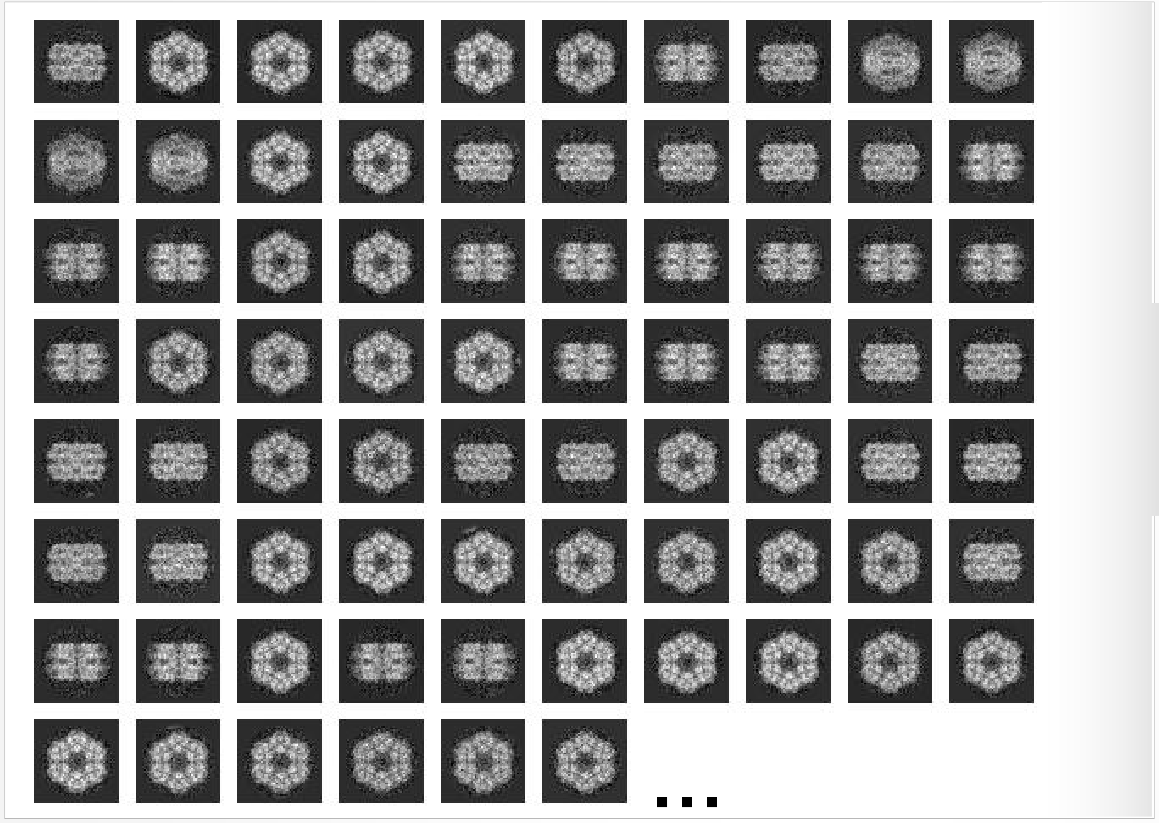
A nonlinear one?



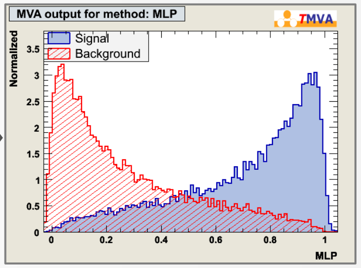
- How can we decide this in an optimal way ? → Let the machine learn it !

Multivariate Event Classification

- All multivariate classifiers have in common to condense (correlated) multi-variable input information in a single scalar output variable
  - It is a $R^n \rightarrow R$ regression problem; classification is in fact a *discretised regression*



$y(H_0) \rightarrow 0,\ y(H_1) \rightarrow 1$



Event Classification in High-Energy Physics (HEP)

- Most HEP analyses require discrimination of signal from background:
  - Event level (Higgs searches, …)
  - Cone level (Tau-vs-jet reconstruction, …)
  - Track level (particle identification, …)
  - Lifetime and flavour tagging (*b*-tagging, …)
  - Parameter estimation (*CP* violation in *B* system, …)
  - etc.
- The multivariate input information used for this has various sources
  - Kinematic variables (masses, momenta, decay angles, …)
  - Event properties (jet/lepton multiplicity, sum of charges, …)
  - Event shape (sphericity, Fox-Wolfram moments, …)
  - Detector response (silicon hits, *dE*/*dx*, Cherenkov angle, shower profiles, muon hits, …)
  - etc.
- Traditionally few powerful input variables were combined; new methods allow to use up to 100 and more variables w/o loss of classification power

Multivariate Classification Algorithms

- A large variety of multivariate classifiers (MVAs) exists

Traditional

Rectangular cuts (optimisation often "by hand")

Projective likelihood (up to 2D)

Linear discriminant analysis (χ^2 estimators, Fisher)

Nonlinear discriminant analysis (Neural nets)

Variants

Prior decorrelation of input variables (input to cuts and likelihood)

Principal component analysis of input variables

Multidimensional likelihood (kernel nearest neighbor methods)

New

Decision trees with boosting and bagging, Random forests

Rule-based learning machines

Support vector machines

Bayesian neural nets, and more general *Committee* classifiers

Multivariate Classification Algorithms

- How to dissipate (often diffuse) skepticism against the use of MVAs

➧ black boxes !

Certainly, cuts are transparent, so
- if cuts are competitive (rarely the case) → use them
- in presence of correlations, cuts loose transparency
- "we should stop calling MVAs black boxes and understand how they behave"

➧ 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 (on the contrary !)

➧ how can one evaluate systematics ?

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

TMVA

What is *T*MVA

- The various classifiers have very different properties
  - Ideally, all should be tested for a given problem
  - Systematically choose the best performing classifier
  - Comparisons between classifiers improves the understanding and takes away mysticism

- *T*MVA — Toolkit for multivariate data analysis with ROOT
  - Framework for *parallel* **training**, **testing**, **evaluation** and **application** of MV classifiers
  - A large number of linear, nonlinear, likelihood and rule-based classifiers implemented
  - Each classifier provides a ranking of the input variables
  - The input variables can be decorrelated or projected upon their principal components
  - Training results and full configuration are written to weight files and applied by a **Reader**
  - Clear and simple user interface

TMVA Development and Distribution

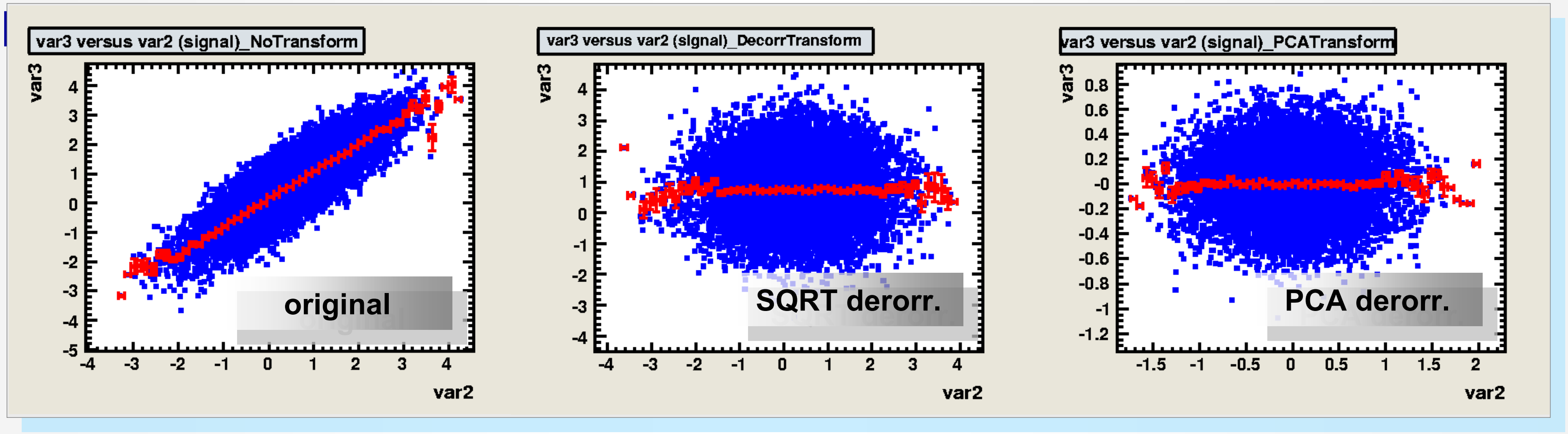
- TMVA is a sourceforge (SF) package for world-wide access
  - Home page ……………….http://tmva.sf.net/
  - SF project page …………. http://sf.net/projects/tmva
  - View CVS …………………http://tmva.cvs.sf.net/tmva/TMVA/
  - Mailing list ………………..http://sf.net/mail/?group_id=152074
  - ROOT class index ……….http://root.cern.ch/root/htmldoc/TMVA_Index.html
- Very active project → fast response time on feature requests
  - Currently 4 main developers, and 24 registered contributors at SF
  - ~ 700 downloads since March 2006 (not accounting cvs checkouts and ROOT users)
- Written in C++, relying on core ROOT functionality
  - Full examples distributed with TMVA, including analysis macros and GUI
  - Scripts are provided for TMVA use in ROOT macro, as C++ executable or with python
- Integrated and distributed with ROOT since ROOT v5.11-03

The *T*MVA Classifiers

- Currently implemented classifiers :
  - Rectangular cut optimisation
  - Projective and multidimensional likelihood estimator
  - Fisher and H-Matrix discriminants
  - Artificial neural networks (three different *multilayer perceptrons*)
  - Boosted/bagged decision trees with automatic node pruning
  - RuleFit
- In work :
  - Support vector machine
  - Committee classifier

Data Preprocessing: Decorrelation

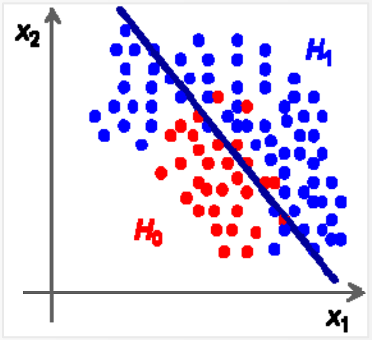
- Commonly realised for all methods in *T*MVA (centrally in **`DataSet`** class)
- Removal of linear correlations by rotating input variables
- Determine *square-root* C' of correlation matrix C, *i.e.*, $C = C'C'$
  - Compute C' by diagonalising C: $D = S^T C S \quad \Rightarrow \quad C' = S\sqrt{D}S^T$
  - Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$
- Various ways to choose basis for decorrelation (also implemented PCA)
- I



Rectangular Cut Optimisation

- Simplest method: cut in rectangular variable volume

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



- Usually training files in TMVA do not contain *realistic* signal and background abundance
  - Cannot optimize for best significance
  - Instead: scan in signal efficiency [0→1] and maximise background rejection
  - From this scan, the best working point (cuts) for any sig/bkg numbers can be derived
- Technical challenge: how to find optimal cuts
  - MINUIT fails due to non-unique solution space
  - TMVA uses: **Monte Carlo sampling**, **<u>Genetics Algorithm</u>**, **Simulated Annealing**
  - Huge speed improvement of volume search by sorting events in binary tree
- Cuts usually benefit from prior decorrelation of cut variables

Projective Likelihood Estimator (PDE Approach)

- Much liked in HEP: probability density estimators for each input variable combined in likelihood estimator

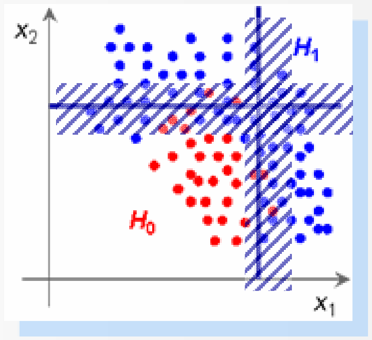
Likelihood ratio for event i_{event}

PDFs

discriminating variables

$$y_L(i_{\text{event}}) = \frac{\prod_{k\in\{\text{variables}\}} p_k^{\text{signal}}\left(x_k(i_{\text{event}})\right)}{\sum_{U\in\{\text{species}\}}\left(\prod_{k\in\{\text{variables}\}} p_k^{U}\left(x_k(i_{\text{event}})\right)\right)}$$

Species: signal, background types



PDE introduces fuzzy logic

- Ignores correlations between input variables
  - Optimal approach if correlations are zero (or linear → decorrelation)
  - Otherwise: significant performance loss
- y_L often strongly peaked $\rightarrow$0,1: transform output (configurable)

$$y_L \rightarrow y_L = -\tau^{-1}\ln\left(y_L^{-1}-1\right)$$

PDE Approach: Estimating PDF Kernels

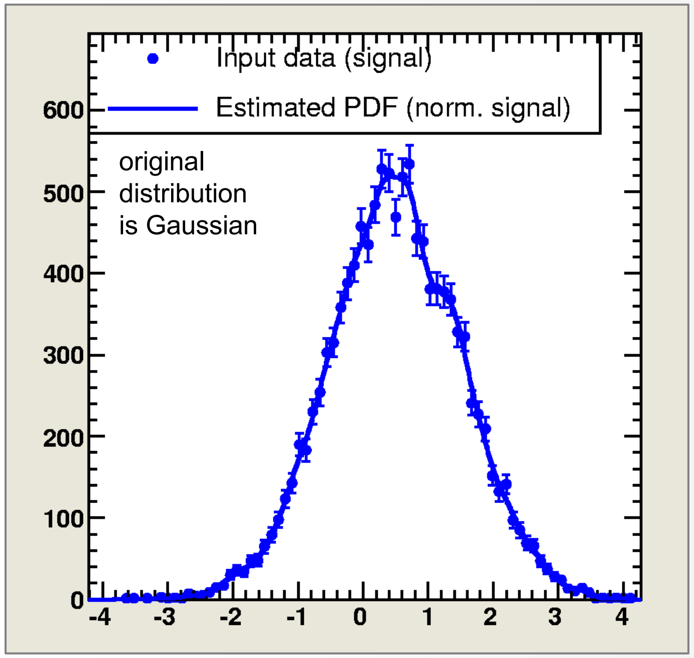
- Technical challenge: how to estimate the PDF shapes

  - 3 ways: **parametric fitting (function)** **nonparametric fitting** **event counting**

    - parametric fitting (function): Difficult to automate for arbitrary PDFs
    - nonparametric fitting: Easy to automate, can create artefacts/suppress information
    - event counting: Automatic, unbiased, but suboptimal

- We have chosen to implement nonparametric fitting in *T*MVA

  - Binned shape interpolation using spline functions (orders: 1, 2, 3, 5)
  - Unbinned kernel density estimation (KDE) with Gaussian smearing
  - *T*MVA performs automatic validation of goodness-of-fit



Multidimensional PDE Approach

- Use a single PDF per event class (sig, bkg), which spans N_{var} dimensions
  - PDE Range-Search: count number of signal and background events in "vicinity" of test event → preset or **adaptive** volume defines "vicinity"

Carli-Koblitz, NIM A501, 576 (2003)

  - The signal estimator is then given by (simplified, full formula accounts for event weights and training population)

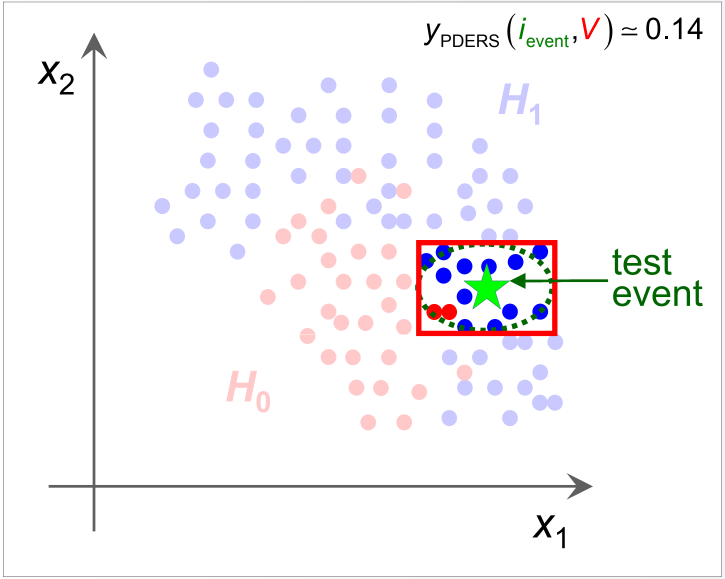
PDE-RS ratio for event i_{event}

chosen volume

#signal events in V

$$y_{PDERS}(i_{event}, V) = \frac{n_S(i_{event}, V)}{n_S(i_{event}, V) + n_B(i_{event}, V)}$$

