



Machine Learning Techniques for HEP Data Analysis with **TMVA**

Andreas Hoecker^(*) (CERN)

Top Workshop, LPSC-Grenoble, Oct 18–20, 2007

^(*) On behalf of the author team: A. Hoecker, P. Speckmayer, J. Stelzer, F. Tegenfeldt, H. Voss, K. Voss

And the contributors: A. Christov, Or Cohen, S. Henrot-Versillé, M. Jachowski,
A. Krasznahorkay Jr., Y. Mahalalel, R. Ospanov, X. Prudent, M. Wolter, A. Zemla

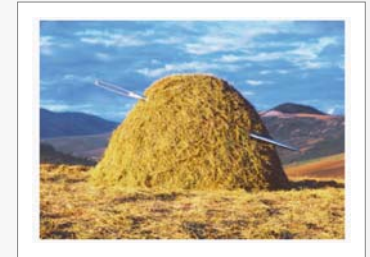
See acknowledgments on page 43



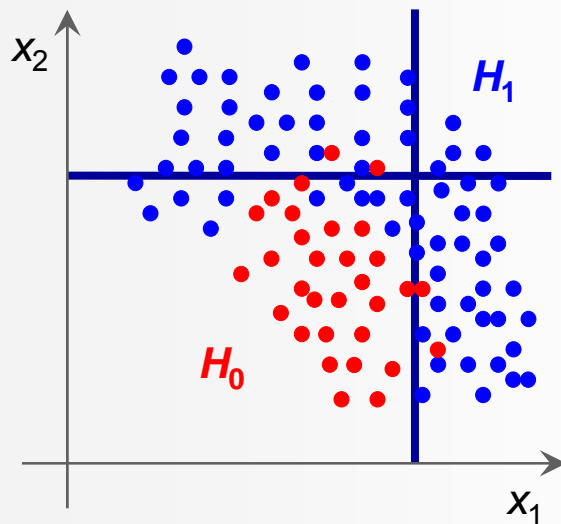
On the web: <http://tmva.sf.net/> (home), <https://wiki.cern.ch/wiki/bin/view/TMVA/WebHome> (tutorial)

Event Classification

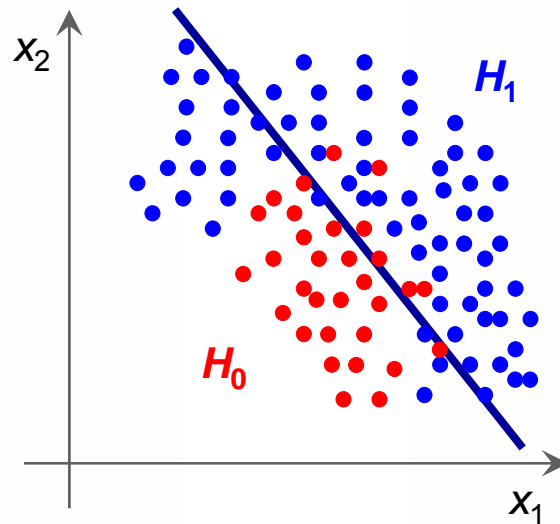
- Suppose data sample with two types of events: H_0 , H_1
 - We have found discriminating input variables x_1, x_2, \dots
 - What decision boundary should we use to select events of type H_1 ?



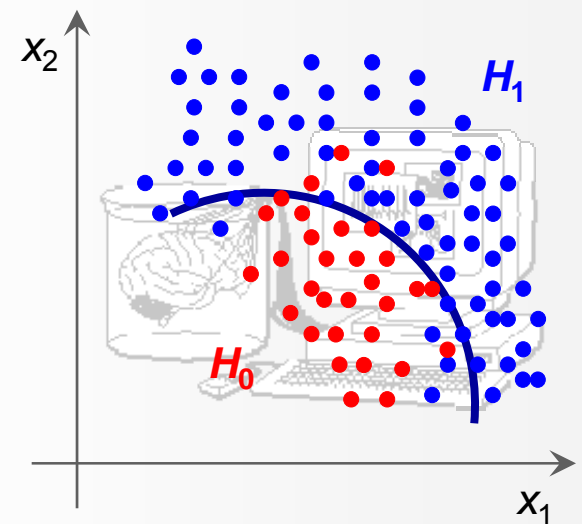
Rectangular cuts?



A linear boundary?



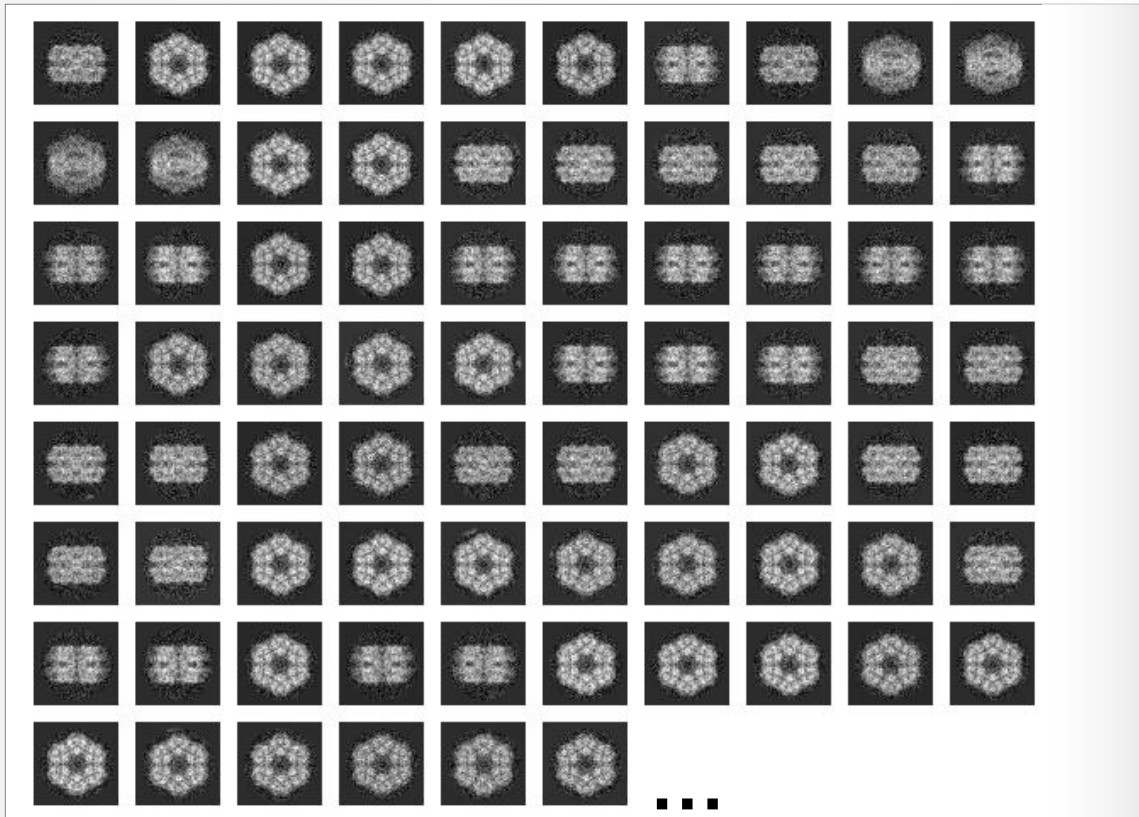
A nonlinear one?



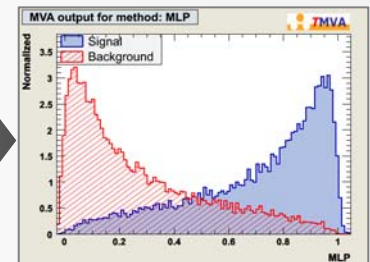
- How can we decide this in an optimal way ? → Let the machine learn it !

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



TMVA

What is **TMVA**

- The various classifiers have very different properties
 - Ideally, all should be tested for a given problem
 - Systematically choose the best performing and simplest classifier
 - Comparisons between classifiers improves the understanding and takes away mysticism
- **TMVA** — **Toolkit** for multivariate data analysis
 - Framework for *parallel training, testing, evaluation* and **application** of MV classifiers
 - Training events can have weights
 - A large number of linear, nonlinear, likelihood and rule-based classifiers implemented
 - The classifiers rank 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
 - Application to data classification using a **Reader** or standalone C++ classes

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
 - Tutorial TWiki <https://twiki.cern.ch/twiki/bin/view/TMVA/WebHome>

- **Active project → fast response time on feature requests**
 - Currently 6 main developers, and 27 registered contributors at SF
 - >1900 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**

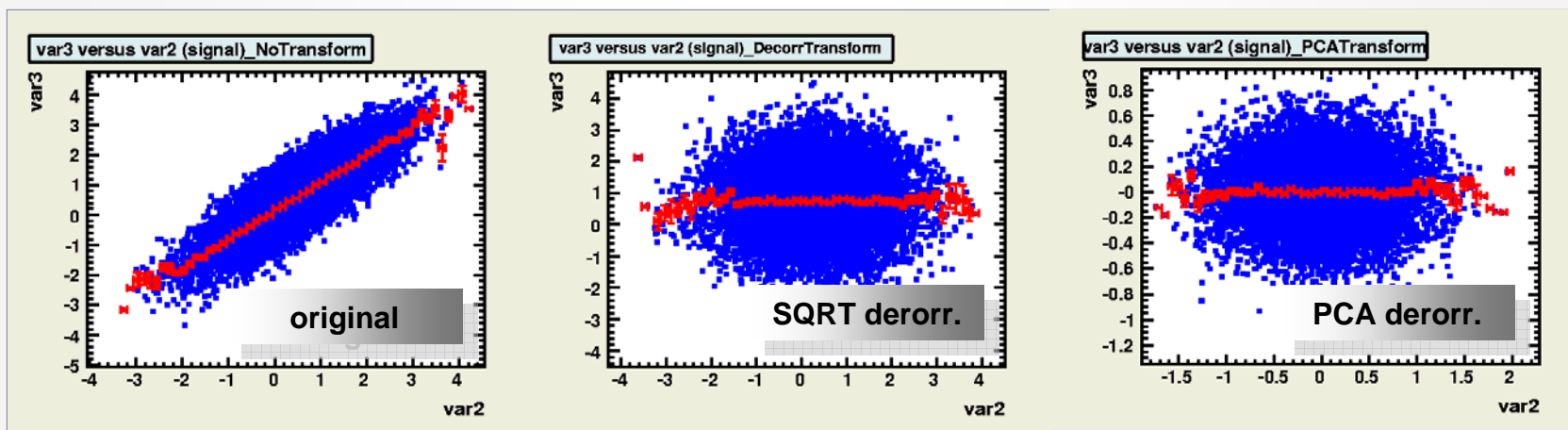
The *TMVA* Classifiers

► Currently implemented classifiers :

- ▶ Rectangular cut optimisation
- ▶ Projective and multidimensional likelihood estimator
- ▶ k-Nearest Neighbor algorithm
- ▶ Fisher and H-Matrix discriminants
- ▶ Function discriminant
- ▶ Artificial neural networks (3 *multilayer perceptron* implementations)
- ▶ Boosted/bagged decision trees with automatic node pruning
- ▶ RuleFit
- ▶ Support Vector Machine

Data Preprocessing: Decorrelation

- Commonly realised for all methods in **TMVA** (centrally in **DataSet** class)
- Removal of linear correlations by rotating input variables
 - ➔ Determine *square-root* C' of covariance matrix C , i.e., $C = C' C'$
 - ➔ Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$
- Various ways to choose basis for decorrelation (also implemented PCA)



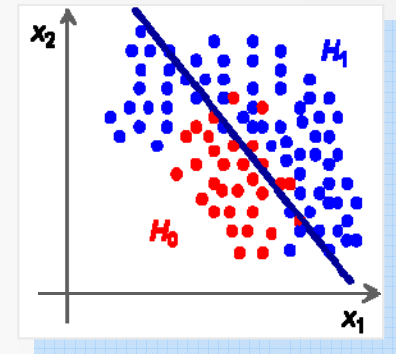
Data Preprocessing: Decorrelation

- Commonly realised for all methods in *TMVA* (centrally in `DataSet` class)
- Removal of linear correlations by rotating input variables
 - ➔ Determine *square-root* C' of covariance matrix C , i.e., $C = C' C'$
 - ➔ Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$
- Various ways to choose basis for decorrelation (also implemented PCA)
- Note that decorrelation is only complete, if
 - Correlations are linear
 - Input variables are Gaussian distributed
 - ➔ Not very accurate conjecture in general