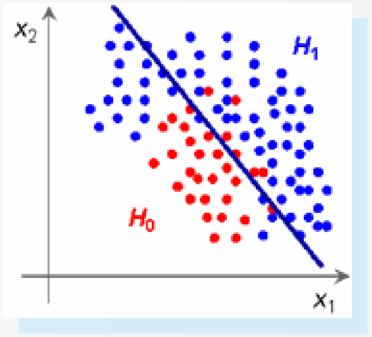
#background events in V



- Improve y_{PDERS} estimate within V by using various N_{var}-D kernel estimators
- Enhance speed of event counting in volume by binary tree search

Fisher Linear Discriminant Analysis (LDA)

- Well known, simple and elegant classifier
  - LDA determines axis in the input variable hyperspace such that a projection of events onto this axis pushes signal and background as far away from each other as possible



- Classifier computation couldn't be simpler:

"Fisher coefficients"

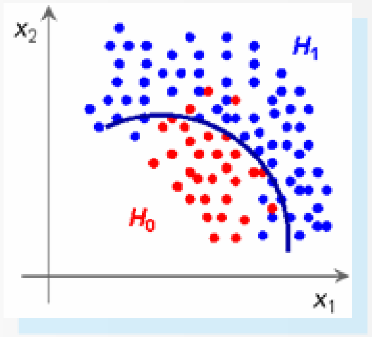
$$y_{\mathrm{Fi}}\left(i_{\mathrm{event}}\right) = F_0 + \sum_{k\in\{\mathrm{variables}\}} x_k\left(i_{\mathrm{event}}\right)\cdot F_k$$

  - Fisher coefficients given by: $F_k \propto \sum_{\ell=1}^{N_{\mathrm{var}}} W_{k\ell}^{-1}\left(\bar{x}_{S,\ell} - \bar{x}_{B,\ell}\right)$, where W is sum $C_S + C_B$
  - Fisher requires distinct sample means between signal and background
  - Optimal classifier for linearly correlated Gaussian-distributed variables

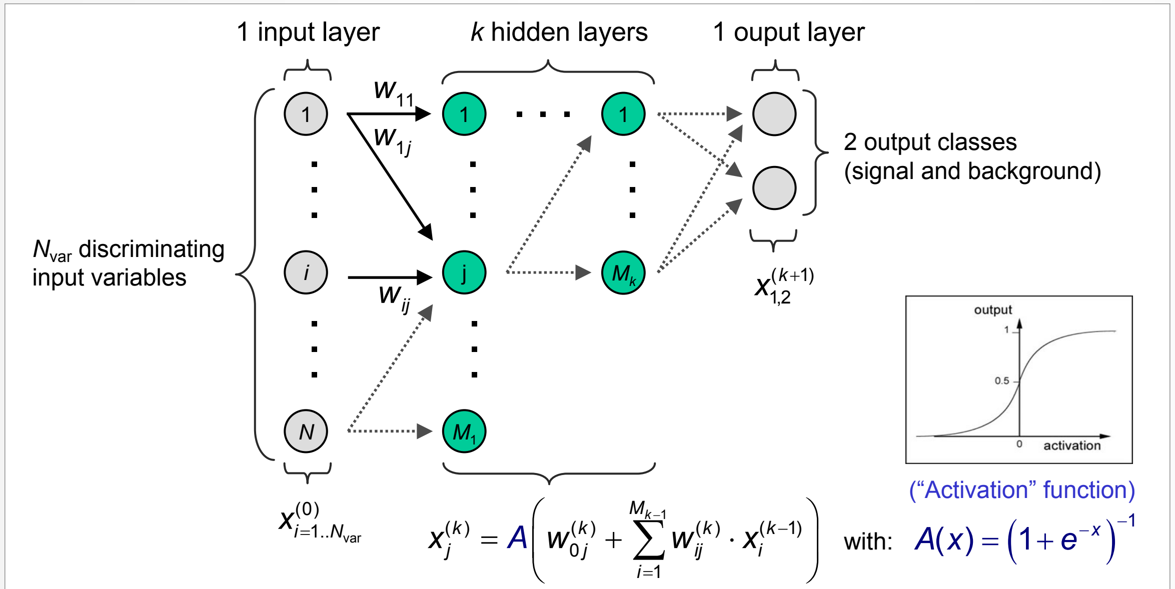
- H-Matrix: poor man's version of Fisher discriminant

Nonlinear Analysis: Artificial Neural Networks

- Achieve nonlinear classifier response by “activating” output nodes using nonlinear weights
- Call nodes “neurons” and arrange them in series:



Feed-forward Multilayer Perceptron



1 input layer
k hidden layers
1 ouput layer
2 output classes (signal and background)
N_{var} discriminating input variables
w_{11}
w_{1j}
w_{ij}
1
i
N
j
M_1
M_k
$x^{(0)}_{i=1..N_{\text{var}}}$
$x^{(k+1)}_{1,2}$
output
activation

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

(“Activation” function)

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

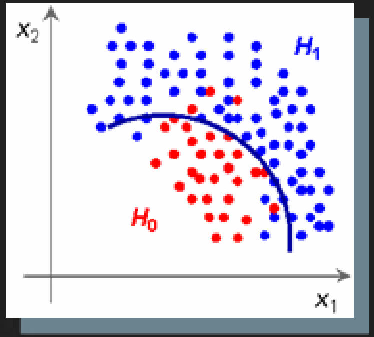
Weierstrass theorem: can approximate any continuous functions to arbitrary precision with a single hidden layer and an infinite number of neurons

Three different multilayer perceptrons available in TMVA

- Adjust weights (=training) using “back-propagation”:
  - For each training event compare *desired* and *received* MLP outputs $\in \{0,1\}$: $\varepsilon = d - r$
  - Correct weights, depending on ε and a “learning rate” η

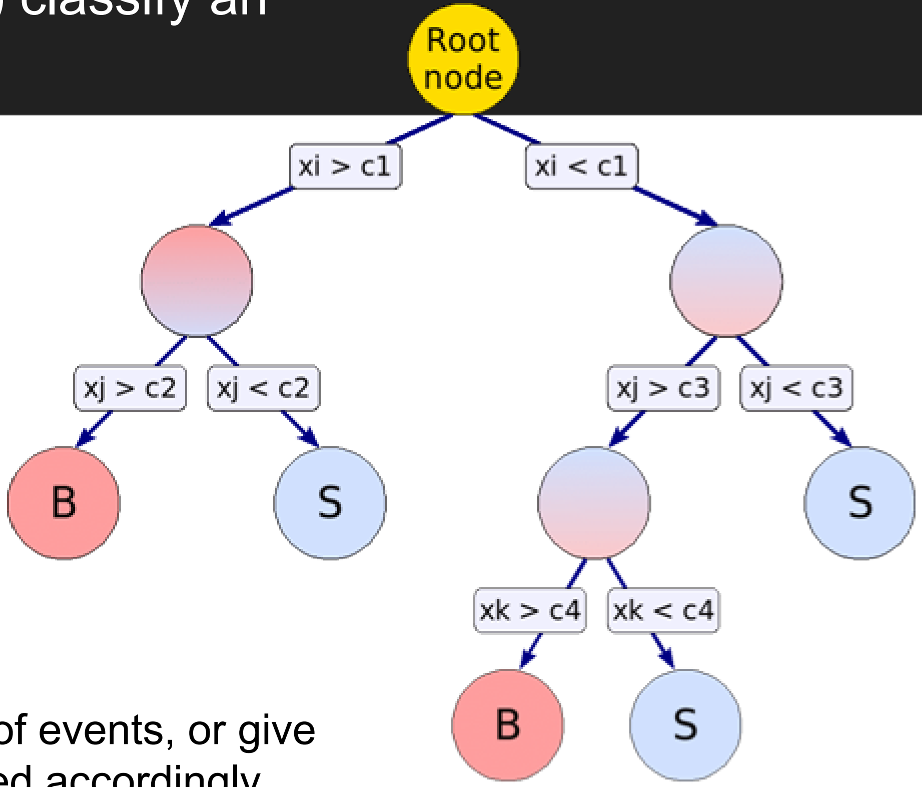
Decision Trees

- Sequential application of cuts splits the data into nodes, where the final nodes (leafs) classify an event as signal or background



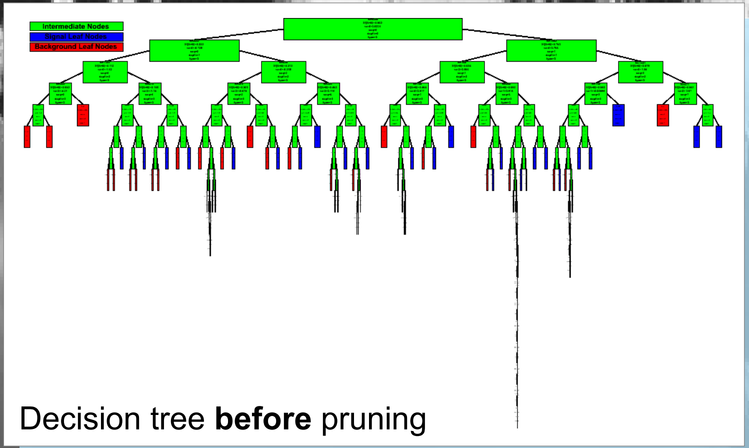
Decision Trees

- Sequential application of cuts splits the data into nodes, where the final nodes (leafs) classify an event as signal or background

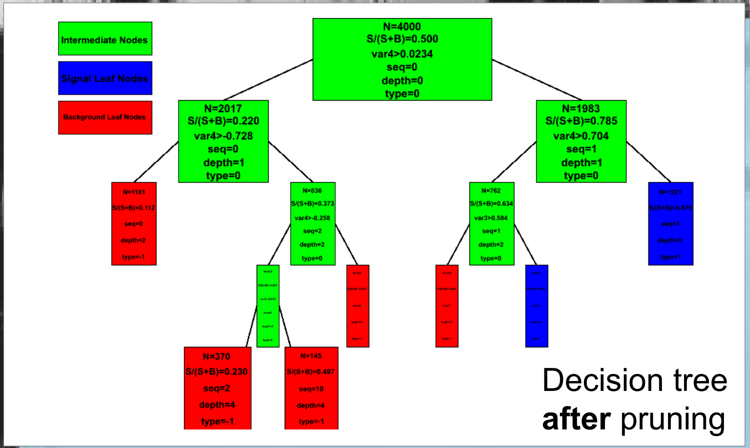


- Growing a decision tree:
  - Start with Root node
  - Split training sample according to cut on best variable at this node
  - Splitting criterion: e.g., maximum "Gini-index": purity × (1– purity)
  - Continue splitting until min. number of events or max. purity reached
  - Classify leaf node according to majority of events, or give weight; unknown test events are classified accordingly
- Bottom-up pruning of a decision tree
  - Remove statistically insignificant nodes to reduce tree overtraining → automatic in *T*MVA

Decision Trees



Decision tree **before** pruning



Intermediate Nodes
Signal Leaf Nodes
Background Leaf Nodes
N=4000 S/(S+B)=0.500 var4>0.0234 seq=0 depth=0 type=0
N=2017 S/(S+B)=0.220 var4>-0.728 seq=0 depth=1 type=0
N=1983 S/(S+B)=0.785 var4>0.704 seq=1 depth=1 type=0
N=1181 S/(S+B)=0.112 seq=0 depth=2 type=-1
N=836 S/(S+B)=0.373 var4>-0.258 seq=2 depth=2 type=0
N=762 S/(S+B)=0.634 var3>0.584 seq=1 depth=2 type=0
N=1221 S/(S+B)=0.879 seq=3 depth=2 type=1
N=370 S/(S+B)=0.230 seq=2 depth=4 type=-1
N=145 S/(S+B)=0.497 seq=10 depth=4 type=-1

Decision tree **after** pruning

- Bottom-up pruning of a decision tree
  - Remove statistically insignificant nodes to reduce tree overtraining → automatic in *T*MVA

Boosted Decision Trees (BDT)

- Data mining with decision trees is popular in science (so far mostly outside of HEP)
  - Advantages:
    - Easy interpretation – can always be represented in 2D tree
    - Independent of monotonous variable transformations, immune against outliers
    - Weak variables are ignored (and don't (much) deteriorate performance)
  - Shortcomings:
    - Instability: small changes in training sample can dramatically alter the tree structure
    - Sensitivity to overtraining (→ requires pruning)
- *Boosted* decision trees: combine *forest* of decision trees, with differently weighted events in each tree (trees can also be weighted), by majority vote
  - e.g., "AdaBoost": incorrectly classified events receive larger weight in next decision tree
  - "Bagging" (instead of boosting): random event weights, resampling with replacement
  - Boosting or bagging are means to create set of "basis functions": the final classifier is linear combination (*expansion*) of these functions

Predictive Learning via Rule Ensembles (RuleFit)

Friedman-Popescu, Tech Rep, Stat. Dpt, Stanford U., 2003

- Following RuleFit approach by Friedman-Popescu
- Model is linear combination of *rules*, where a rule is a sequence of cuts

RuleFit classifier

rules (cut sequence → r_m=1 if all cuts satisfied, =0 otherwise)

normalised discriminating event variables

$$y_{RF}(\vec{x}) = a_0 + \underbrace{\sum_{m=1}^{M_R} a_m r_m(\hat{\vec{x}})}_{\text{Sum of rules}} + \underbrace{\sum_{k=1}^{n_R} b_k \hat{x}_k}_{\text{Linear Fisher term}}$$

- The problem to solve is
  - Create rule ensemble: use forest of decision trees
  - Fit coefficients a_m, b_k: "gradient direct regularization" (Friedman et al.)
- Fast, rather robust and good performance

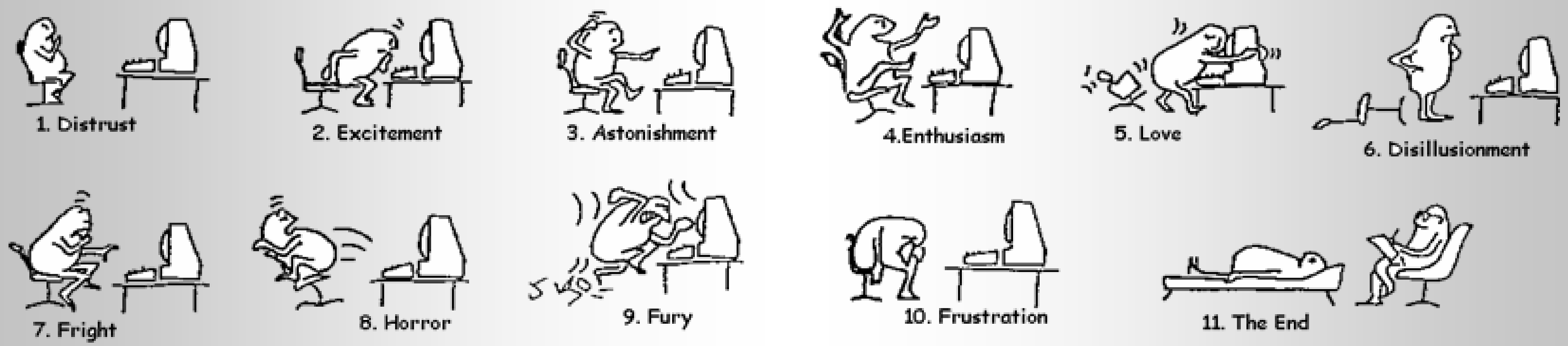
One of the elementary cellular automaton rules (Wolfram 1983, 2002). It specifies the next color in a cell, depending on its color and its immediate neighbors. Its rule outcomes are encoded in the binary representation $30=00011110_2$.

Using *T*MVA

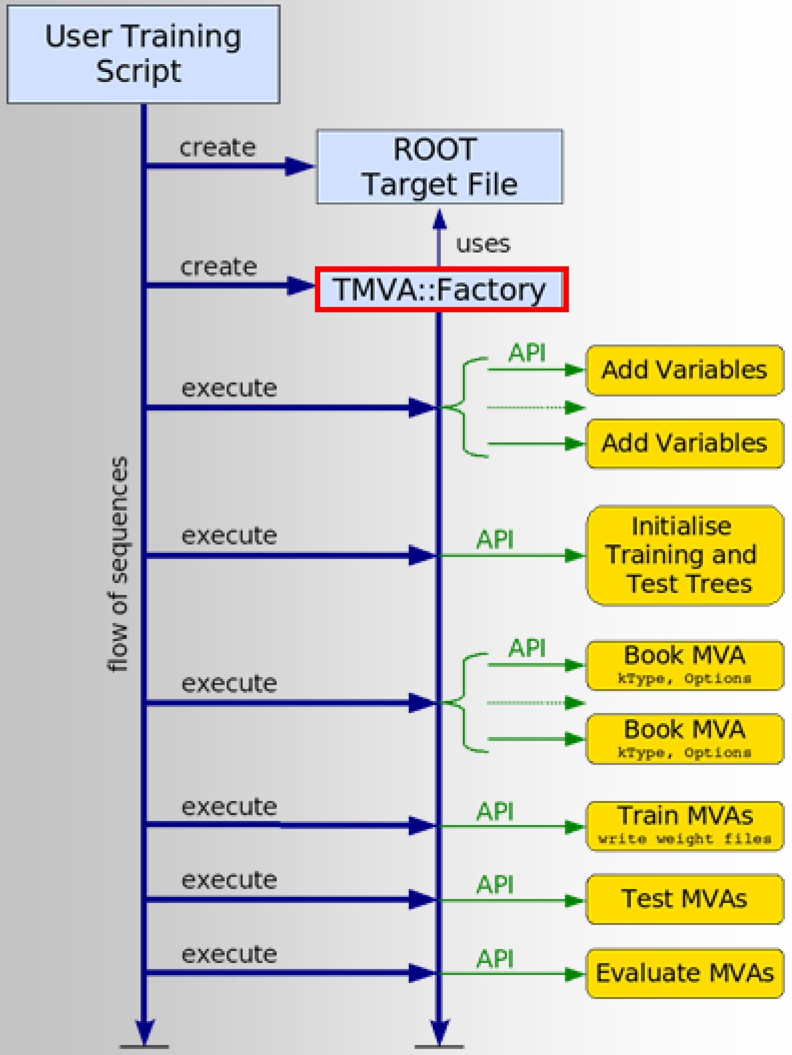
A typical *T*MVA analysis consists of two main steps:

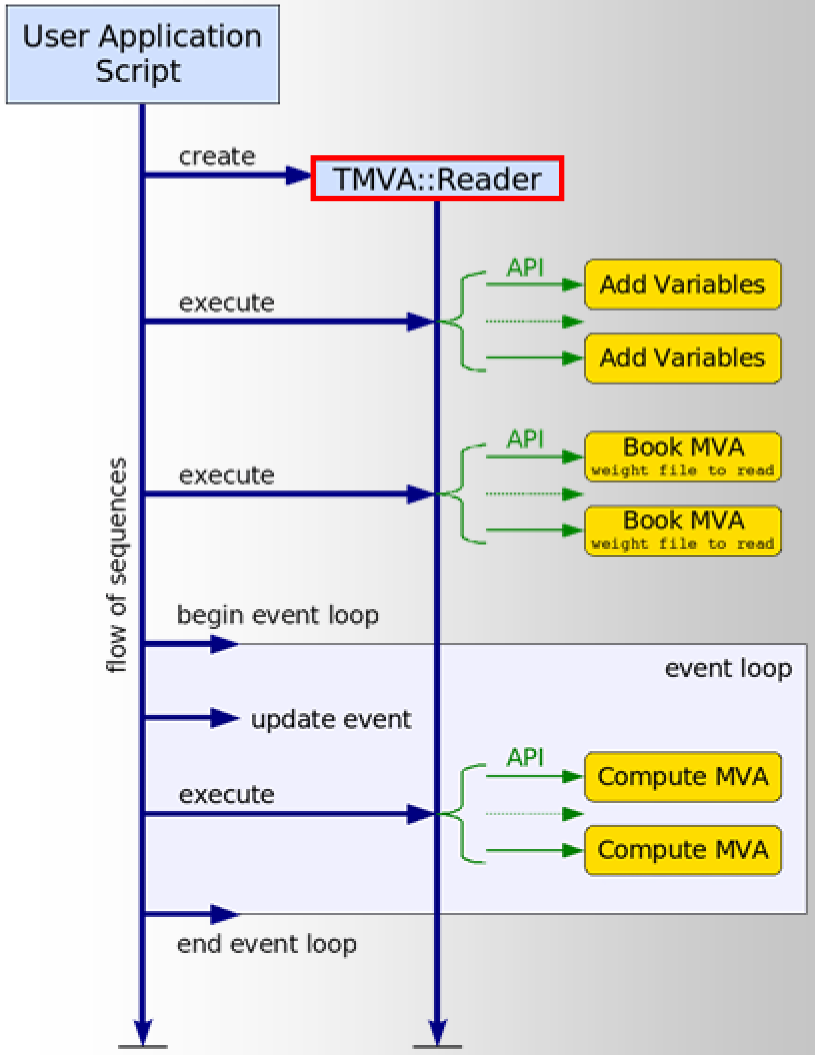
1. *Training phase*: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. *Application phase*: using selected trained classifiers to classify unknown data samples

➡ Illustration of these steps with toy data samples

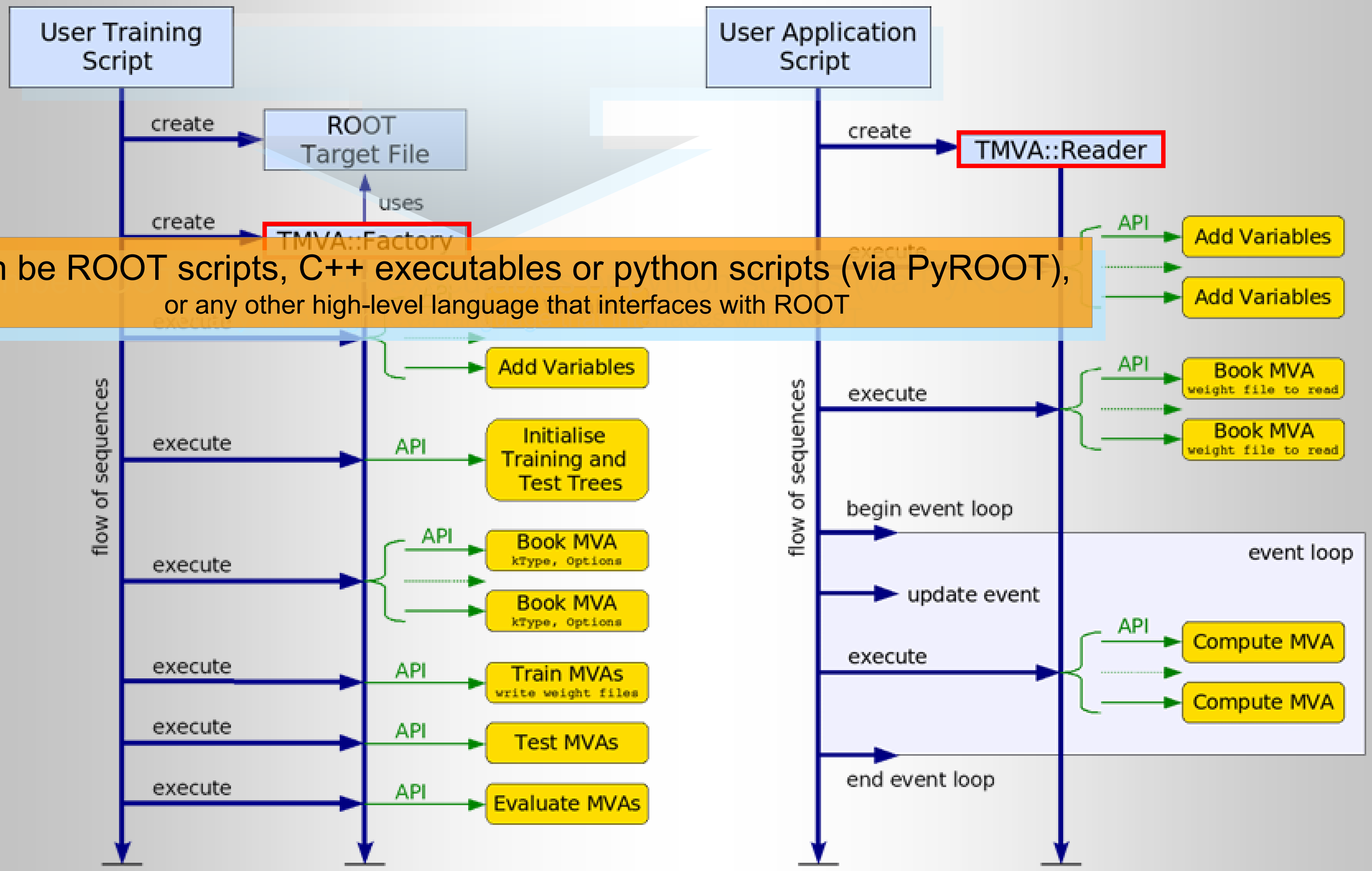


Code Flow for *Training* and *Application* Phases





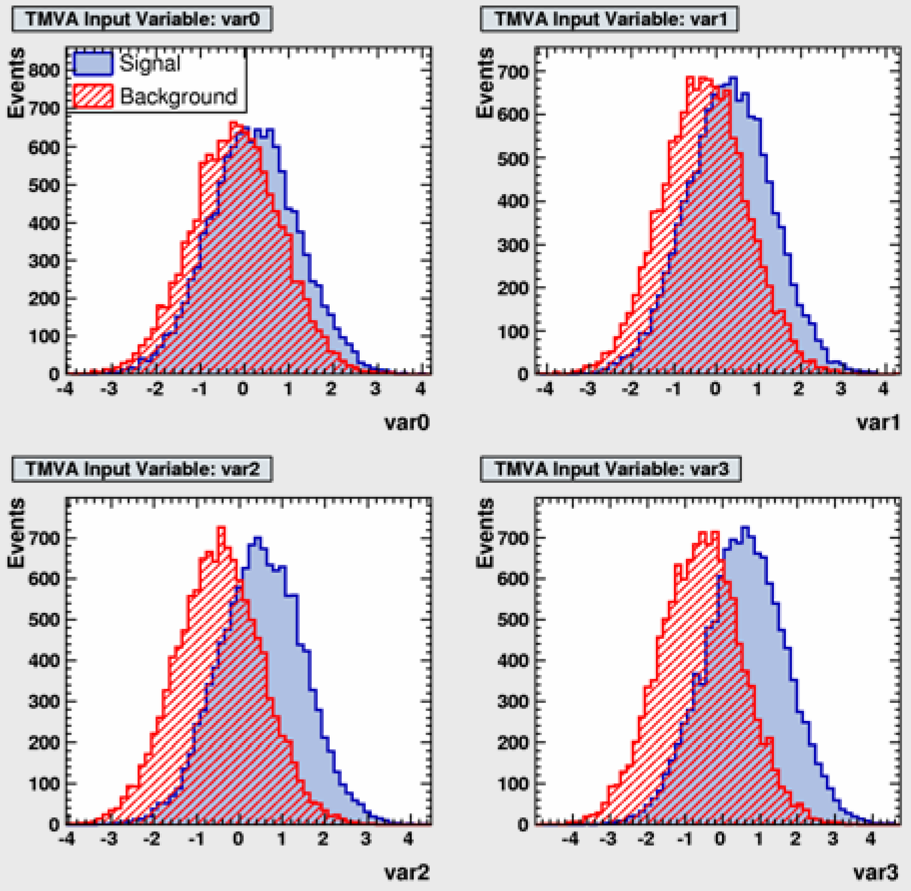
Code Flow for *Training* and *Application* Phases

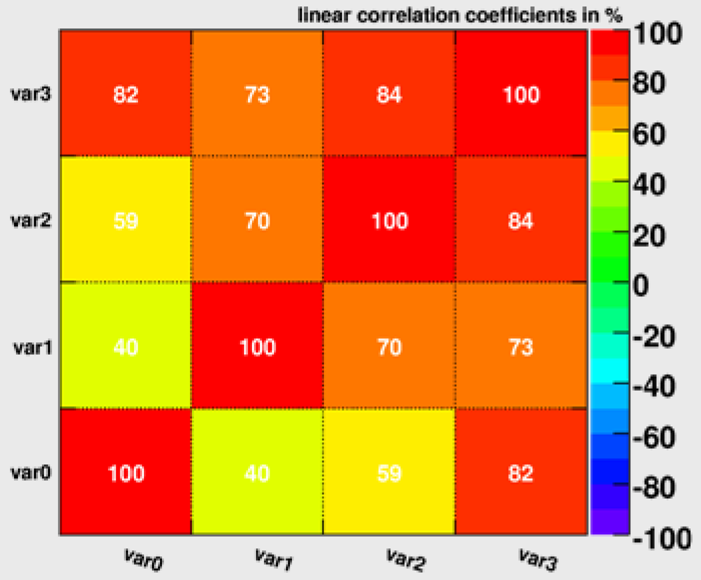


Can be ROOT scripts, C++ executables or python scripts (via PyROOT),
or any other high-level language that interfaces with ROOT

A Toy Example (idealized)

- Use data set with 4 linearly correlated Gaussian distributed variables:



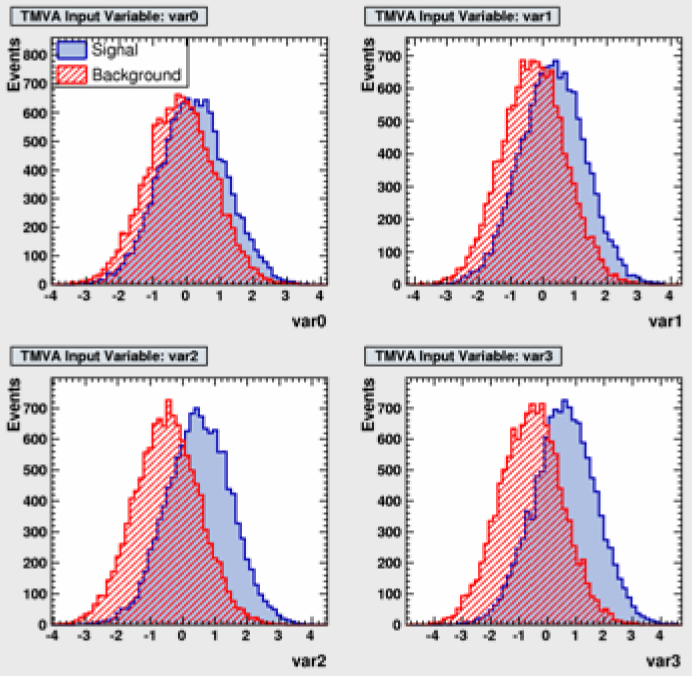


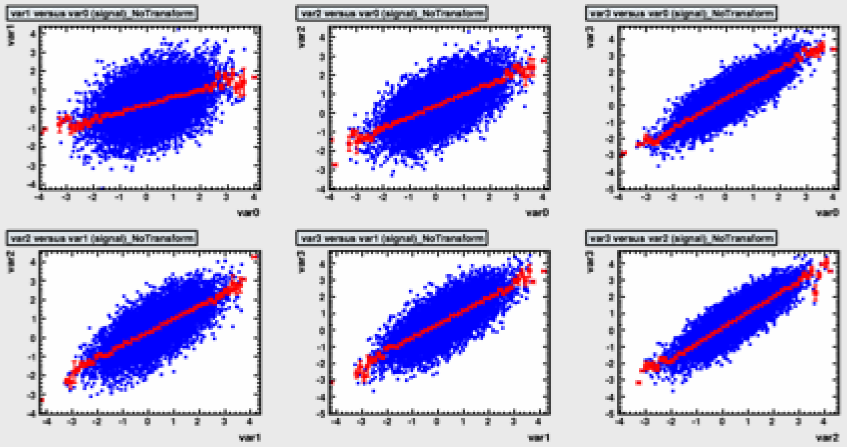
	var0	var1	var2	var3
var3	82	73	84	100
var2	59	70	100	84
var1	40	100	70	73
var0	100	40	59	82

```
--------------------------------------
Rank : Variable  : Separation
--------------------------------------
   1 : var3        : 3.834e+02
   2 : var2        : 3.062e+02
   3 : var1        : 1.097e+02
   4 : var0        : 5.818e+01
--------------------------------------
```