Rectangular Cut Optimisation

- Simplest method: cut in rectangular variable volume

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



- Technical challenge: **how to find optimal cuts ?**
 - MINUIT fails due to non-unique solution space
 - **TMVA** uses: **Monte Carlo sampling**, **Genetic Algorithm**, **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

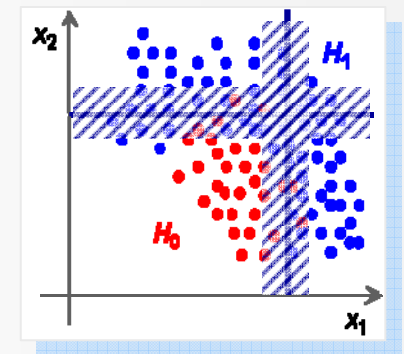
Likelihood ratio for event i_{event}

PDFs

discriminating variables

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

Species: signal, background types



PDE introduces fuzzy logic

- Ignores correlations between input variables
 - Optimal approach if correlations are zero (or linear \rightarrow decorrelation)
 - Otherwise: significant performance loss

PDE Approach: Estimating PDF Kernels

- Technical challenge: how to estimate the PDF shapes

➔ 3 ways:

parametric fitting (function)

Difficult to automate
for arbitrary PDFs

nonparametric fitting

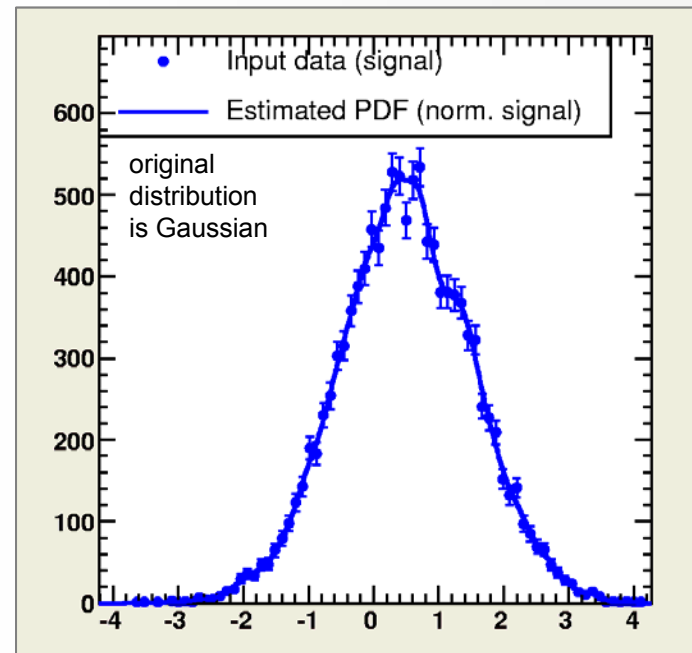
Easy to automate, can create
artefacts/suppress information

event counting

Automatic, unbiased,
but suboptimal

- We have chosen to implement nonparametric fitting in **TMVA**

- Binned shape interpolation using spline functions (orders: 1, 2, 3, 5)
- Unbinned adaptive kernel density estimation (KDE) with Gaussian smearing
- ➔ **TMVA** 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)

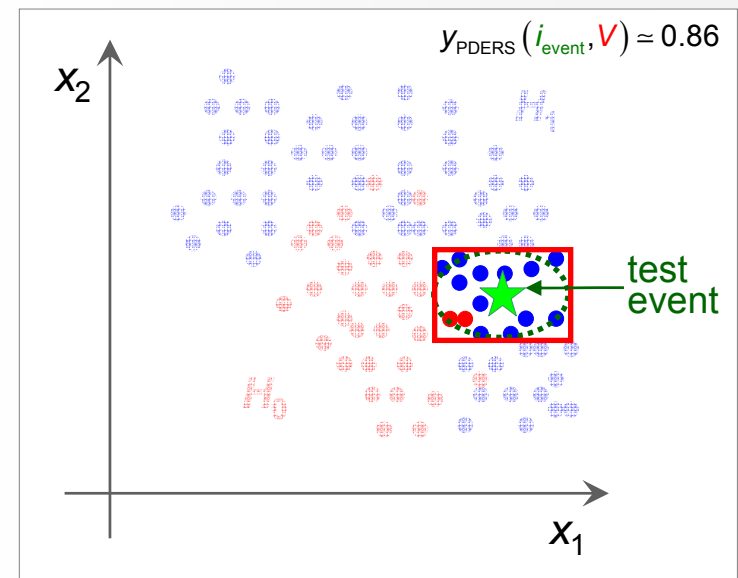
PDE-RS ratio
for event i_{event}

chosen
volume

#signal events in V

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

#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

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”

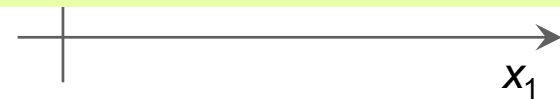
Carli-Koblitz, NIM
A501, 576 (2003)

New classifier: **k-Nearest Neighbor** – implemented by R. Ospanov (Texas U.):

Better than searching within a volume (fixed or floating), count adjacent reference events till statistically significant number reached

- ➔ Method intrinsically adaptive
- ➔ Very fast search with kd-tree event sorting

#background events in V

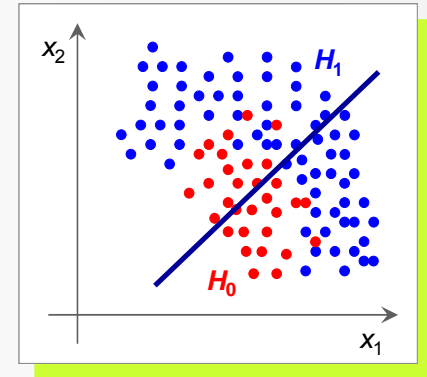


- Improve y_{PDEERS} estimate within V by using various N_{var} -D kernel estimators
- Enhance speed of event counting in volume by binary tree search

Fisher's 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 response couldn't be simpler:

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

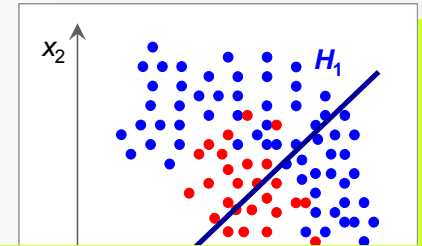
“Fisher coefficients”

- Compute Fisher coefficients from signal and background covariance matrices
- Fisher requires distinct sample means between signal and background
- Optimal classifier for linearly correlated Gaussian-distributed variables

Fisher's 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



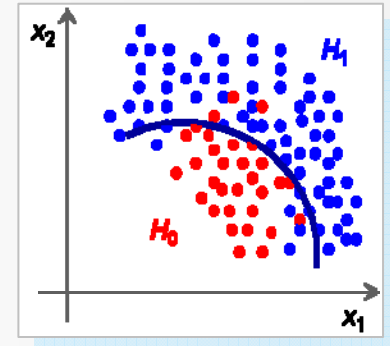
New classifier: **Function discriminant analysis (FDA)**

Fit any user-defined function of input variables requiring that signal events return $\rightarrow 1$ and background $\rightarrow 0$

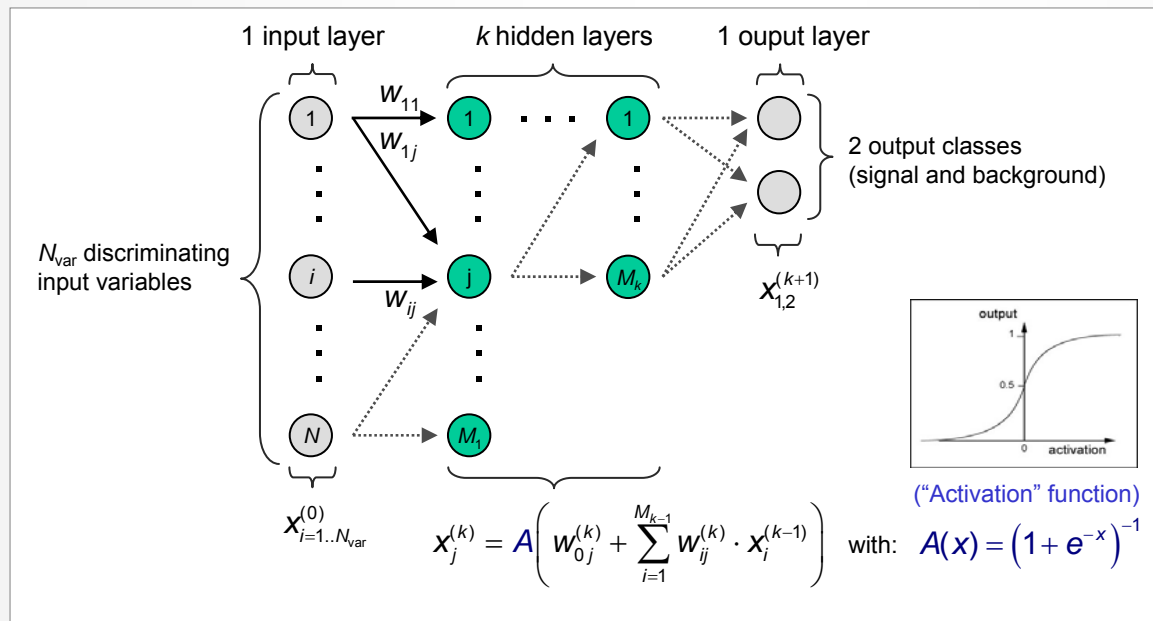
- ➔ Parameter fitting: Genetics Alg., MINUIT, MC and combinations
 - ➔ Easy reproduction of Fisher result, but can add nonlinearities
 - ➔ Very transparent discriminator
- Compute Fisher coefficients from signal and background covariance matrices
 - ➔ Fisher requires distinct sample means between signal and background
 - ➔ Optimal classifier for linearly correlated Gaussian-distributed variables

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



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

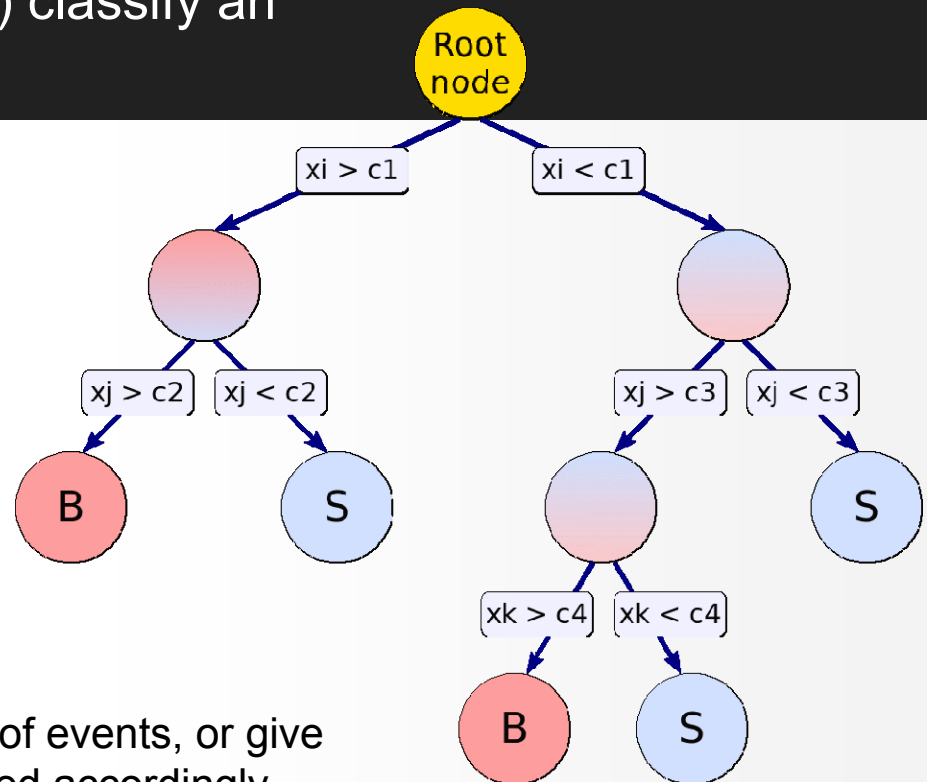
- Adjust weights (=training) using “back-propagation”

Decision Trees

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

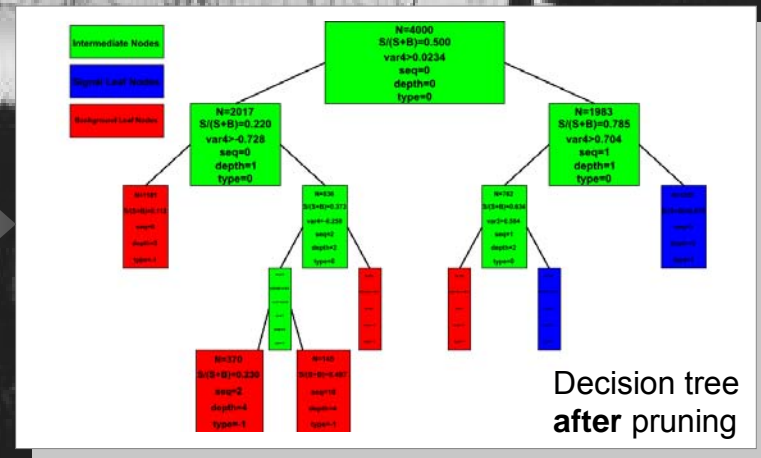
- Growing a decision tree:

- Start with Root node
- Split training sample according to cut on best variable at this node
- Splitting criterion: e.g., maximum “Gini-index”: $\text{purity} \times (1 - \text{purity})$
- 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



Decision Trees

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



- Bottom-up “pruning” of a decision tree

- Remove statistically insignificant nodes to reduce tree overtraining → automatic in *TMVA*

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 → **improves stability !**

Predictive Learning via Rule Ensembles (RuleFit)

- Following RuleFit approach by [Friedman-Popescu](#)

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

- Model is linear combination of *rules*, where a rule is a sequence of cuts

RuleFit classifier

rules (cut sequence
 $\rightarrow r_m=1$ if all cuts satisfied, $=0$ otherwise)

normalised discriminating event variables

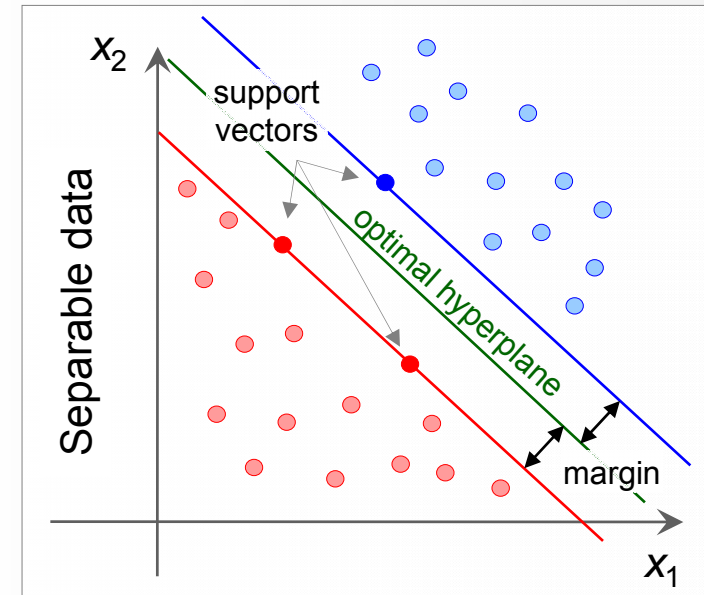
$$y_{\text{RF}}(\vec{x}) = a_0 + \underbrace{\sum_{m=1}^{M_R} a_m r_m(\vec{x})}_{\text{Sum of rules}} + \underbrace{\sum_{k=1}^{n_R} b_k \hat{x}_k}_{\text{Linear Fisher term}}$$

- The problem to solve is
 - Create rule ensemble: use forest of decision trees
 - Fit coefficients a_m, b_k : gradient direct regularization minimising *Risk* (Friedman et al.)
- Pruning removes topologically equal rules” (same variables in cut sequence)

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=00011110₂.

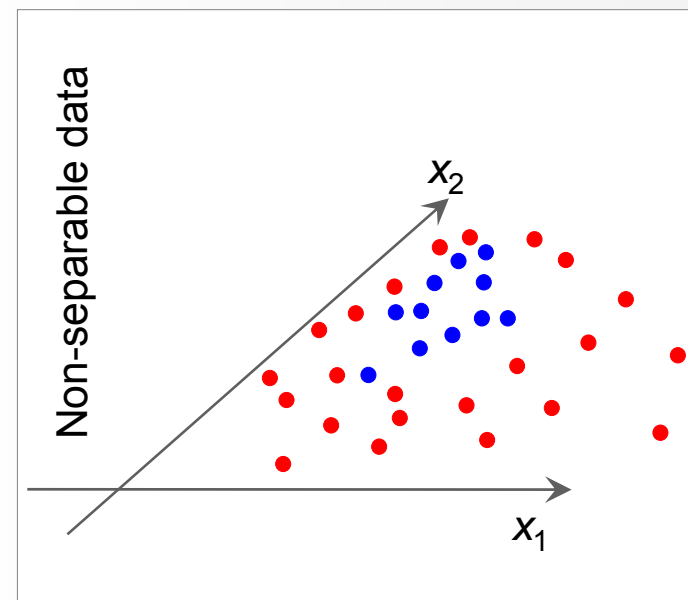
Support Vector Machine (SVM)

- Find hyperplane that best separates signal from background
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - Linear decision boundary
 - If data non-separable add *misclassification cost* parameter to minimisation function



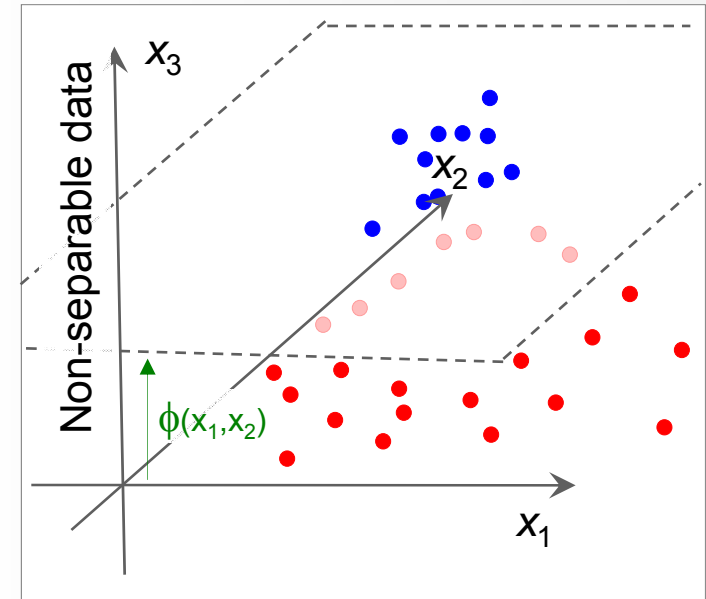
Support Vector Machine (SVM)

- Find hyperplane that best separates signal from background
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - Linear decision boundary
 - If data non-separable add *misclassification cost* parameter to minimisation function



Support Vector Machine (SVM)

- Find hyperplane that best separates signal from background
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - Linear decision boundary
 - If data non-separable add *misclassification cost* parameter to minimisation function



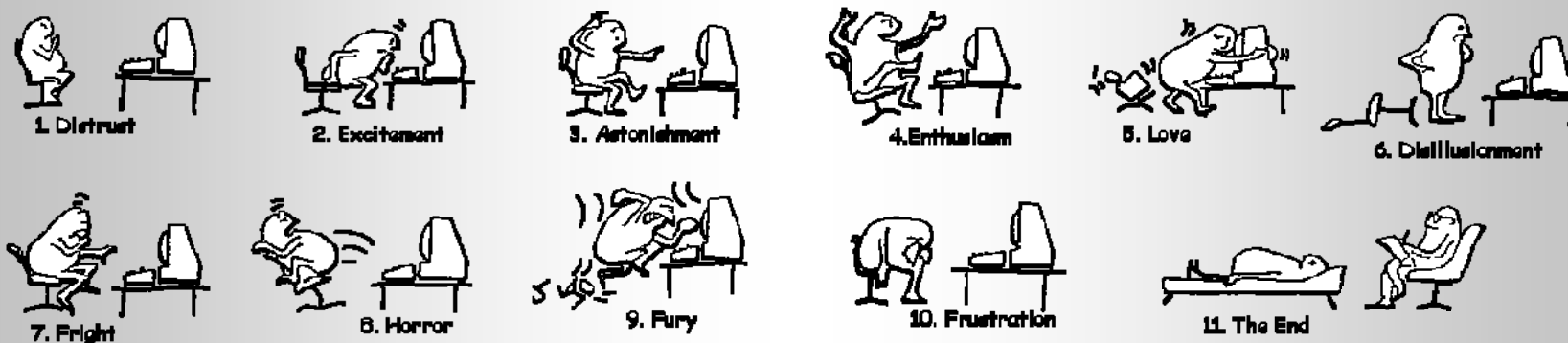
- Non-linear cases:
 - Transform variables into higher dimensional space where again a linear boundary (hyperplane) can separate the data
 - Explicit transformation form not required: use Kernel Functions to approximate scalar products between transformed vectors in the higher dimensional space
 - Choose Kernel and fit the hyperplane using the linear techniques developed above
 - ➔ Available Kernels: **Gaussian, Polynomial, Sigmoid**

Using *TMVA*

A typical *TMVA* analysis consists of two main steps:

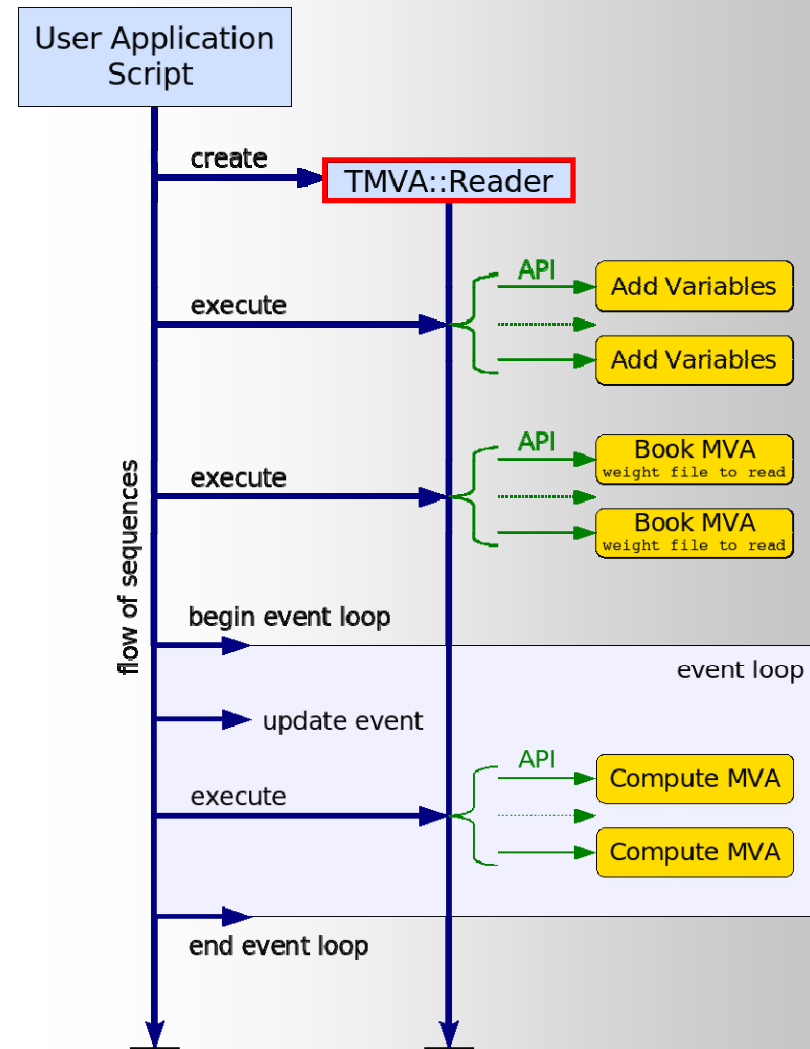
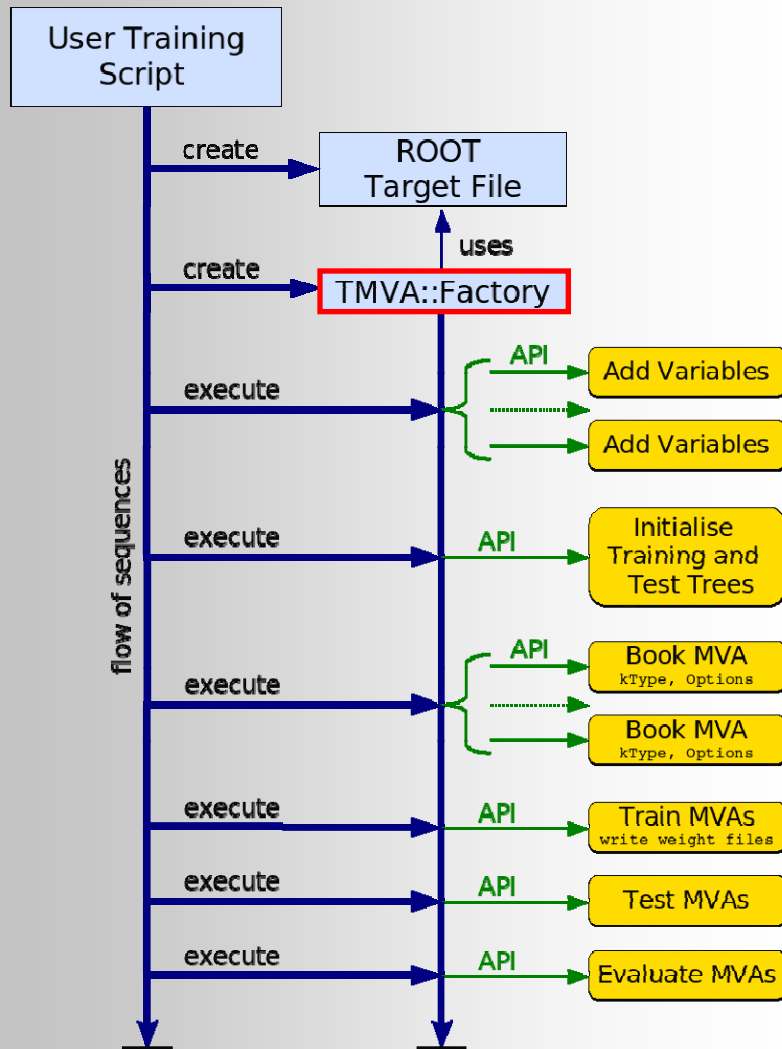
1. *Training phase*: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. *Application phase*: using selected trained classifiers to classify unknown data samples

► Illustration of these steps with toy data samples



[→ *TMVA* tutorial](#)

Code Flow for *Training* and *Application* Phases



→ [TMVA tutorial](#)

A Simple Example for *Training*

```
void TMVAnalysis( )
```

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

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

← create *Factory*

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

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

← give training/test trees

```
  factory->AddVariable("var1+var2", 'F');  
  factory->AddVariable("var1-var2", 'F');  
  factory->AddVariable("var3", 'F');  
  factory->AddVariable("var4", 'F');
```

← register input variables

```
  factory->PrepareTrainingAndTestTree("", "NSigTrain=3000:NBkgTrain=3000:SplitMode=Random:!V" );
```

```
  factory->BookMethod( TMVA::Types::kLikelihood, "Likelihood",  
                     "!V:!TransformOutput:Spline=2:NSmooth=5:NAvEvtPerBin=50" );
```

← select MVA
methods

```
  factory->BookMethod( TMVA::Types::kMLP, "MLP", "!V:NCycles=200:HiddenLayers=N+1,N:TestRate=5" );
```

```
  factory->TrainAllMethods();  
  factory->TestAllMethods();  
  factory->EvaluateAllMethods();
```

← train, test and evaluate

```
  outputFile->Close();  
  delete factory;
```

```
}
```

→ [TMVA tutorial](#)

A Simple Example for an *Application*

```
void TMVApplication( )
```

```
{
```

```
TMVA::Reader *reader = new TMVA::Reader("!Color");
```

← create *Reader*

```
Float_t var1, var2, var3, var4;  
reader->AddVariable( "var1+var2", &var1 );  
reader->AddVariable( "var1-var2", &var2 );  
reader->AddVariable( "var3", &var3 );  
reader->AddVariable( "var4", &var4 );
```

← register the variables

```
reader->BookMVA( "MLP classifier", "weights/MVAnalysis_MLP.weights.txt" );
```

← book classifier(s)

```
TFile *input = TFile::Open("tmva_example.root");  
TTree* theTree = (TTree*)input->Get("TreeS");
```

```
// ... set branch addresses for user TTree  
for (Long64_t ievt=3000; ievt<theTree->GetEntries();ievt++) {  
    theTree->GetEntry(ievt);
```

← prepare event loop

```
var1 = userVar1 + userVar2;  
var2 = userVar1 - userVar2;  
var3 = userVar3;  
var4 = userVar4;
```

← compute input variables

```
Double_t out = reader->EvaluateMVA( "MLP classifier" );
```

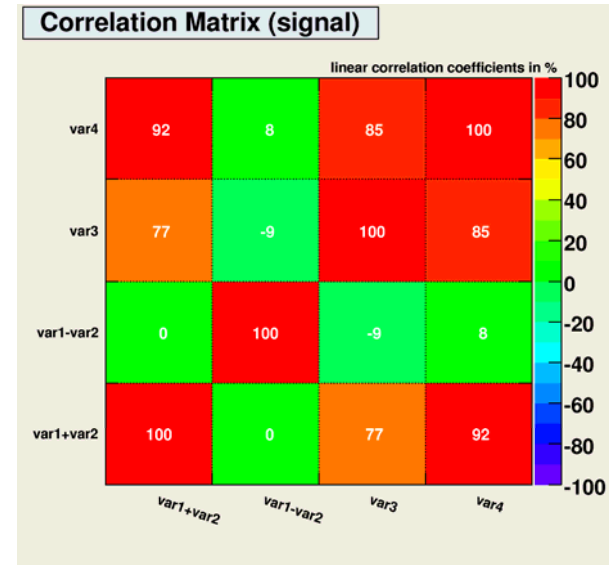
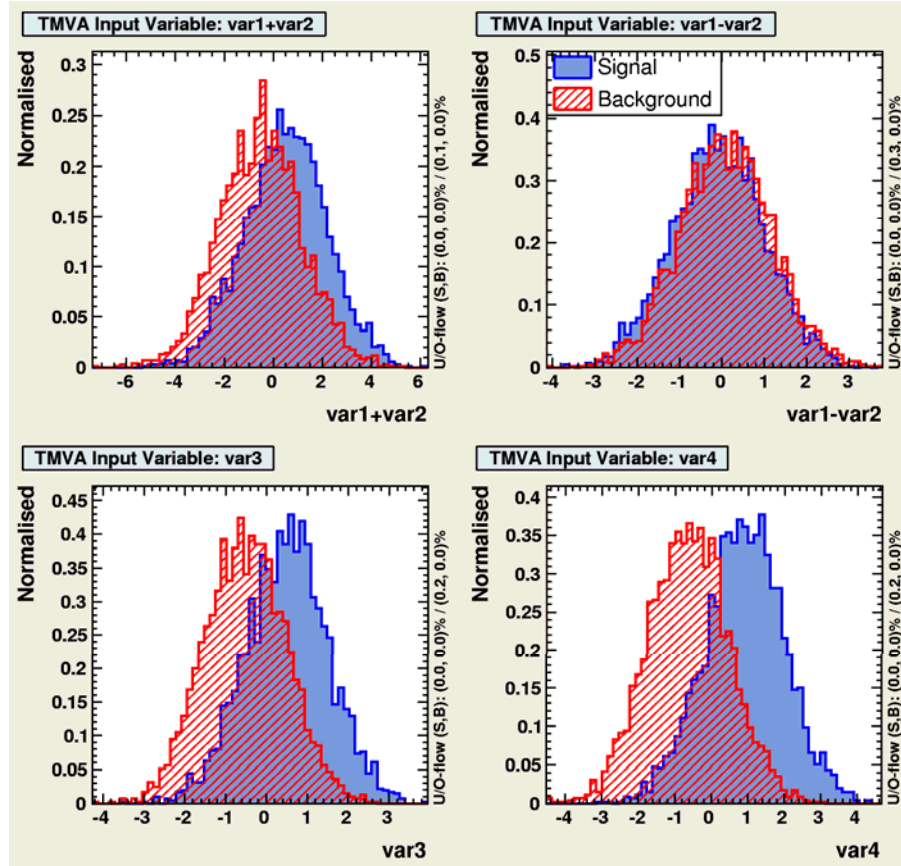
← calculate classifier output

```
    // do something with it ...  
}  
delete reader;  
}
```

→ [TMVA tutorial](#)

A Toy Example (idealized)

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

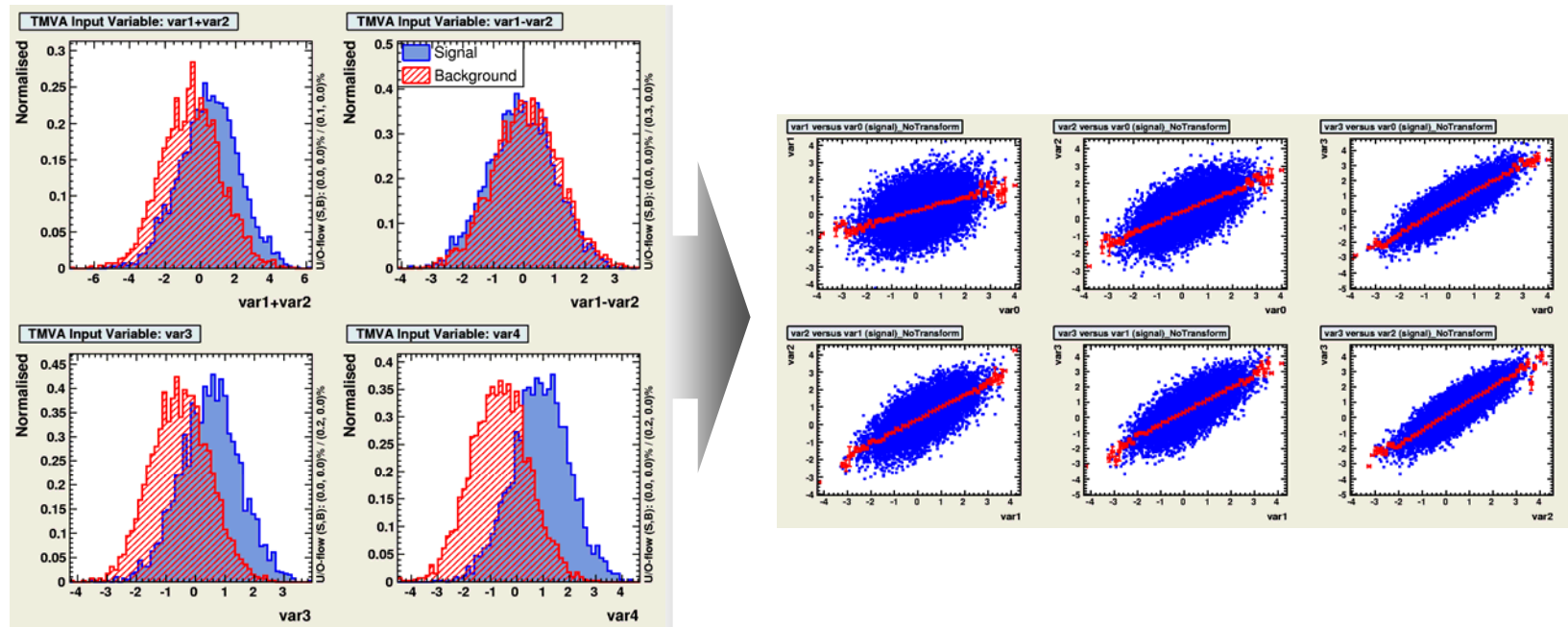


Rank : Variable : Separation

- 1 : var4 : 0.606
- 2 : var1+var2 : 0.182
- 3 : var3 : 0.173
- 4 : var1-var2 : 0.014