Preprocessing the Input Variables

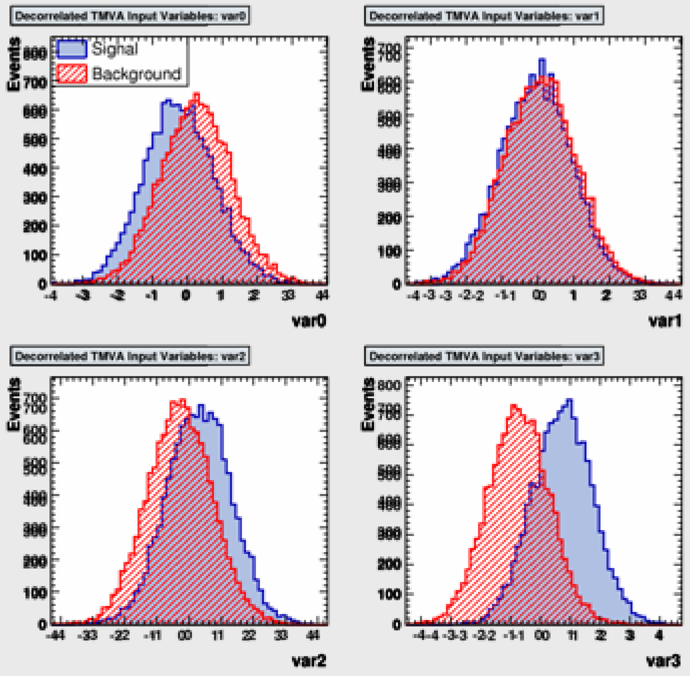
- Decorrelation of variables before training is useful for *this* example

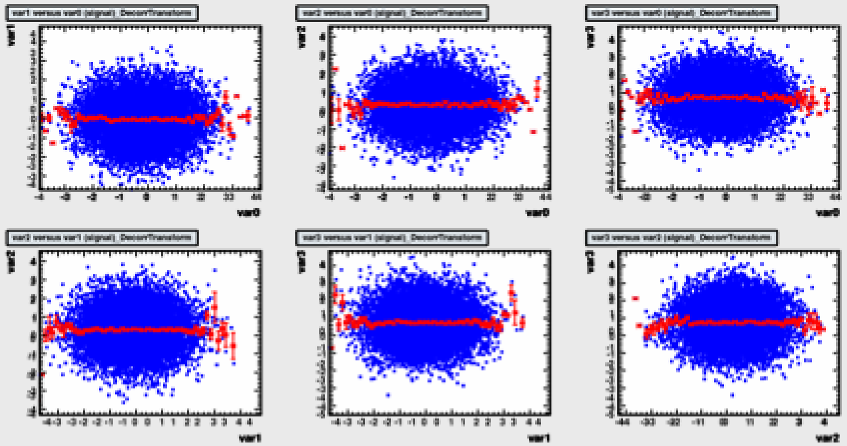




Preprocessing the Input Variables

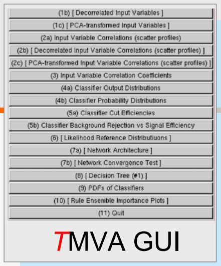
- Decorrelation of variables before training is useful for *this* example





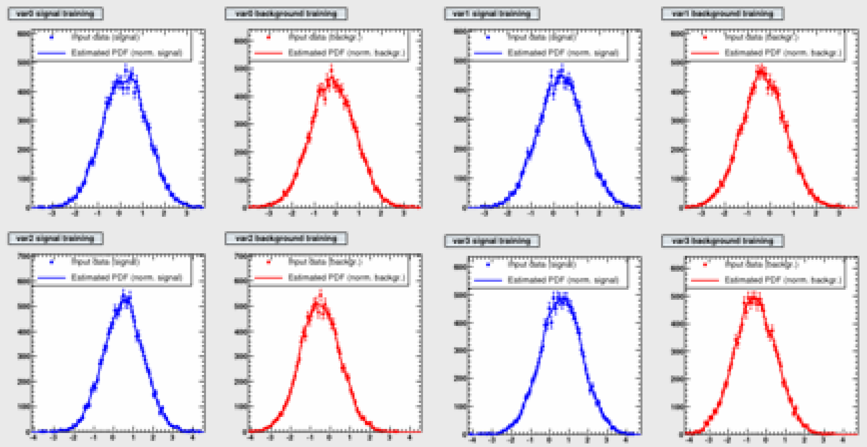
  - Similar distributions for PCA
  - Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

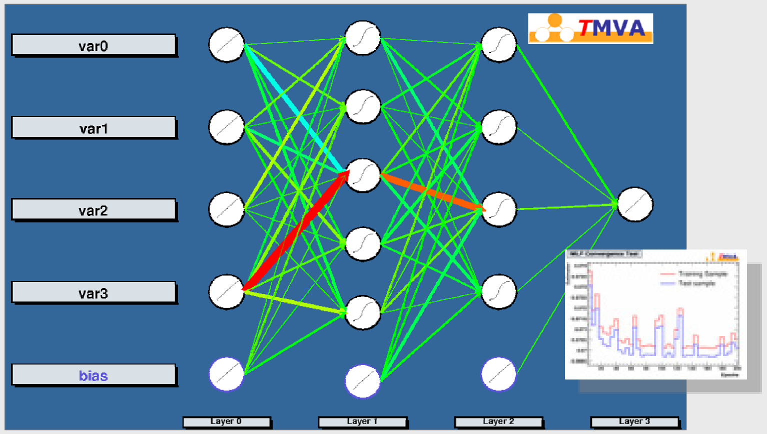
Validating the Classifier Training

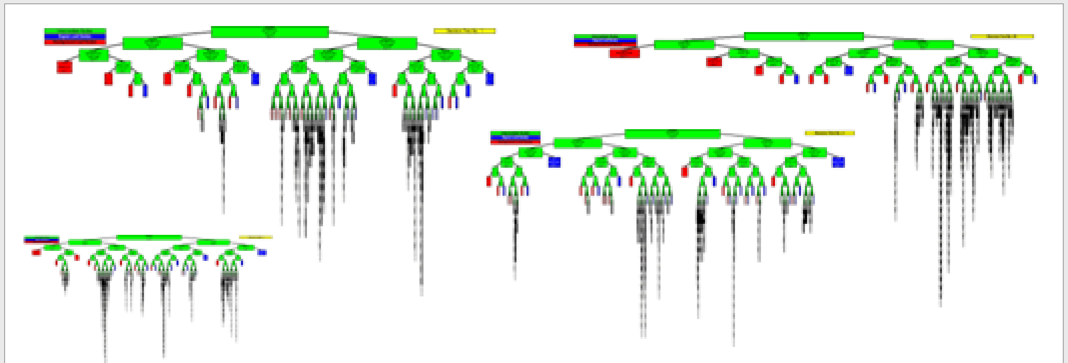


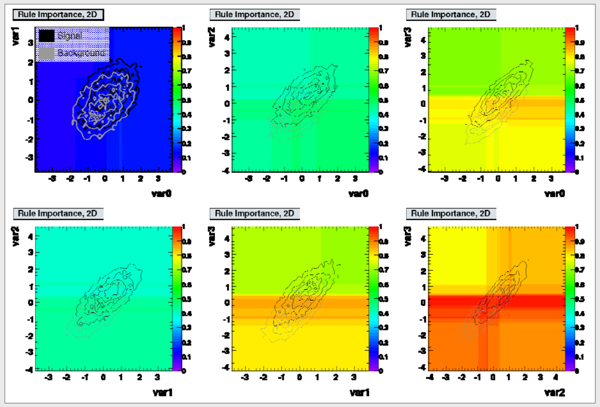
(1b) [Decorrelated Input Variables]
(1c) [PCA-transformed Input Variables]
(2a) Input Variable Correlations (scatter profiles)
(2b) [Decorrelated Input Variable Correlations (scatter profiles)]
(2c) [PCA-transformed Input Variable Correlations (scatter profiles)]
(3) Input Variable Correlation Coefficients
(4a) Classifier Output Distributions
(4b) Classifier Probability Distributions
(5a) Classifier Cut Efficiencies
(5b) Classifier Background Rejection vs Signal Efficiency
(6) [Likelihood Reference Distributiuons]
(7a) [Network Architecture]
(7b) [Network Convergence Test]
(8) [Decision Tree (#1)]
(9) PDFs of Classifiers
(10) [Rule Ensemble Importance Plots]
(11) Quit
TMVA GUI

- Projective likelihood PDFs, MLP training, BDTs, …



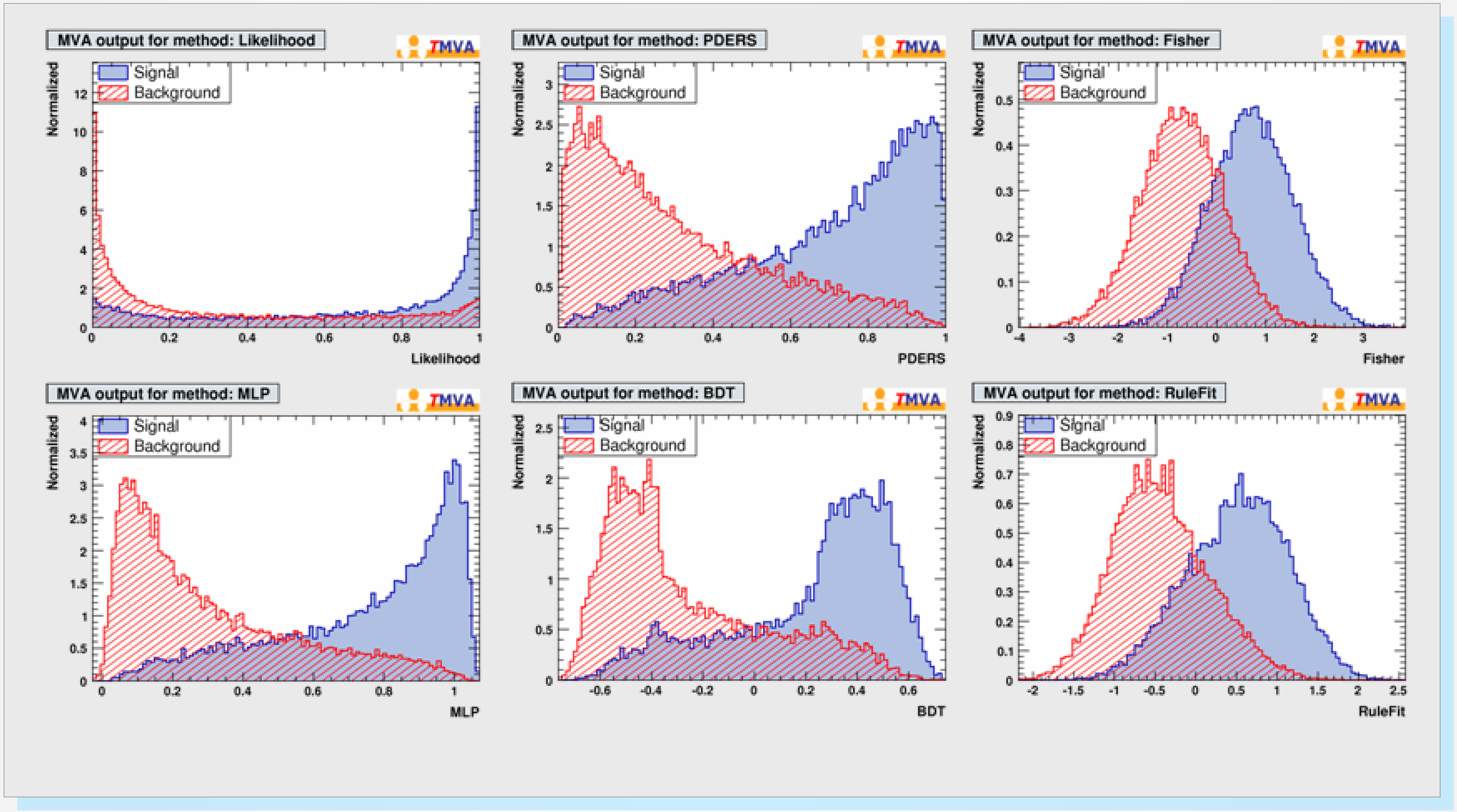



average no. of nodes before/after pruning: 4193 / 968



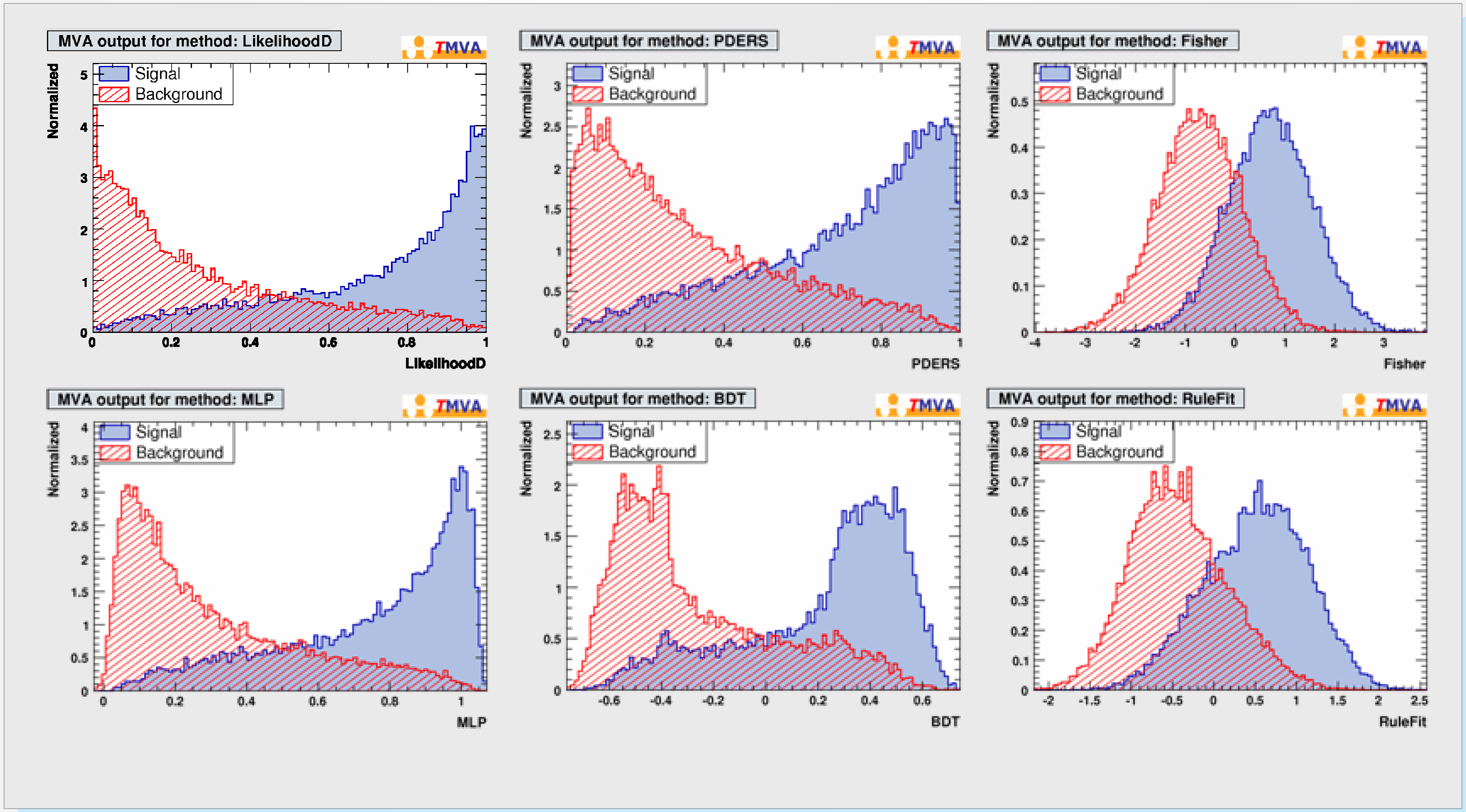
Testing the Classifiers

- Classifier output distributions for independent test sample:



Testing the Classifiers

- Classifier output distributions for independent test sample:

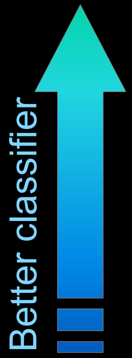


Evaluating the Classifiers

- There is no unique way to express the performance of a classifier
  → several benchmark quantities computed by *T*MVA
  - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
  - The *Separation*: $\frac{1}{2}\int\frac{(\hat{y}_S(y)-\hat{y}_B(y))^2}{\hat{y}_S(y)+\hat{y}_B(y)}dy$
  - The discrimination *Significance*: $(\bar{y}_S-\bar{y}_B)\Big/\sqrt{\sigma_{y,S}^2+\sigma_{y,B}^2}$
  - The average of the signal μ-transform: $\int y\mu(\hat{y}_S(y))\,dy$
    (the μ-transform of a classifier yields a uniform background distribution, so that the signal shapes can be directly compared among the classifiers)

- Remark on **overtraining**
  - Occurs when classifier training has too few degrees of freedom because the classifier has too many adjustable parameters for too few training events
  - Sensitivity to overtraining depends on classifier: e.g., Fisher weak, BDT strong
  - Compare performance between training and test sample to detect overtraining
  - Actively counteract overtraining: e.g., smooth likelihood PDFs, prune decision trees, …

Evaluating the Classifiers (taken from TMVA output…)



```
    Evaluation results ranked by best signal efficiency and purity (area)
 ------------------------------------------------------------------------------
  MVA                Signal efficiency at bkg eff. (error):  |  Sepa-    Signifi-
  Methods:           @B=0.01    @B=0.10    @B=0.30    Area   |  ration:  cance:
 ------------------------------------------------------------------------------
  Fisher           : 0.268(03)  0.653(03)  0.873(02)  0.882  |  0.444    1.189
  MLP              : 0.266(03)  0.656(03)  0.873(02)  0.882  |  0.444    1.260
  LikelihoodD      : 0.259(03)  0.649(03)  0.871(02)  0.880  |  0.441    1.251
  PDERS            : 0.223(03)  0.628(03)  0.861(02)  0.870  |  0.417    1.192
  RuleFit          : 0.196(03)  0.607(03)  0.845(02)  0.859  |  0.390    1.092
  HMatrix          : 0.058(01)  0.622(03)  0.868(02)  0.855  |  0.410    1.093
  BDT              : 0.154(02)  0.594(04)  0.838(03)  0.852  |  0.380    1.099
  CutsGA           : 0.109(02)  1.000(00)  0.717(03)  0.784  |  0.000    0.000
  Likelihood       : 0.086(02)  0.387(03)  0.677(03)  0.757  |  0.199    0.682
 ------------------------------------------------------------------------------
```

Evaluating the Classifiers (taken from TMVA output…)

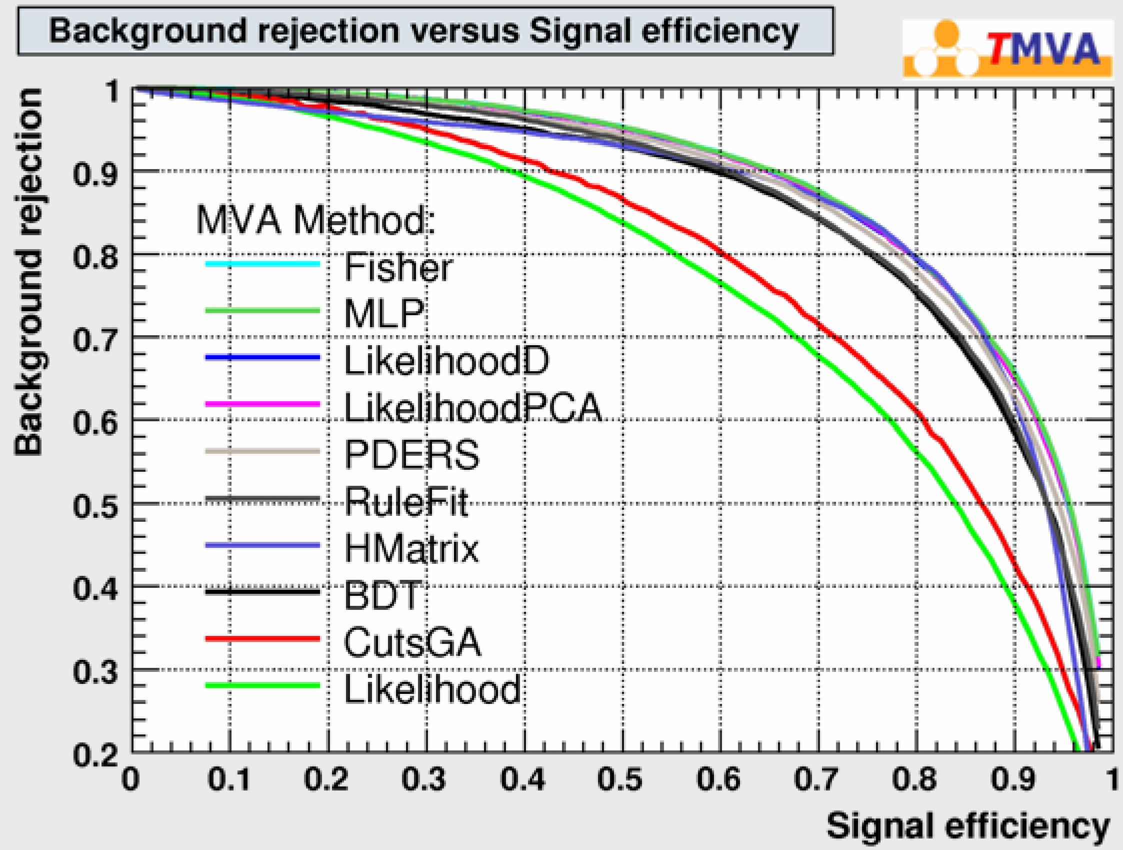
Better classifier

```
     Evaluation results ranked by best signal efficiency and purity (area)
 ------------------------------------------------------------------------------
 MVA                 Signal efficiency at bkg eff. (error):  |  Sepa-    Signifi-
 Methods:            @B=0.01    @B=0.10    @B=0.30    Area   |  ration:  cance:
 ------------------------------------------------------------------------------
  Fisher           : 0.268(03)  0.653(03)  0.873(02)  0.882  |  0.444    1.189
  MLP              : 0.266(03)  0.656(03)  0.873(02)  0.882  |  0.444    1.260
  LikelihoodD      : 0.259(03)  0.649(03)  0.871(02)  0.880  |  0.441    1.251
  PDERS            : 0.223(03)  0.628(03)  0.861(02)  0.870  |  0.417    1.192
  RuleFit          : 0.196(03)  0.607(03)  0.845(02)  0.859  |  0.390    1.092
  HMatrix          : 0.058(01)  0.622(03)  0.868(02)  0.855  |  0.410    1.093
  BDT              : 0.154(02)  0.594(04)  0.838(03)  0.852  |  0.380    1.099
  CutsGA           : 0.109(02)  1.000(00)  0.717(03)  0.784  |  0.000    0.000
  Likelihood       : 0.086(02)  0.387(03)  0.677(03)  0.757  |  0.199    0.682
 ------------------------------------------------------------------------------
    Testing efficiency compared to training efficiency (overtraining check)
 ------------------------------------------------------------------------------
       MVA           Signal efficiency: from test sample (from traing sample)
       Methods:         @B=0.01             @B=0.10            @B=0.30
 ------------------------------------------------------------------------------
     Fisher          : 0.268 (0.275)       0.653 (0.658)      0.873 (0.873)
     MLP             : 0.266 (0.278)       0.656 (0.658)      0.873 (0.873)
     LikelihoodD     : 0.259 (0.273)       0.649 (0.657)      0.871 (0.872)
     PDERS           : 0.223 (0.389)       0.628 (0.691)      0.861 (0.881)
     RuleFit         : 0.196 (0.198)       0.607 (0.616)      0.845 (0.848)
     HMatrix         : 0.058 (0.060)       0.622 (0.623)      0.868 (0.868)
     BDT             : 0.154 (0.268)       0.594 (0.736)      0.838 (0.911)
     CutsGA          : 0.109 (0.123)       1.000 (0.424)      0.717 (0.715)
     Likelihood      : 0.086 (0.092)       0.387 (0.379)      0.677 (0.677)
 -----------------------------------------------------------------------------
```

Check for over-training

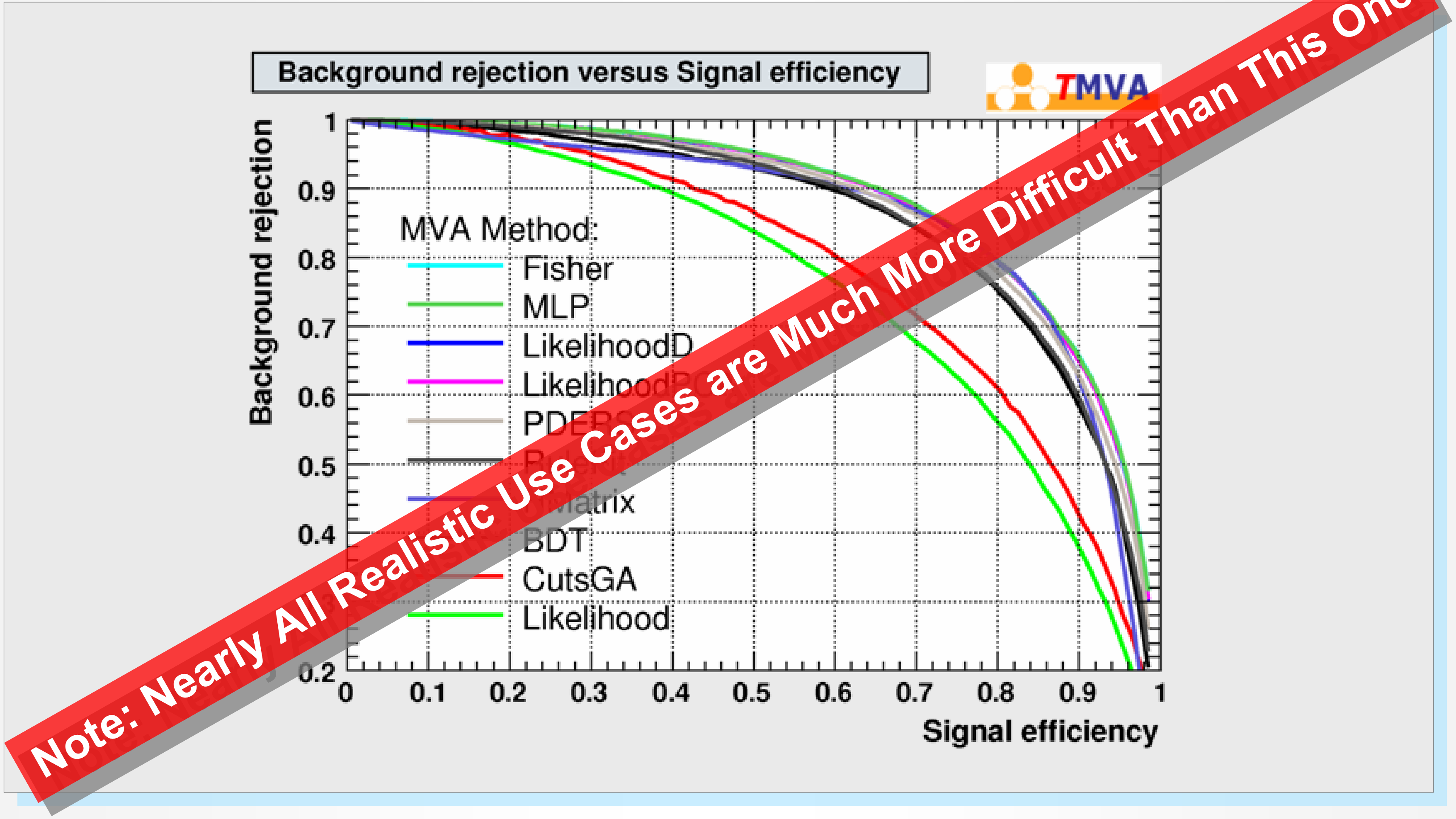
Evaluating the Classifiers (with a single plot…)

- Smooth background rejection versus signal efficiency curve:
  (from cut on classifier output)



Evaluating the Classifiers (with a single plot…)

- Smooth background rejection versus signal efficiency curve:
  (from cut on classifier output)



Note: Nearly All Realistic Use Cases are Much More Difficult Than This One

Remarks

- Additional information provided during the *T*MVA training phase
  - Classifiers provide specific ranking of input variables
  - Correlation matrix and classification "overlap" matrix for classifiers: if two classifiers have similar performance, but significant non-overlapping classifications → combine them!
  - Such a combination of classifiers is denoted a "Committee classifier": currently under development (BDTs and RuleFit are committees of base learners, Bayesian ANNs are committees of ANNs, etc)
  - PDFs for the classifier outputs can be used to derive signal probabilities:

$$P_{\mathrm{MVA}}\left(i_{\mathrm{event}}\right)=\frac{f_S\hat{y}_S(i_{\mathrm{event}})}{f_S\hat{y}_S(i_{\mathrm{event}})+\left(1-f_S\right)\hat{y}_B(i_{\mathrm{event}})}\ ,$$

with f_S the signal fraction in sample

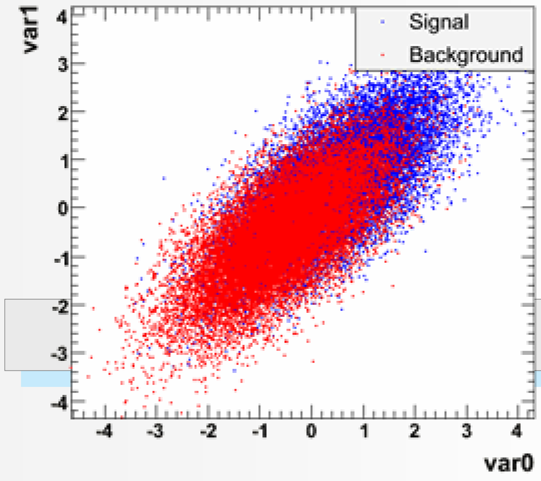
  - All classifiers write ROOT and text weight files for configuration and training results → feeds the application phase (Reader)

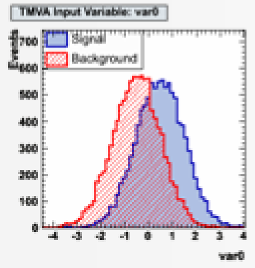
More Toy Examples

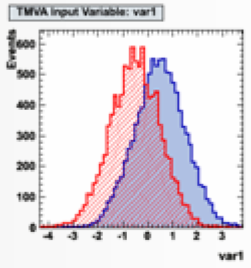
More Toys: Linear-, Cross-, Circular Correlations

- Illustrate the behaviour of linear and nonlinear classifiers

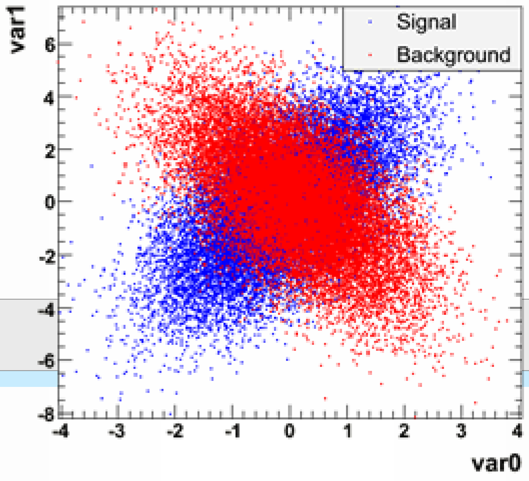
Linear correlations
(same for signal and background)

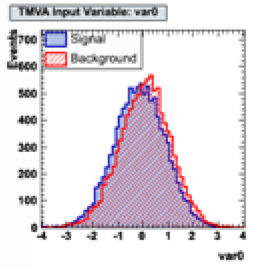


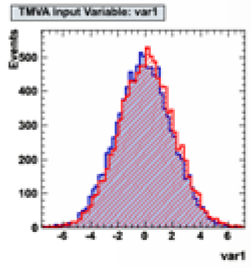




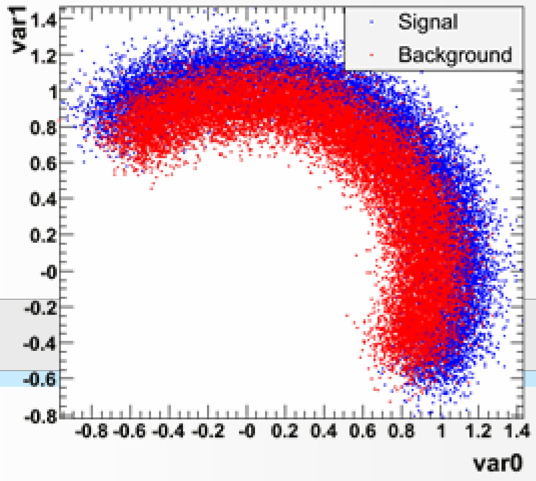
Linear correlations
(opposite for signal and background)