Preprocessing the Input Variables

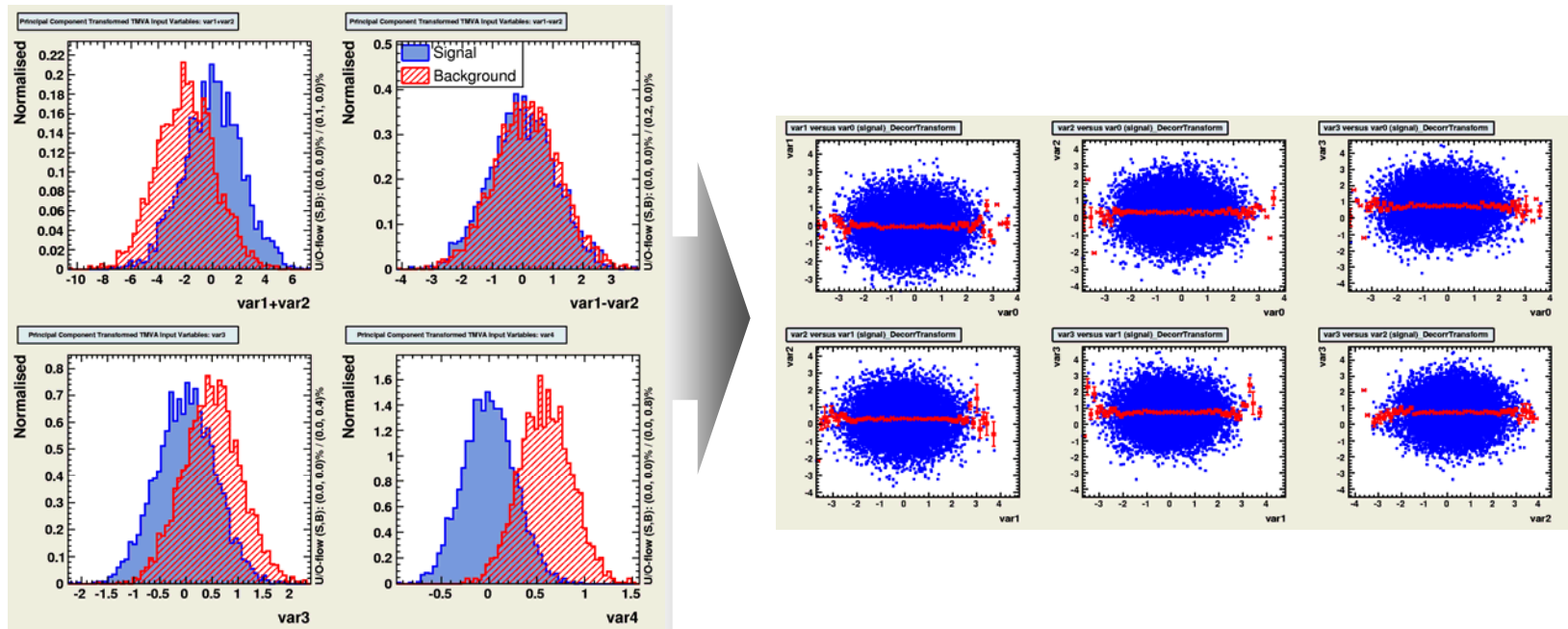
- Decorrelation of variables before training is useful for *this* example



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

Preprocessing the Input Variables

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

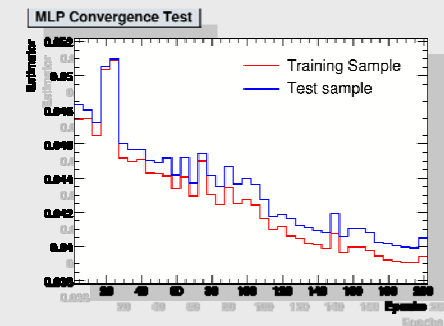
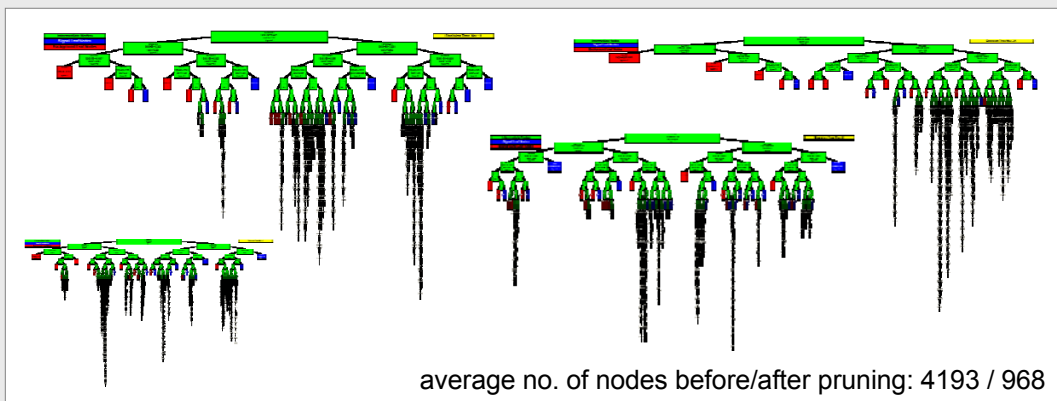
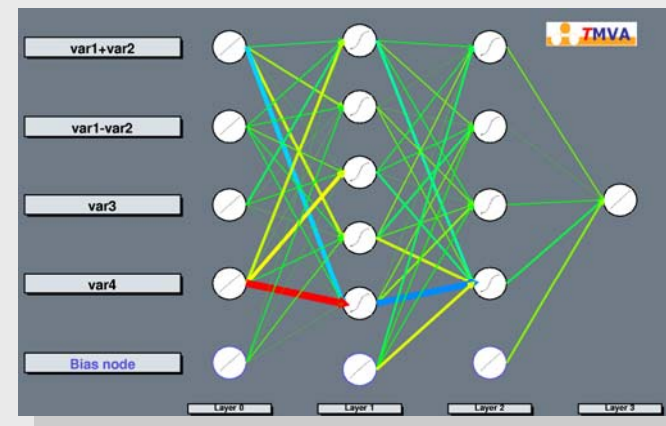
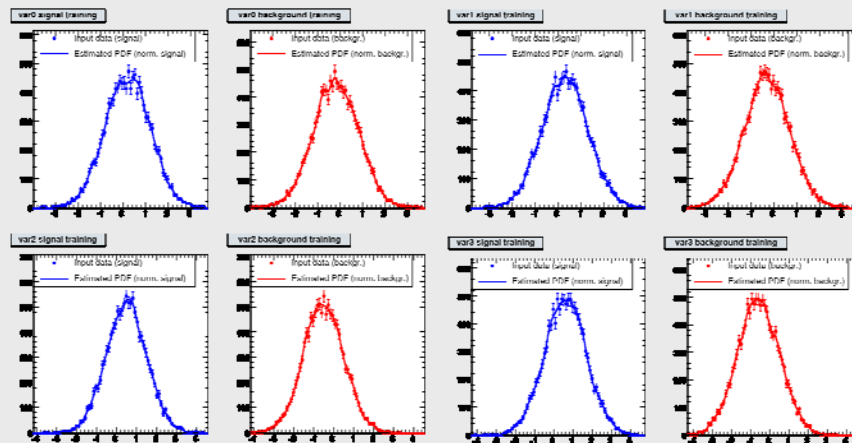


- Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

Validating the Classifier Training

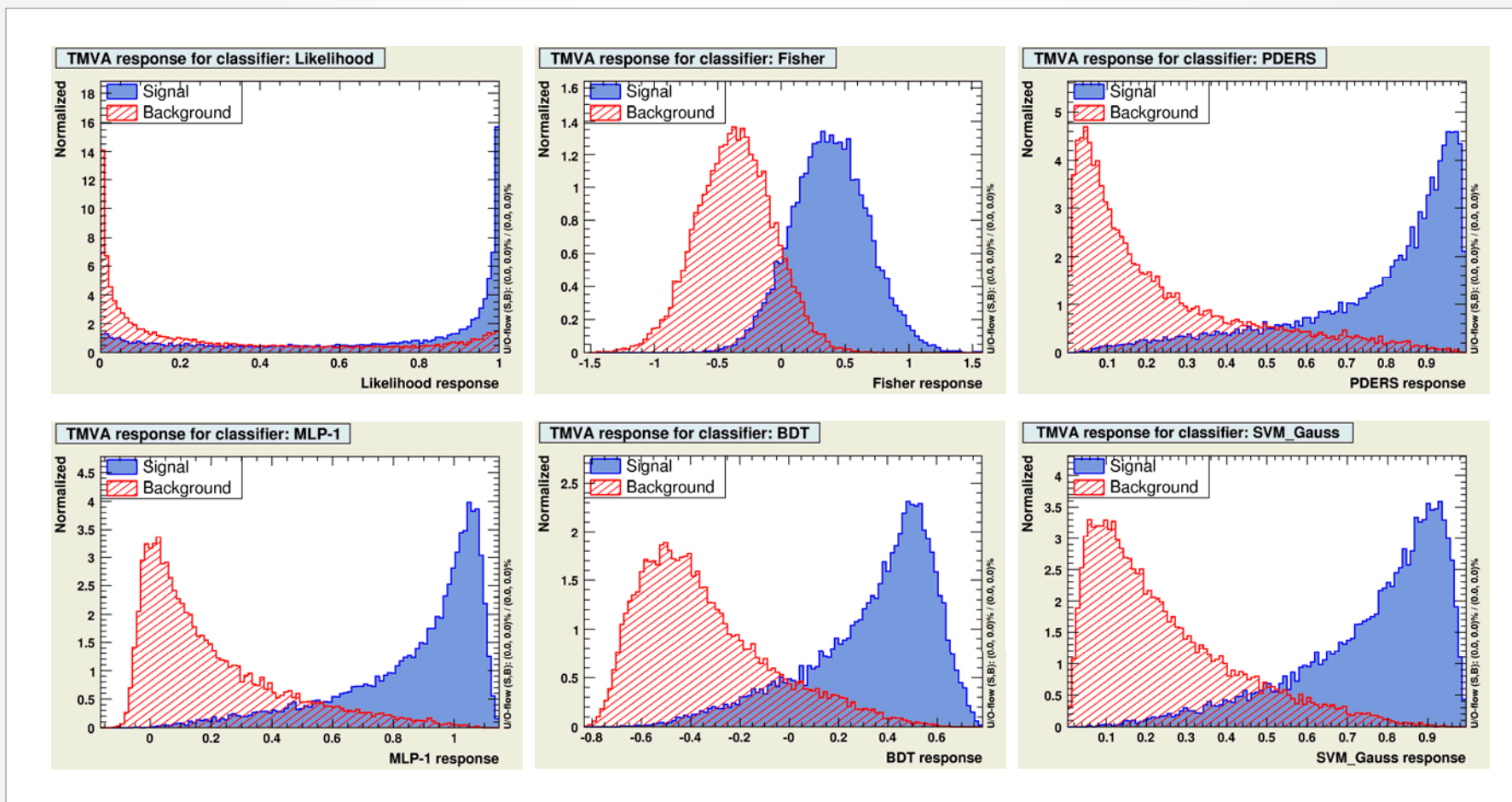
- Projective likelihood PDFs, MLP training, BDTs, ...