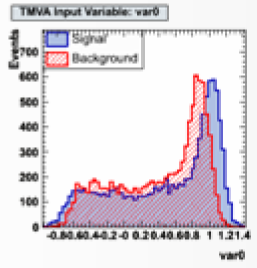


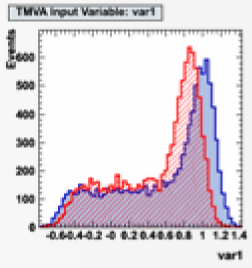




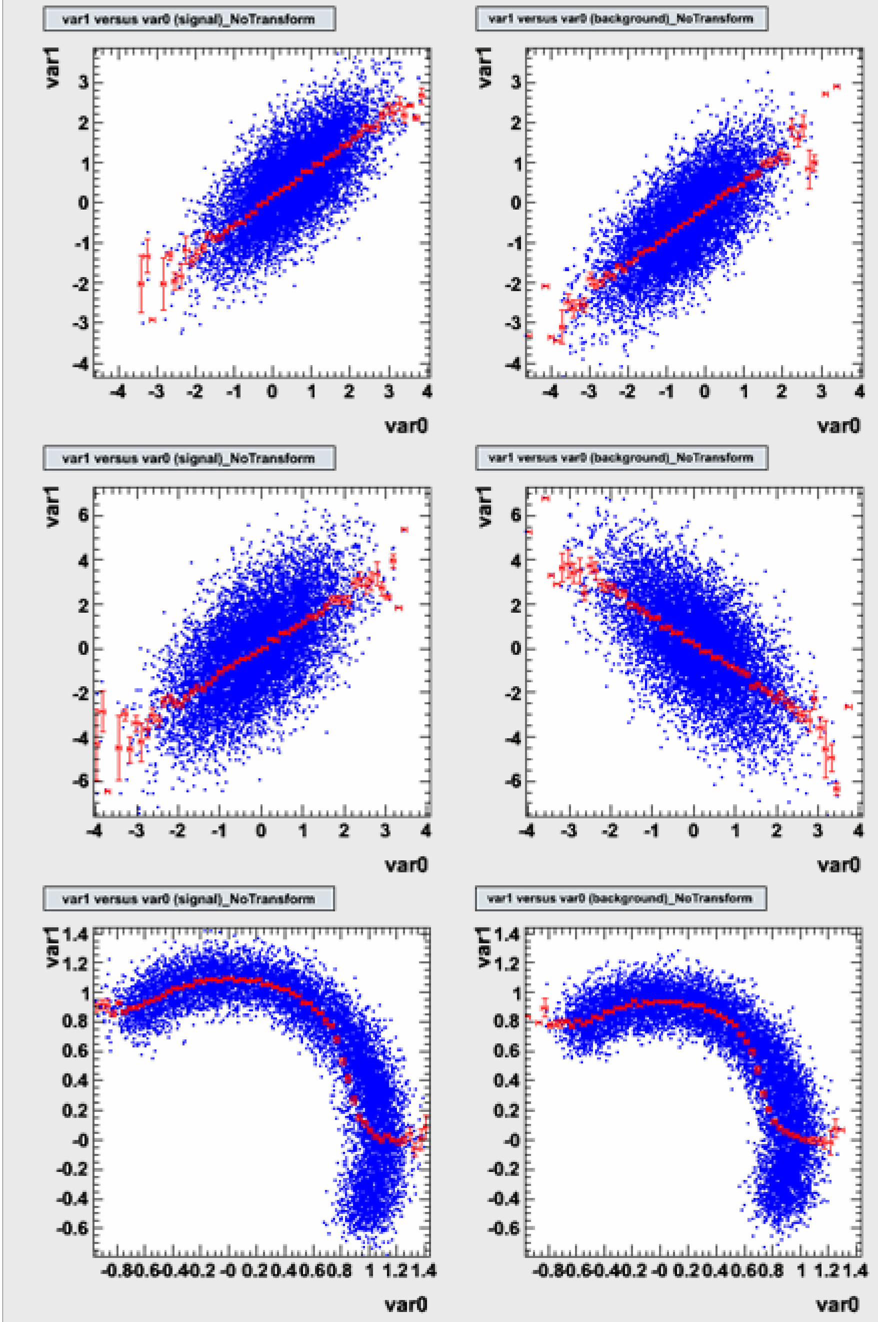
Circular correlations
(same for signal and background)







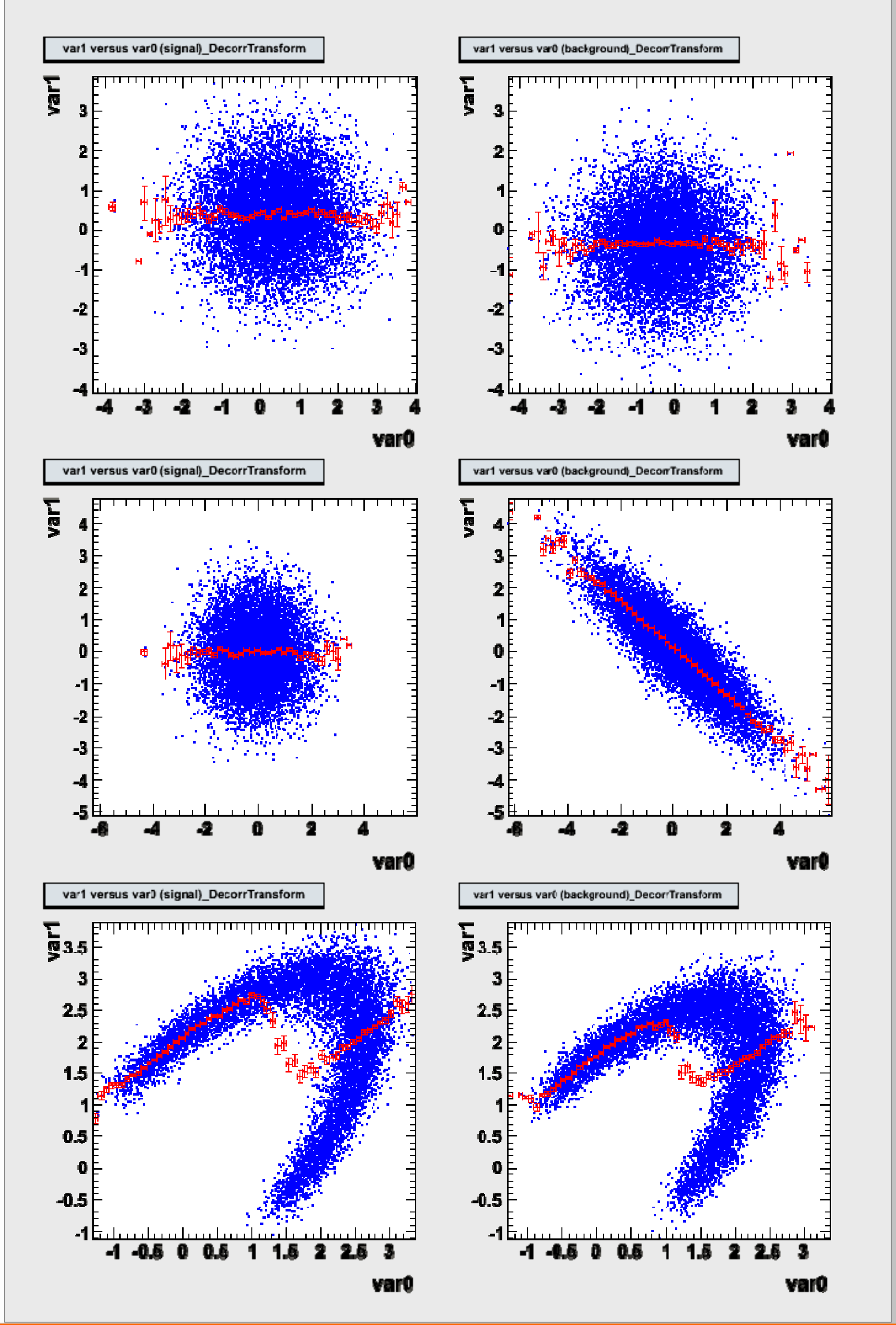
- How does linear decorrelation affect strongly nonlinear cases ?



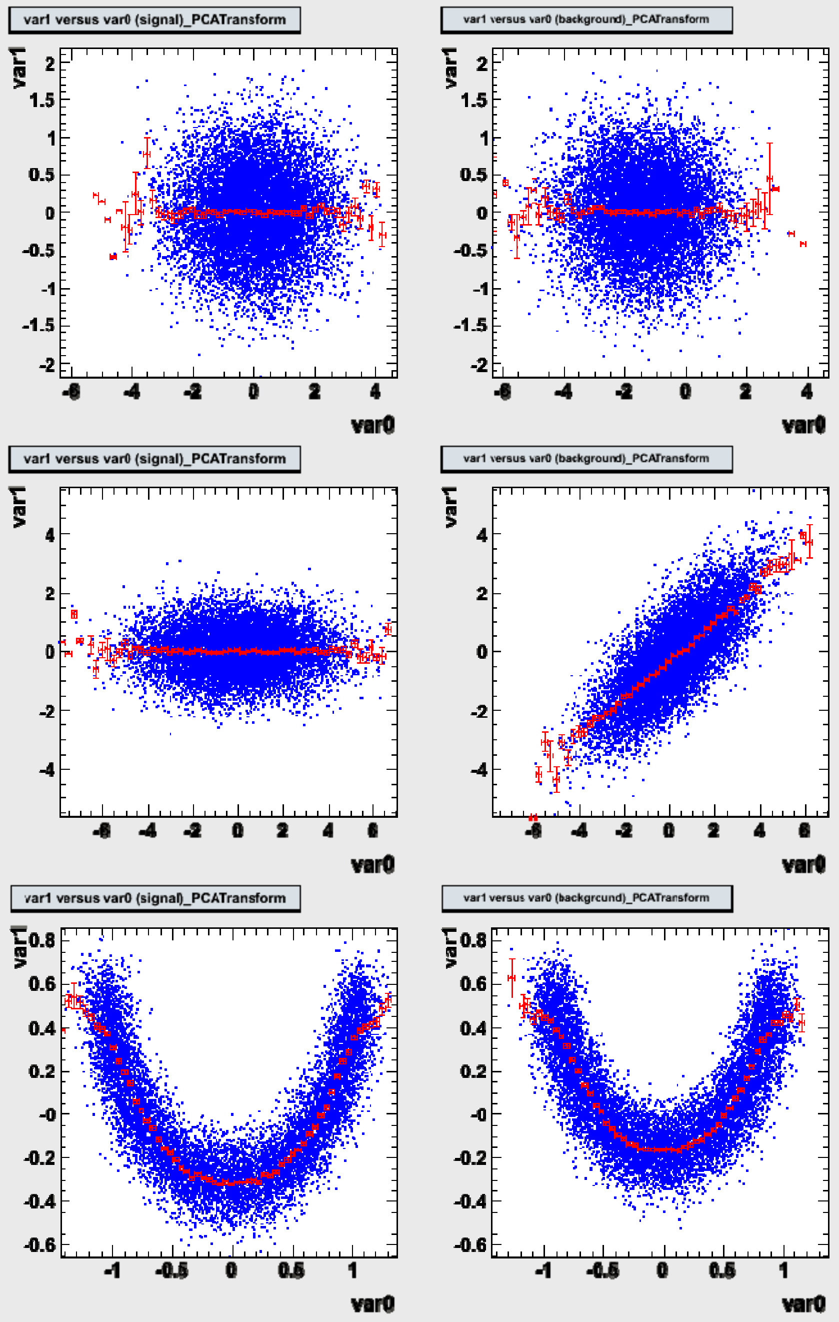
Original correlations

- How does linear decorrelation affect strongly nonlinear cases ?

SQRT decorrelation



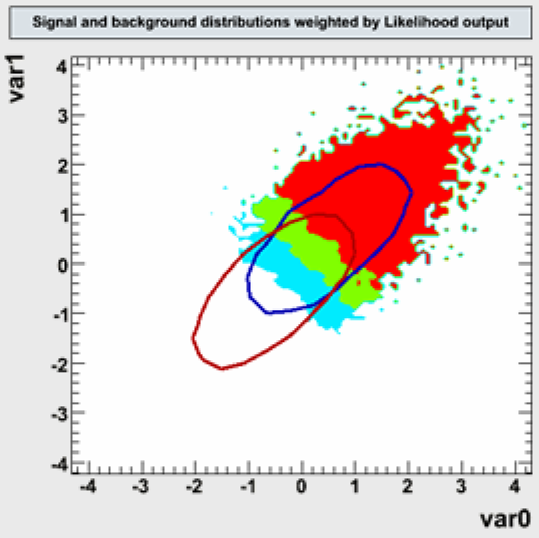
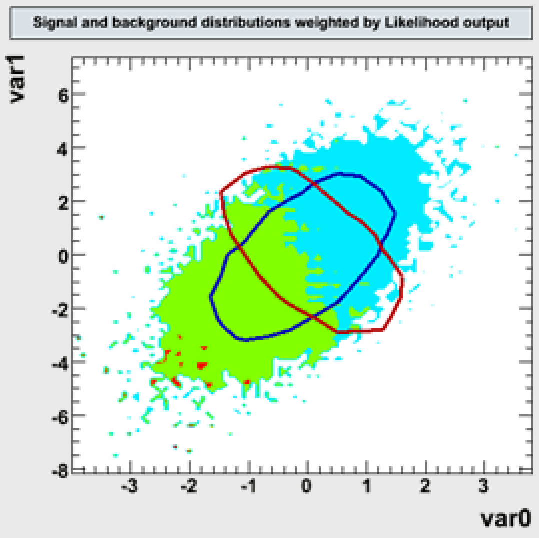
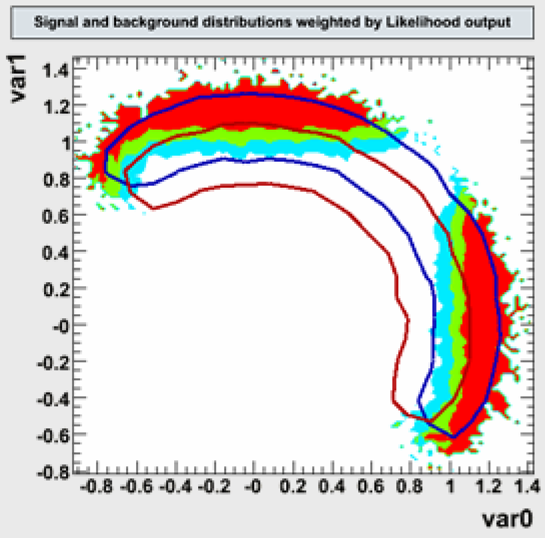
- How does linear decorrelation affect strongly nonlinear cases ?



PCA decorrelation

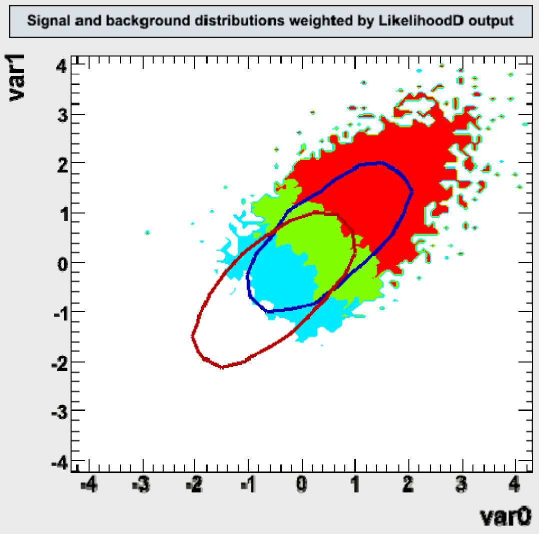
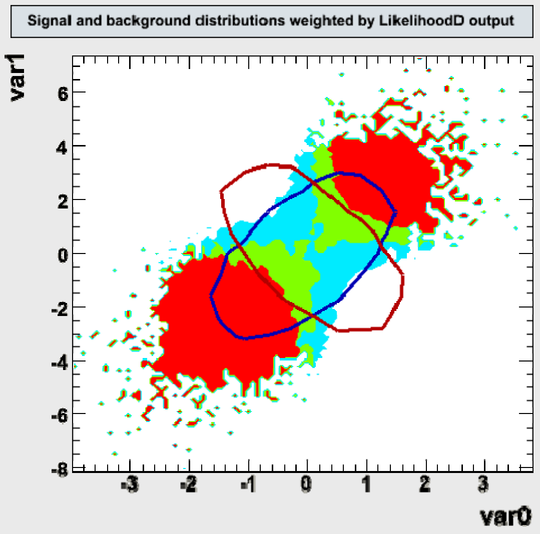
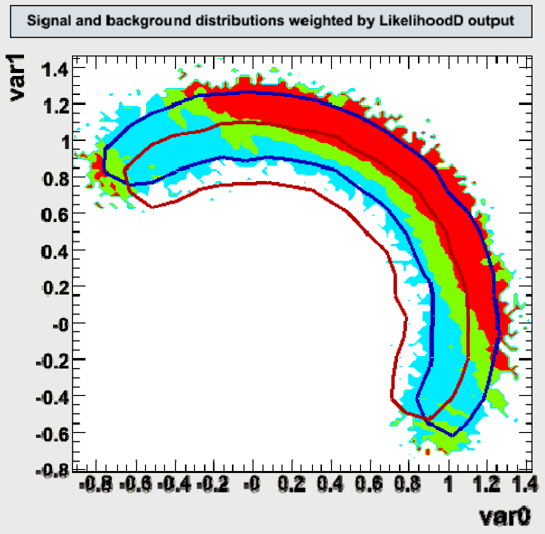
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

Linear correlations (same for signal and background)	Linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)
		

Weight Variables by Classifier Performance

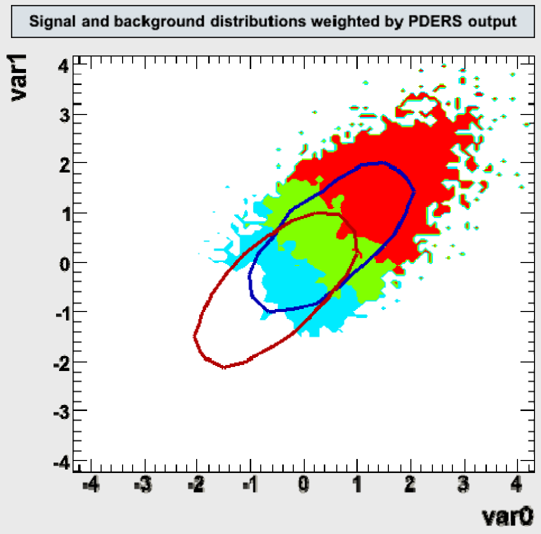
- How well do the classifier resolve the various correlation patterns ?

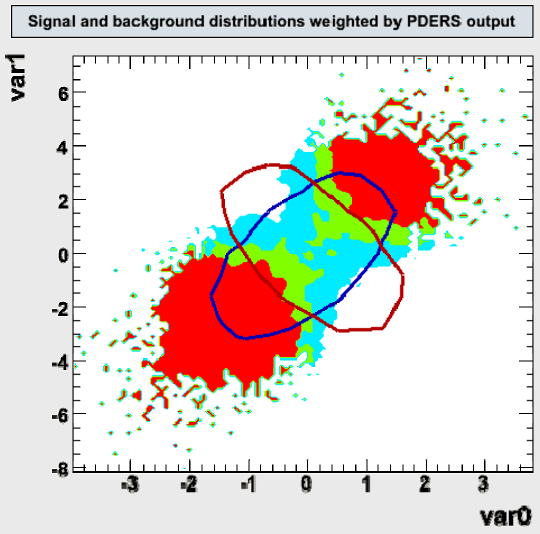
Linear correlations (same for signal and background)	Linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)
		

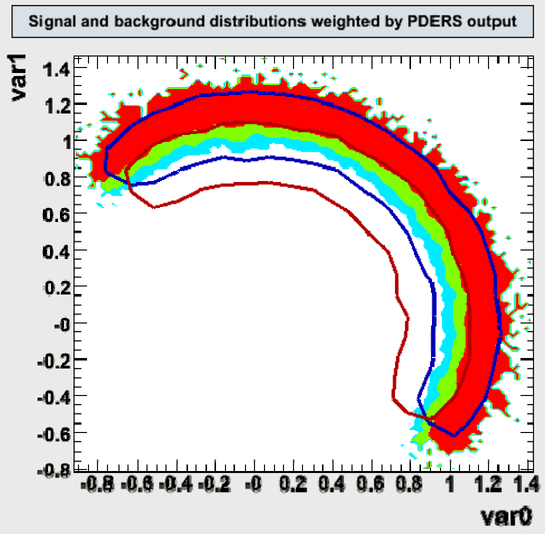
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

Linear correlations (same for signal and background)	Linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)



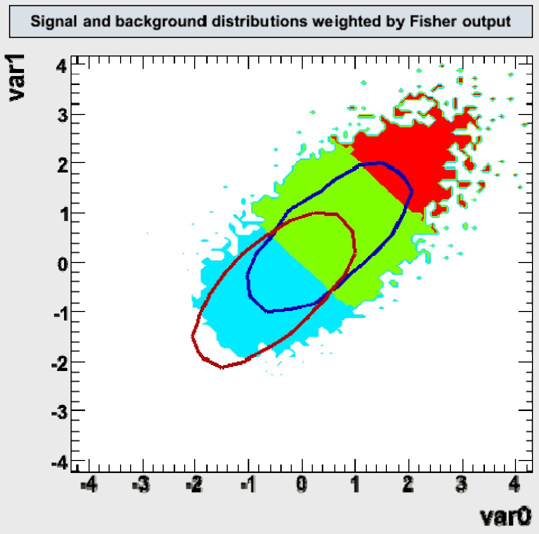




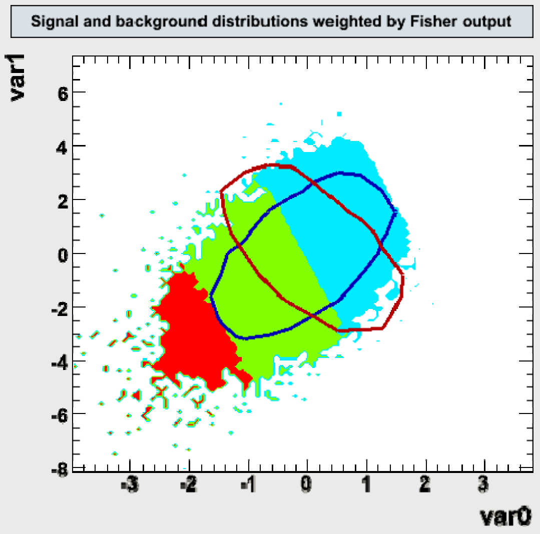
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

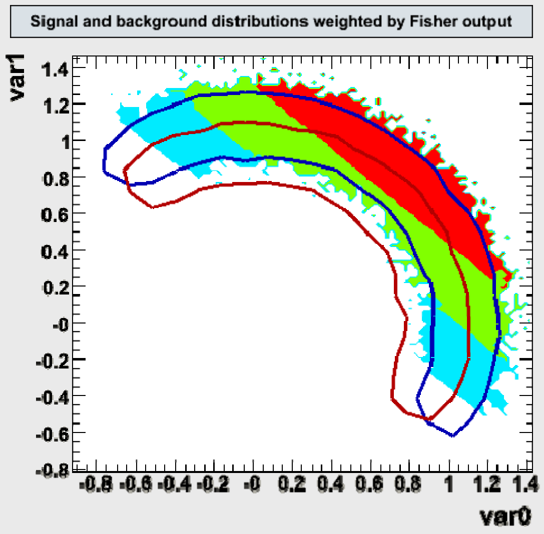
Linear correlations
(same for signal and background)



Linear correlations
(opposite for signal and background)



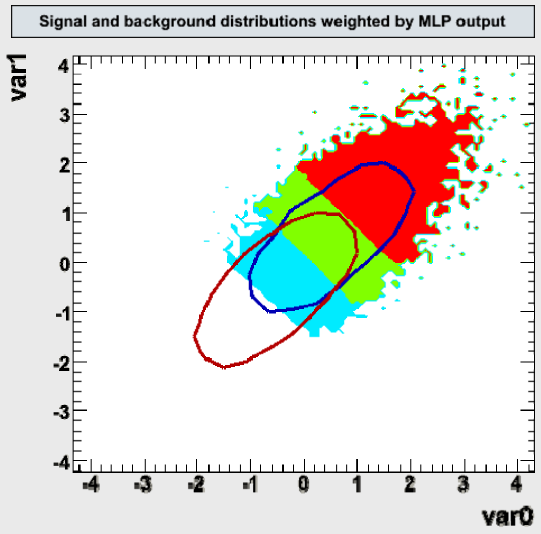
Circular correlations
(same for signal and background)



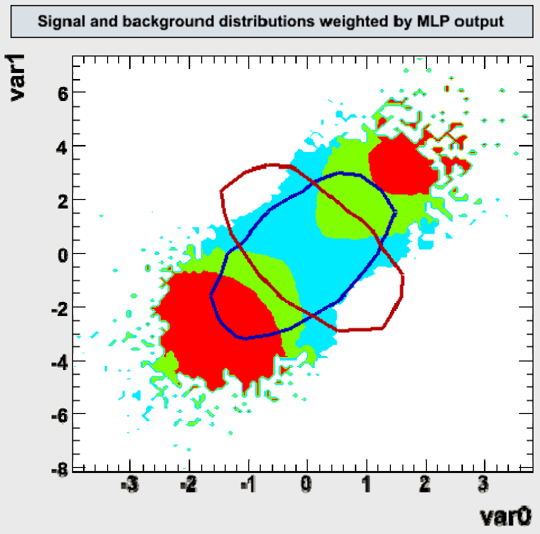
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

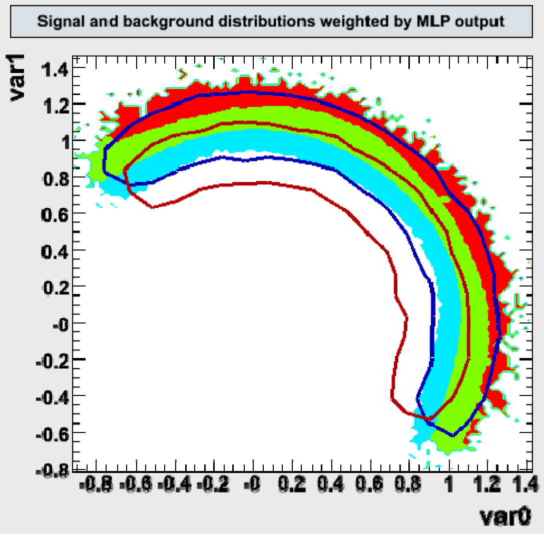
Linear correlations
(same for signal and background)



Linear correlations
(opposite for signal and background)



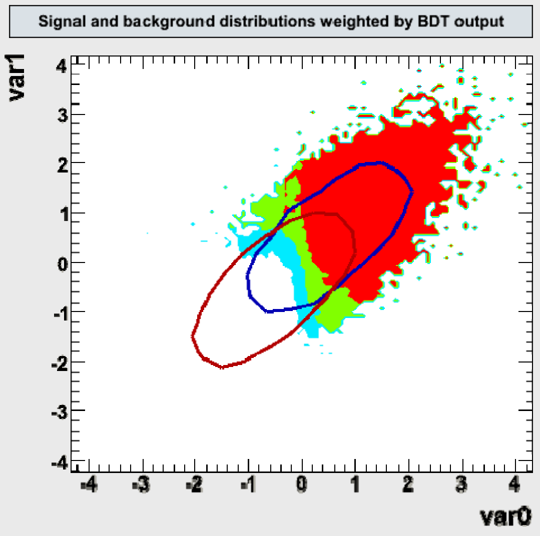
Circular correlations
(same for signal and background)

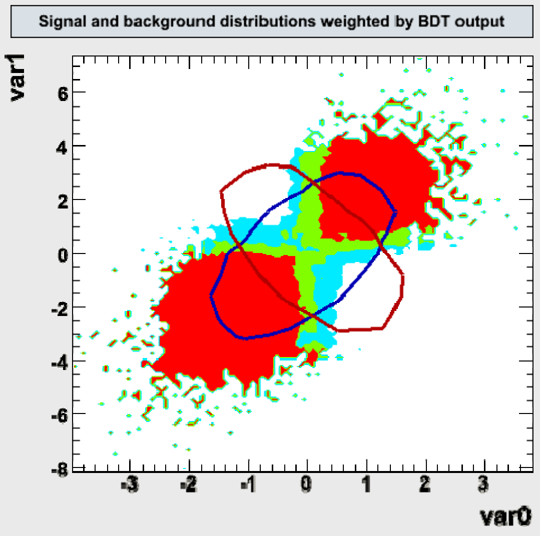


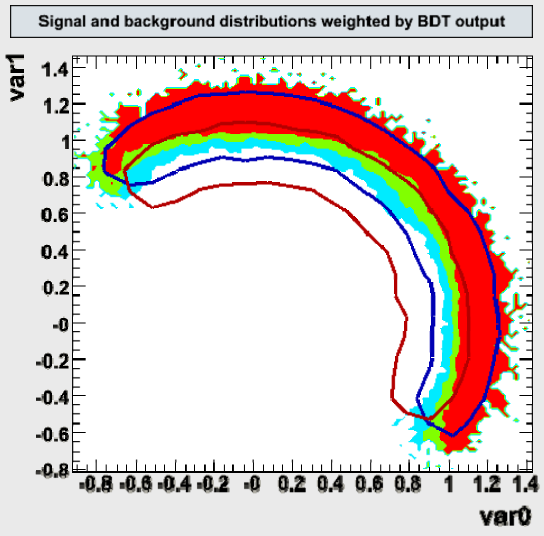
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

Linear correlations (same for signal and background)	Linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)



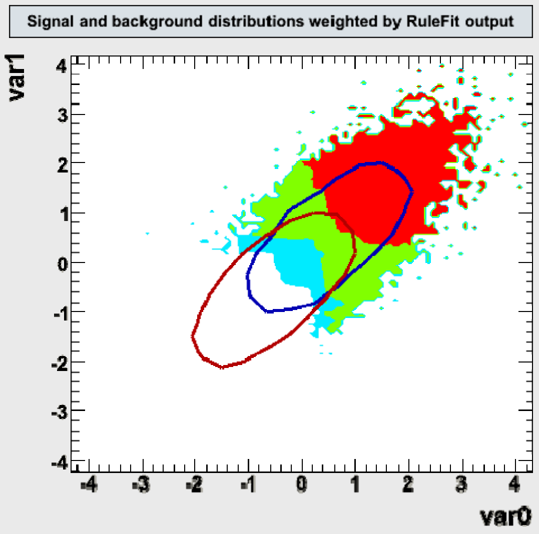


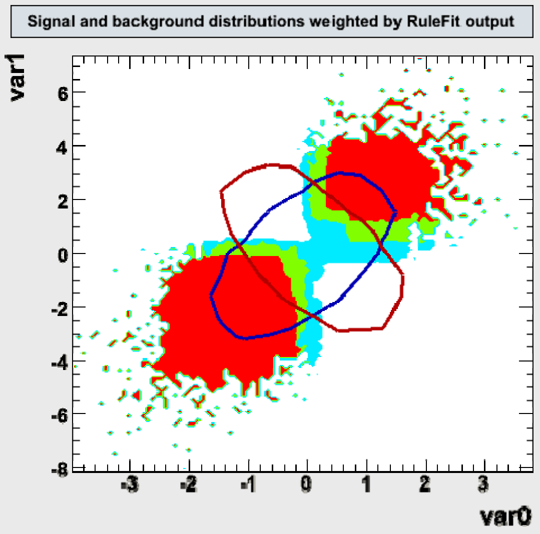


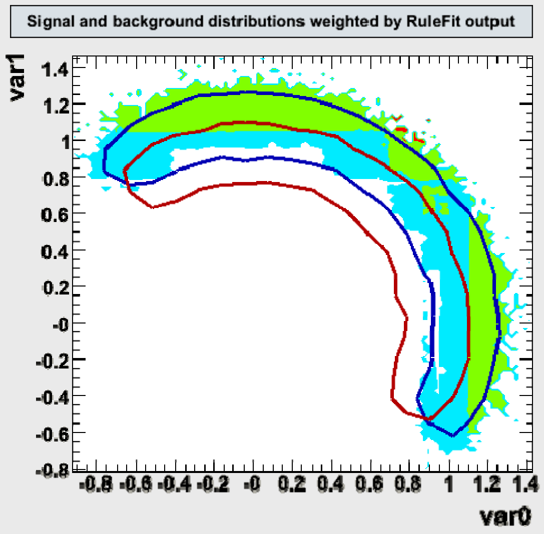
Weight Variables by Classifier Performance

- How well do the classifier resolve the various correlation patterns ?