TMVA Plotting Macros	
(1a)	Input Variables
(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 Output Distributions for Training and Test Samples
(4c)	Classifier Probability Distributions
(4d)	Classifier Rarity Distributions
(5a)	Classifier Cut Efficiencies
(6)	Classifier Background Rejection vs Signal Efficiency (ROC curve)
(7a)	[Likelihood Reference Distributions]
(7b)	[Network Architecture]
(7c)	[Network Convergence Test]
(8)	[Decision Tree (T)]
(9)	PDFs of Classifiers
(10)	Rule Ensemble Importance Plots
(11)	Out



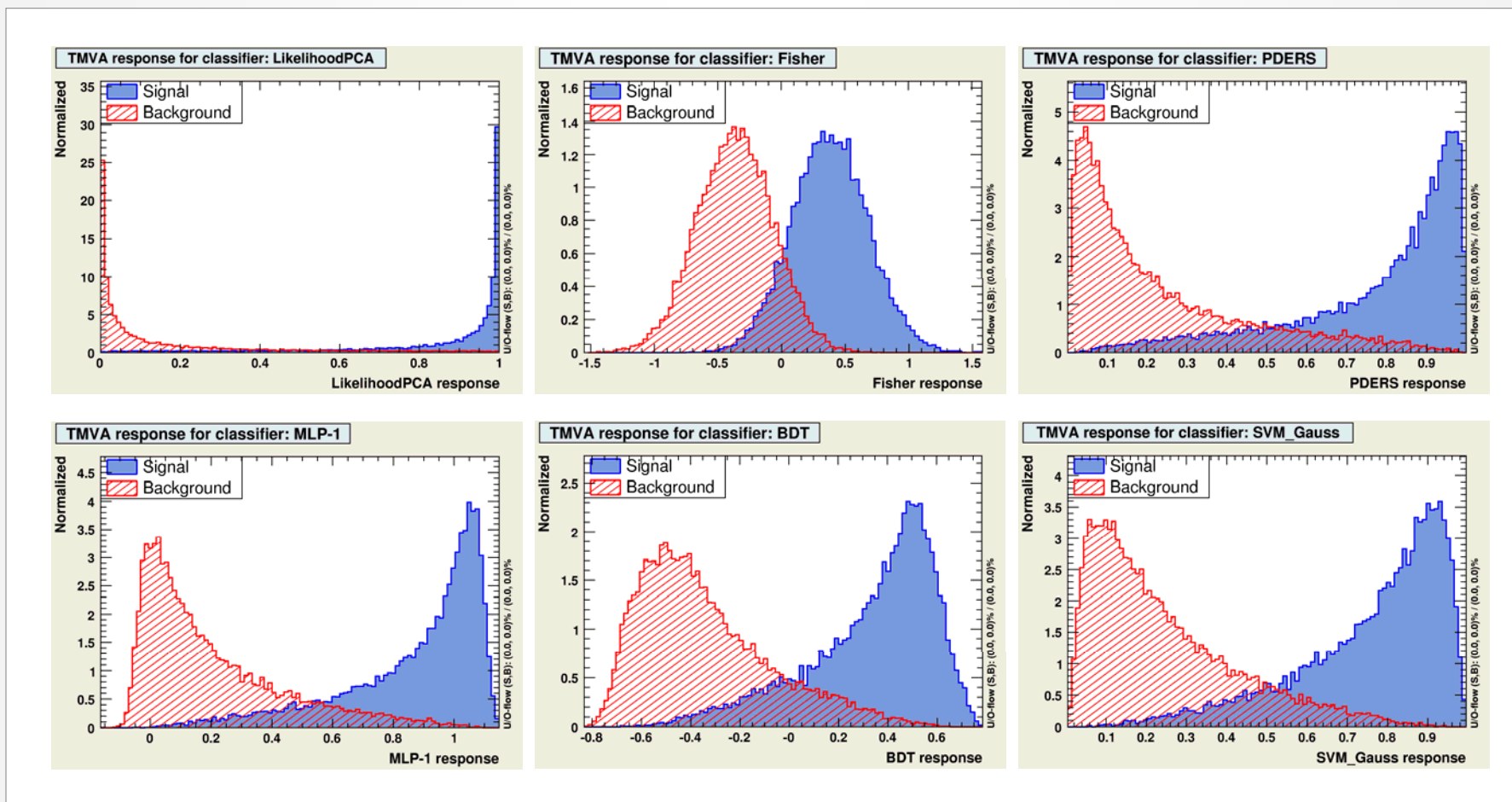
Testing the Classifiers

- Classifier output distributions for independent test sample:



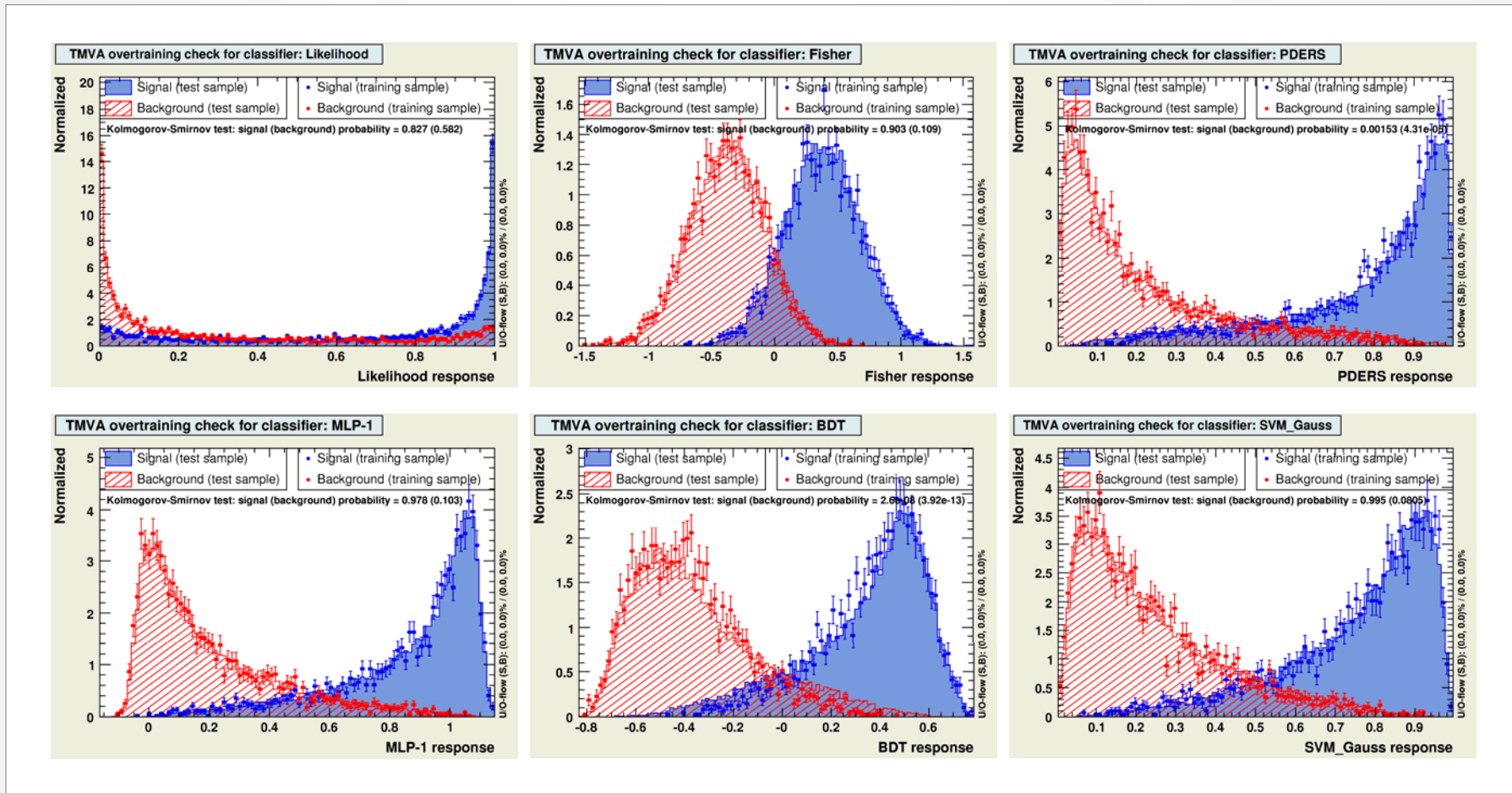
Testing the Classifiers

- Classifier output distributions for independent test sample:




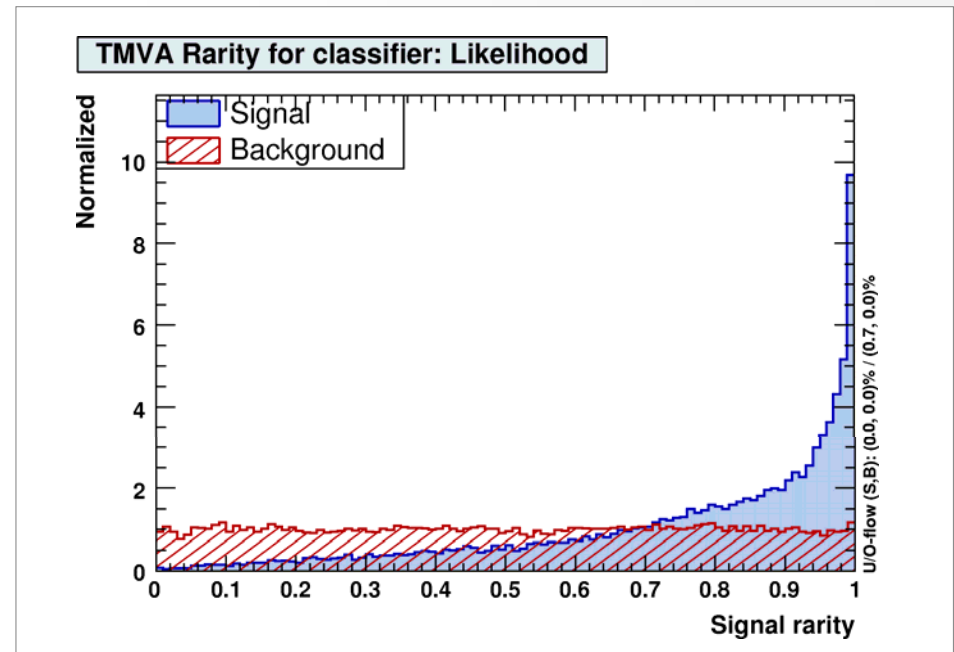
Testing the Classifiers

- Classifier output distributions for independent test sample:




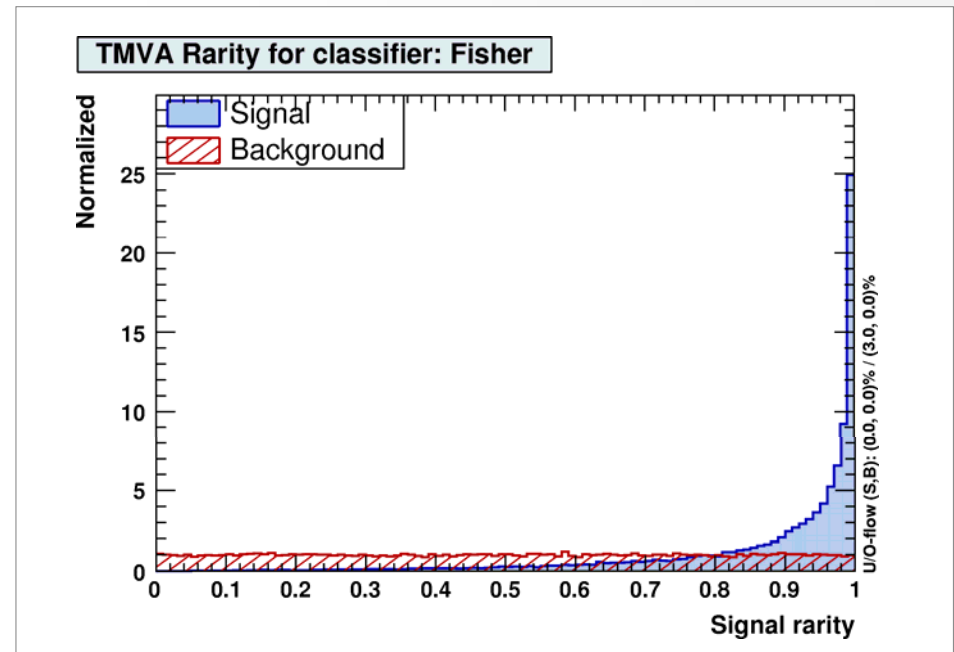
Evaluating the Classifiers

- There is no unique way to express the performance of a classifier
→ several benchmark quantities computed by **TMVA**
 - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
 - The *Separation*: $\frac{1}{2} \int \frac{(\hat{y}_S(y) - \hat{y}_B(y))^2}{\hat{y}_S(y) + \hat{y}_B(y)} dy$
 - “Rarity” implemented (background flat): $R(y) = \int_{-\infty}^y \hat{y}(y') dy'$ 
 - Other quantities ... see [Users Guide](#)