Linear correlations (same for signal and background)	Linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)

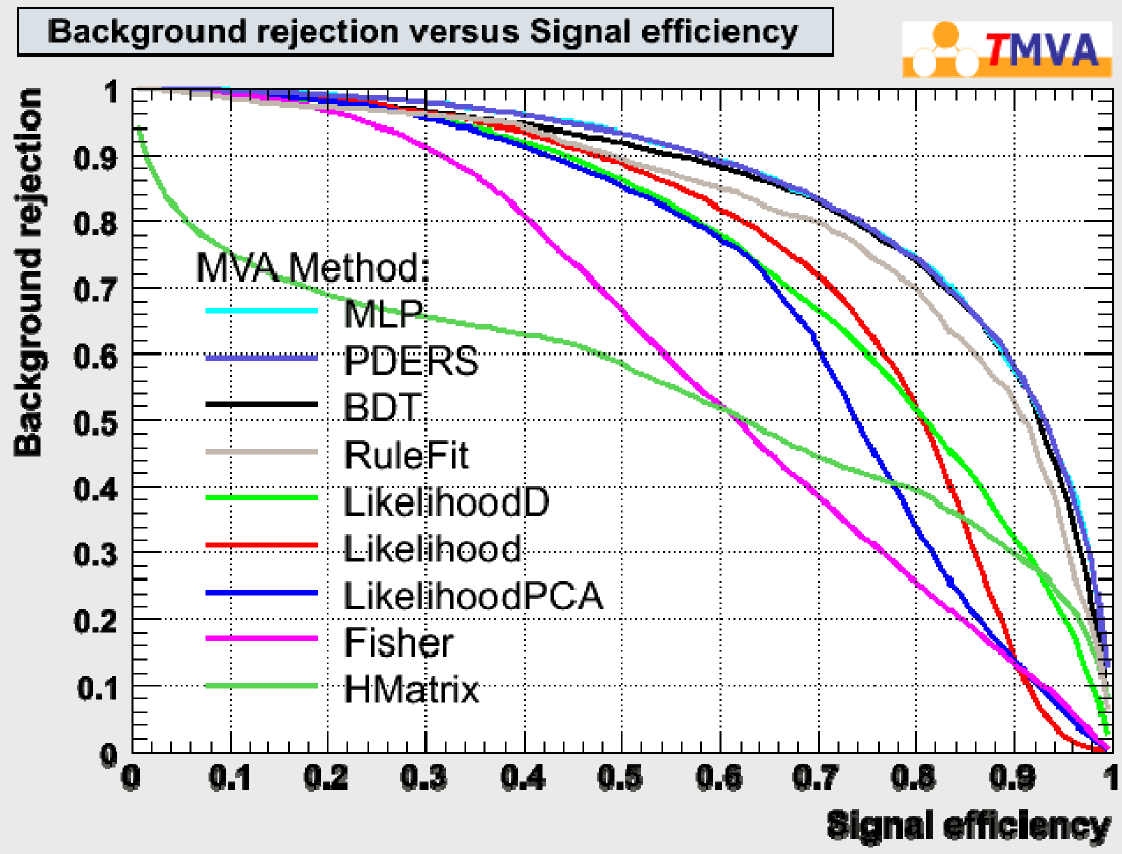






Final Classifier Performance

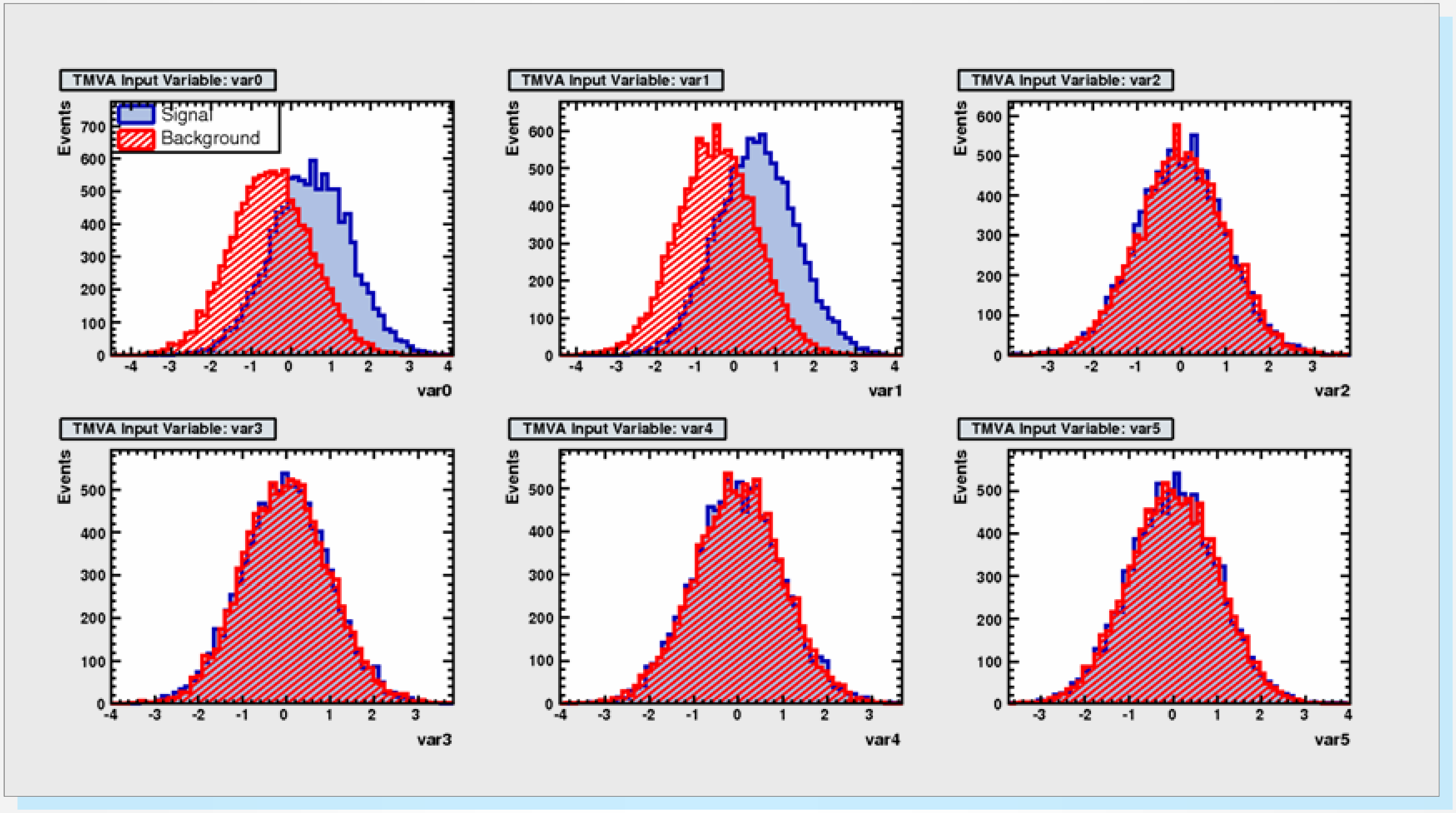
- Background rejection versus signal efficiency curve:



Circular Example

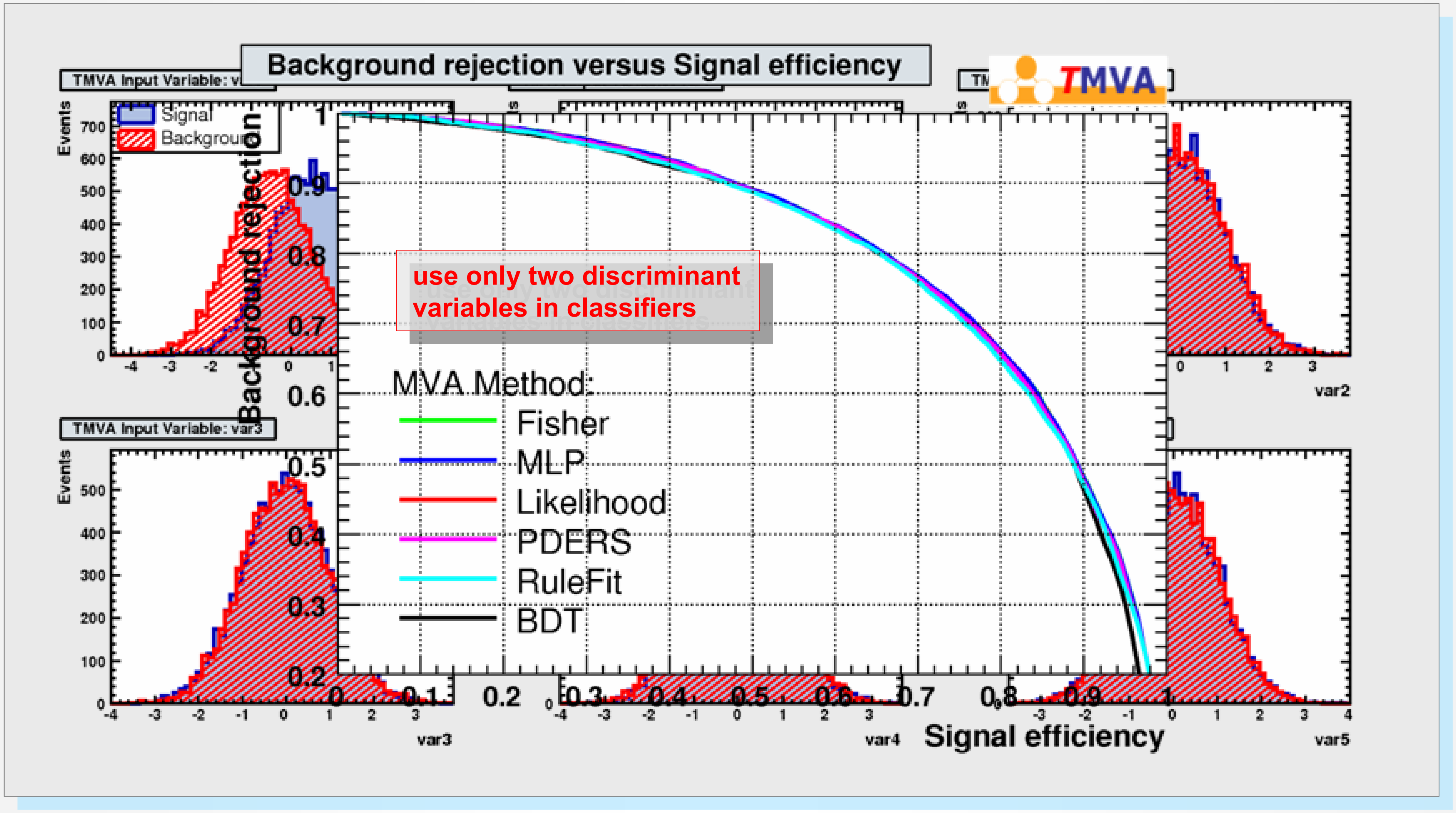
Stability with Respect to Irrelevant Variables

- Toy example with 2 discriminating and 4 non-discriminating variables ?



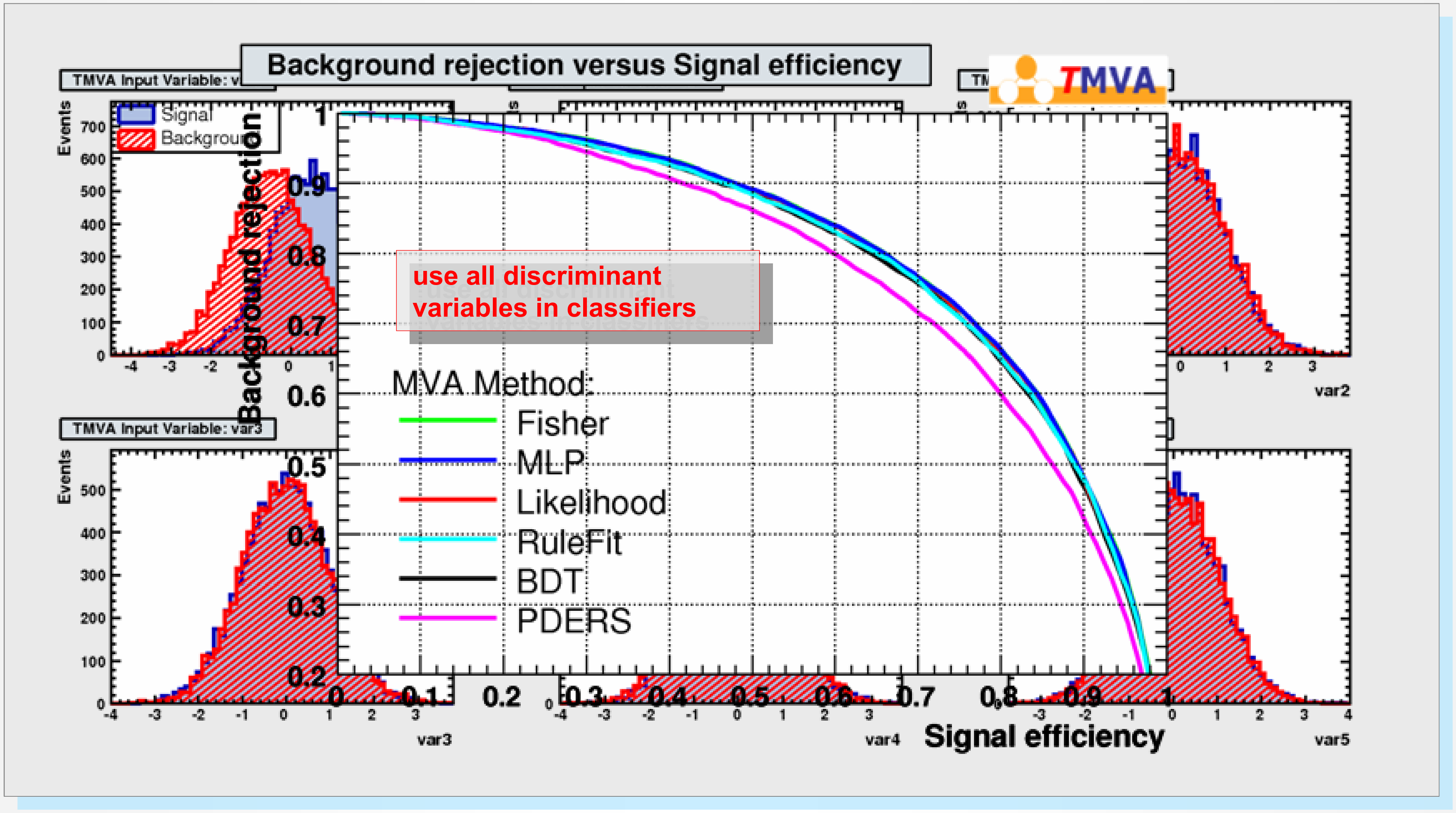
Stability with Respect to Irrelevant Variables

- Toy example with 2 discriminating and 4 non-discriminating variables ?



Stability with Respect to Irrelevant Variables

- Toy example with 2 discriminating and 4 non-discriminating variables ?



S u m m a r y & P l a n s

Summary of the Classifiers and their Properties

Criteria		Classifiers							
		Rectangular Cuts	Projective Likelihood	Multi-D PDERS	H-Matrix	Fisher	MLP	BDT	RuleFit
Perfor-mance	no / linear correlations	😐	🙂	🙂	😐	🙂	🙂	😐	🙂
	nonlinear correlations	😐	☹️	🙂	☹️	☹️	🙂	🙂	😐
Speed	Training	☹️	🙂	🙂	🙂	🙂	😐	☹️	😐
	Response	🙂	🙂	☹️	🙂	🙂	🙂	😐	😐
Robust-ness	Overtraining	🙂	😐	😐	🙂	🙂	☹️	☹️	😐
	Weak input variables	🙂	🙂	☹️	🙂	🙂	😐	😐	😐
Course of dimensionality		☹️	🙂	☹️	🙂	🙂	😐	😐	😐
Clarity		🙂	🙂	😐	🙂	🙂	☹️	☹️	☹️

Summary & Plans

- *T*MVA unifies highly customizable and performing multivariate classification algorithms in a single user-friendly framework
- This ensures most objective classifier comparisons, and simplifies their use
- *T*MVA is available on tmva.sf.net, and in ROOT (≥ 5.11/03)
- A typical *T*MVA analysis requires user interaction with a *Factory* (for the classifier training) and a *Reader* (for the classifier application)
- ROOT Macros are provided for the display of the evaluation results
- Forthcoming:
  - Imminent: *T*MVA version 3.6.0 with new features (together with a detailed Users Guide)
  - Support Vector Machine (M. Wolter & A. Zemla)
  - Bayesian classifiers
  - Committee method

Copyrights & Credits

- *T*MVA is open source !
- Use & redistribution of source permitted according to terms in BSD license
- Several similar data mining efforts with rising importance in most fields of science and industry
- Important for HEP:
  - Parallelised MVA training and evaluation pioneered by *Cornelius* package (BABAR)
  - Also frequently used: *StatPatternRecognition* package by I. Narsky
  - Many implementations of individual classifiers exist

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. We thank in particular the CERN Summer students Matt Jachowski (Stanford) for the implementation of TMVA's new MLP neural network, and Yair Mahalalel (Tel Aviv) for a significant improvement of PDERS. We are grateful to Doug Applegate, Kregg Arms, Ren\'e Brun and the ROOT team, Tancredi Carli, Elzbieta Richter-Was, Vincent Tisserand and Marcin Wolter for helpful conversations.