Evaluating the Classifiers

- There is no unique way to express the performance of a classifier
→ several benchmark quantities computed by **TMVA**
 - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
 - The *Separation*: $\frac{1}{2} \int \frac{(\hat{y}_S(y) - \hat{y}_B(y))^2}{\hat{y}_S(y) + \hat{y}_B(y)} dy$
 - “Rarity” implemented (background flat): $R(y) = \int_{-\infty}^y \hat{y}(y') dy'$ 
 - Other quantities ... see [Users Guide](#)



Evaluating the Classifiers

- There is no unique way to express the performance of a classifier
→ several benchmark quantities computed by **TMVA**
 - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
 - The *Separation*: $\frac{1}{2} \int \frac{(\hat{y}_S(y) - \hat{y}_B(y))^2}{\hat{y}_S(y) + \hat{y}_B(y)} dy$
 - “Rarity” implemented (background flat): $R(y) = \int_{-\infty}^y \hat{y}(y') dy'$
 - Other quantities ... see [Users Guide](#)
- 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)

Better classifier


MVA Methods:	Signal efficiency at bkg eff. (error):				Sepa- ration:	Signifi- cance:
	@B=0.01	@B=0.10	@B=0.30	Area		
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...)

Evaluation results ranked by best signal efficiency and purity (area)



MVA Methods:	Signal efficiency at bkg eff. (error):				Sepa- ration:	Signifi- cance:
	@B=0.01	@B=0.10	@B=0.30	Area		
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

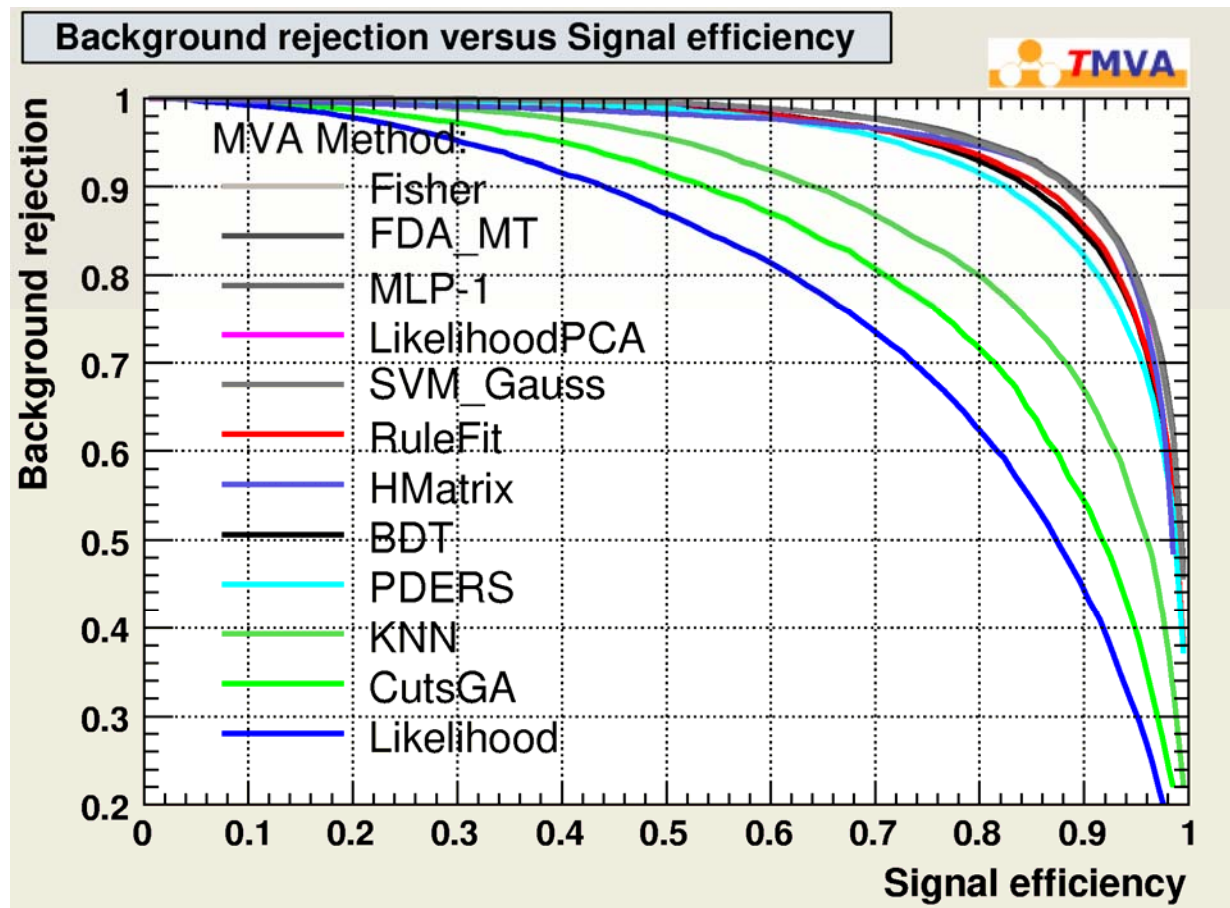
Testing efficiency compared to training efficiency (overtraining check)

MVA Methods:	Signal efficiency: from test sample (from traing sample)		
	@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

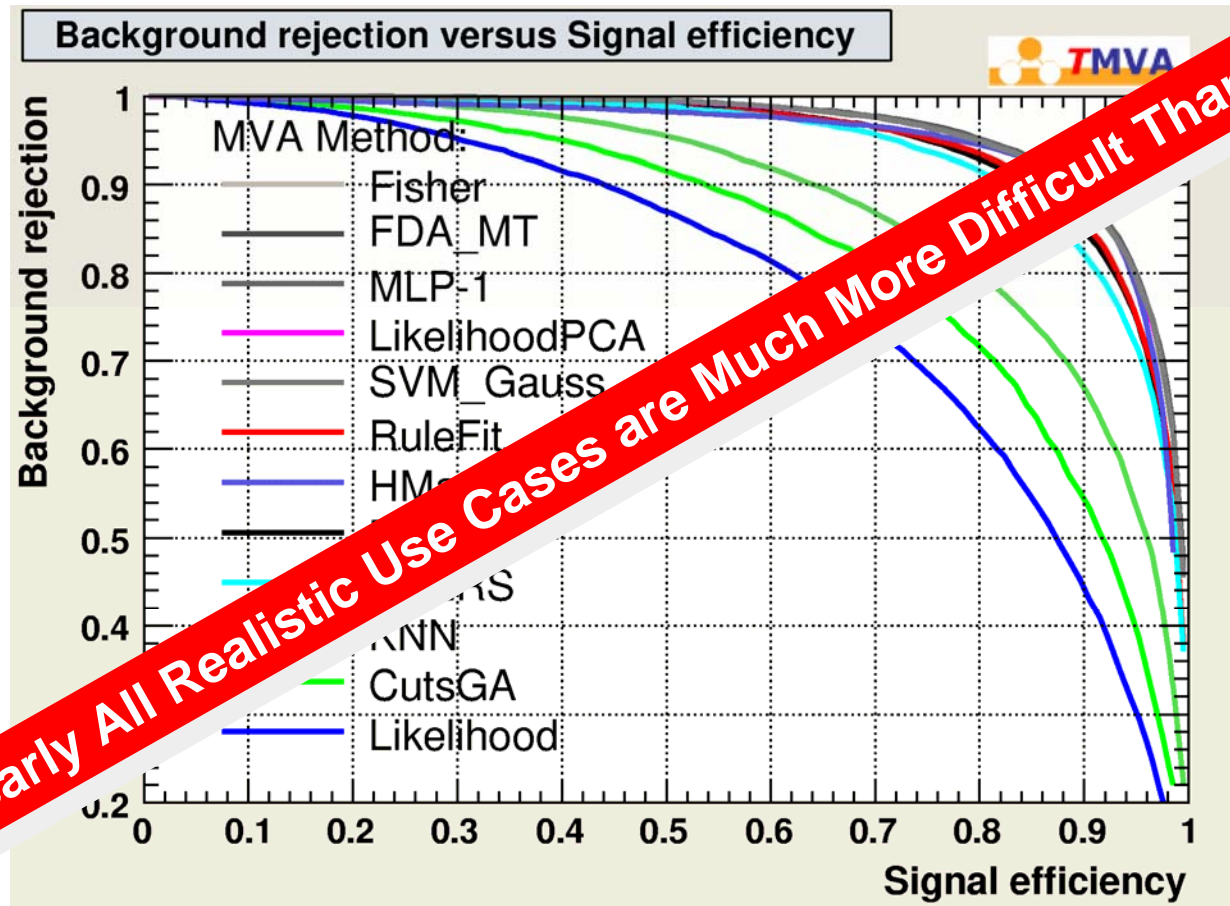
Evaluating the Classifiers (with a single plot...)

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

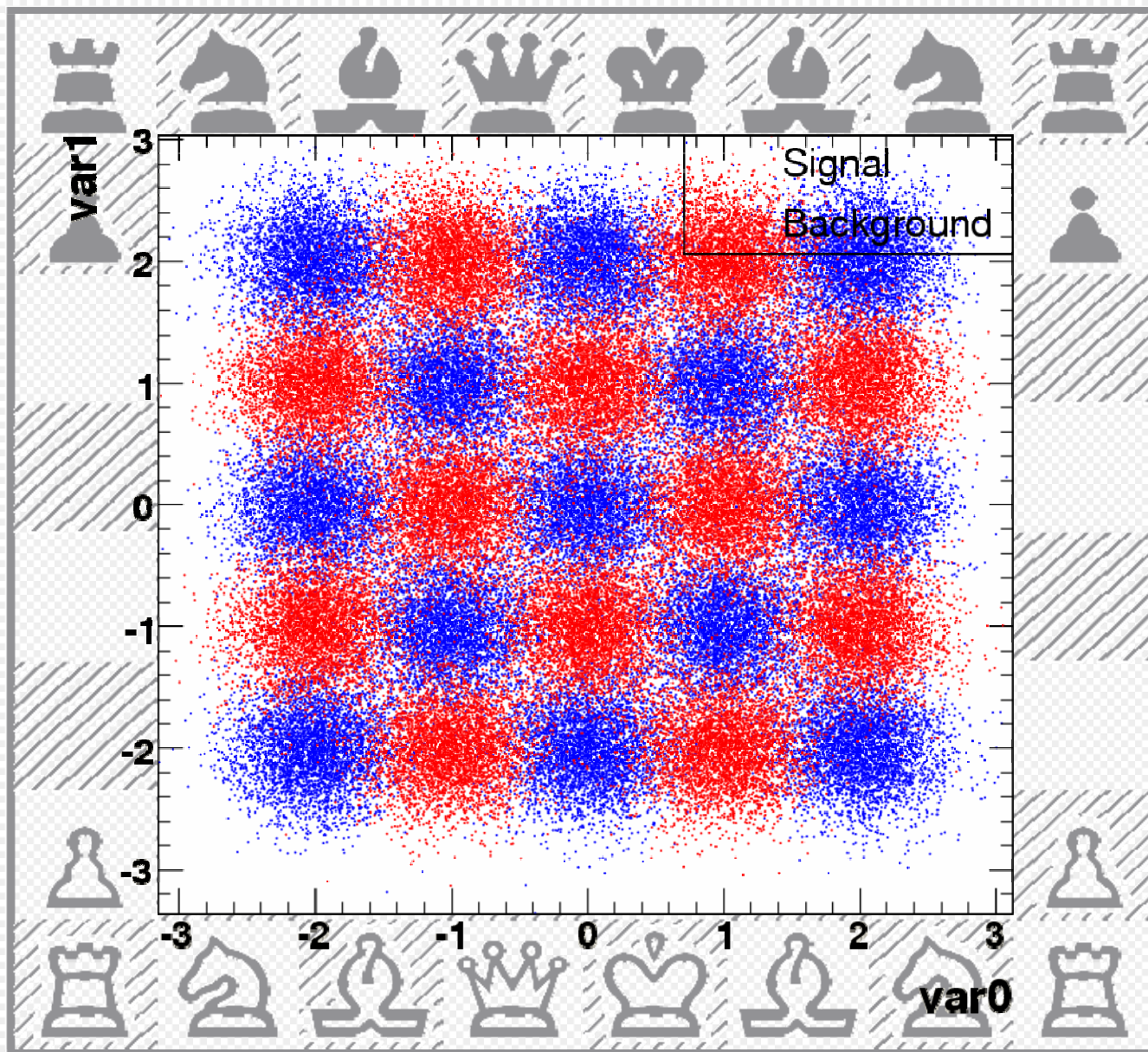


Evaluating the Classifiers (with a single plot...)

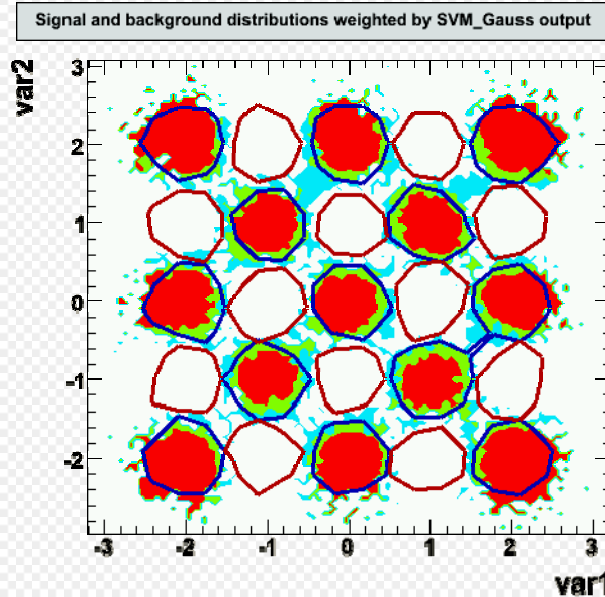
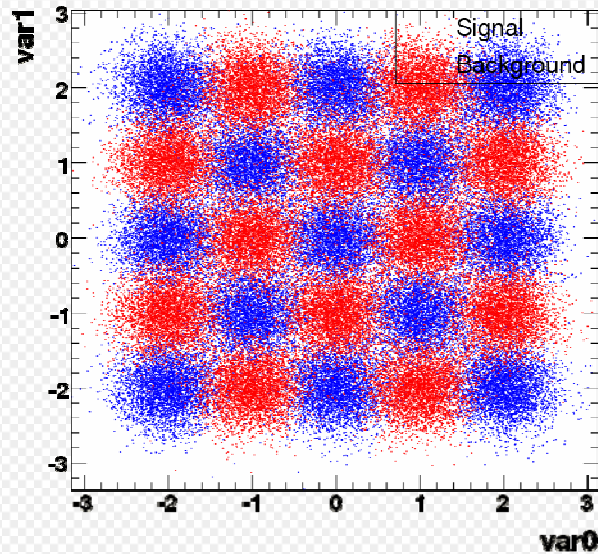
- Smooth background rejection versus signal efficiency curve:
(from cut on classifier output)



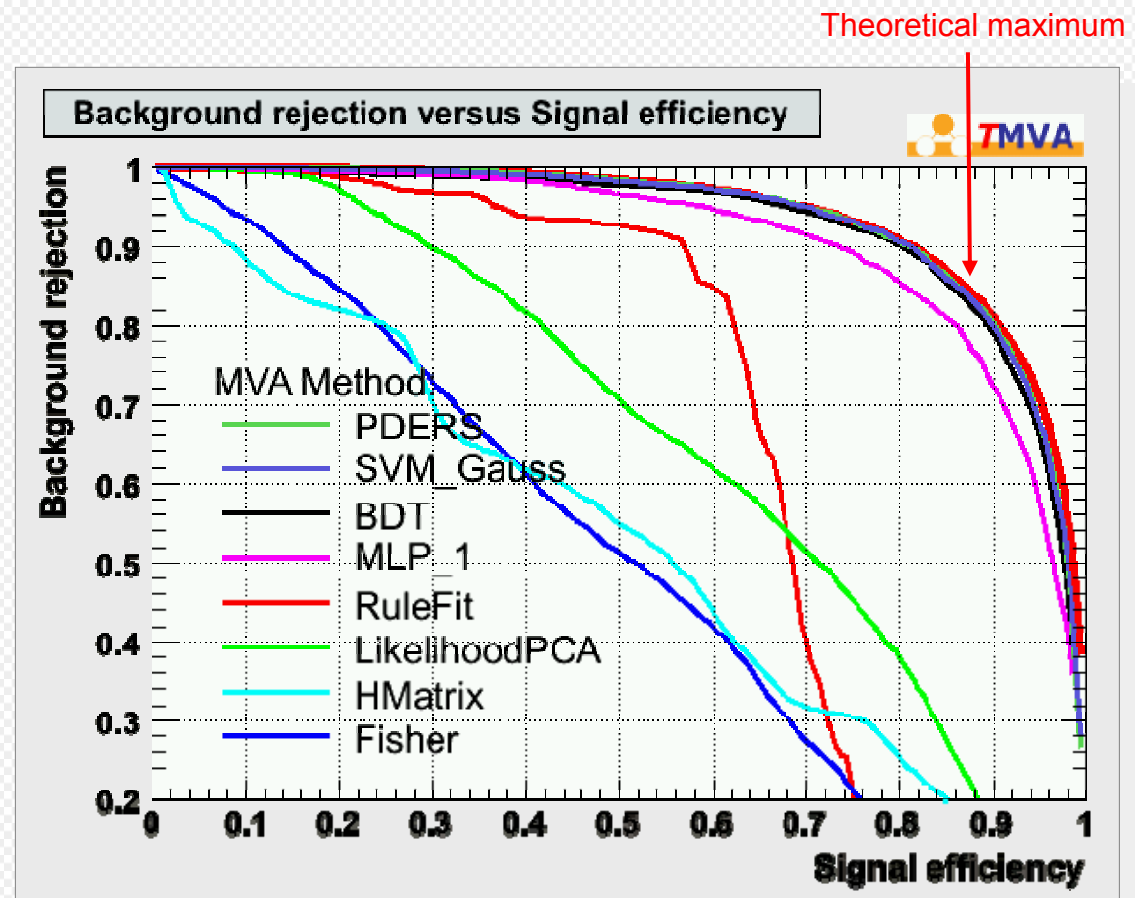
The “Schachbrett” Toy



The “Schachbrett” Toy



- Performance achieved without parameter tuning: PDERS and BDT best “out of the box” classifiers
- After specific tuning, also SVM und MLP perform well



Summary & Plans

Summary of the Classifiers and their Properties

Criteria		Classifiers								
		Cuts	Likelihood	PDERS / k-NN	H-Matrix	Fisher	MLP	BDT	RuleFit	SVM
Performance	no / linear correlations	☹	😊	😊	☹	😊	😊	☹	😊	😊
	nonlinear correlations	☹	☹	😊	☹	☹	😊	😊	☹	😊
Speed	Training	☹	😊	😊	😊	😊	☹	☹	☹	☹
	Response	😊	😊	☹/☹	😊	😊	😊	☹	☹	☹
Robustness	Overtraining	😊	☹	☹	😊	😊	☹	☹	☹	☹
	Weak input variables	😊	😊	☹	😊	😊	☹	☹	☹	☹
Curse of dimensionality		☹	😊	☹	😊	😊	☹	😊	☹	☹
Transparency		😊	😊	☹	😊	😊	☹	☹	☹	☹

The properties of the Function discriminant (FDA) depend on the chosen function

Outlook

Primary development from this Summer: **Generalised classifiers**

- Combine *any* classifier with *any other* classifier using *any* combination of input variables in *any* phase space region
- Be able to boost or bag any classifier
- Code is ready – now in testing mode. Dispatched till end of 2007.

Backup slides on:

- (i) more toy examples
- (ii) treatment of systematic uncertainties
- (iii) sensitivity to weak input variables

advertisement

We have a Users Guide !

Available on <http://tmva.sf.net>

arXiv physics/0703039
CERN-OPEN-2007-007
Document version 4
TMVA version 3.8
June 19, 2007
<http://tmva.sf.net>

TMVA

Toolkit for Multivariate Data Analysis with ROOT

Users Guide

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

With contributions from

A. Christov, S. Henrot-Versillé, M. Jachowski, A. Krasznahorkay Jr.,
Y. Mahalalel, R. Ospanov, X. Prudent, M. Wolter, A. Zemla

TMVA Users Guide
97pp, incl. code examples
arXiv physics/0703039

Copyrights & Credits

- **TMVA** is open source software
- 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

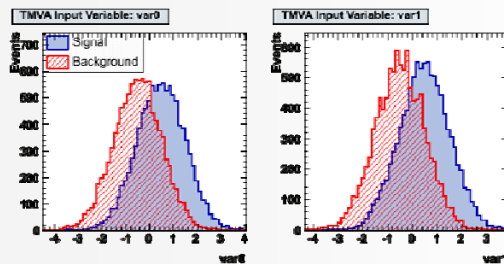
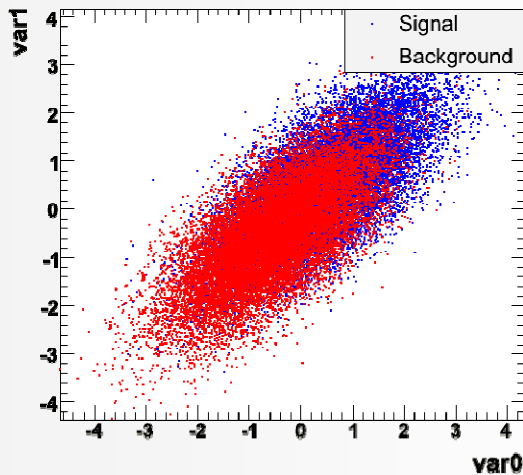
Acknowledgments: The fast development of TMVA would not have been possible without the contribution and feedback from many developers and users to whom we are indebted. We thank in particular the CERN Summer students Matt Jachowski (Stanford) for the implementation of TMVA's new MLP neural network, and Yair Mahalalel (Tel Aviv) for a significant improvement of PDERS, the Krakow student Andrzej Zemla and his supervisor Marcin Wolter for programming a powerful Support Vector Machine, as well as Rustem Ospanov for the development of a fast k-NN algorithm. We are grateful to Doug Applegate, Kregg Arms, René Brun and the ROOT team, Tancredi Carli, Zhiyi Liu, Elzbieta Richter-Was, Vincent Tisserand and Alexei Volk for helpful conversations.

More Toy Examples

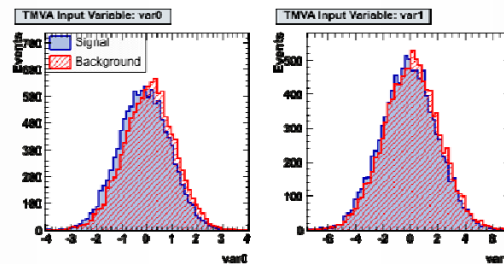
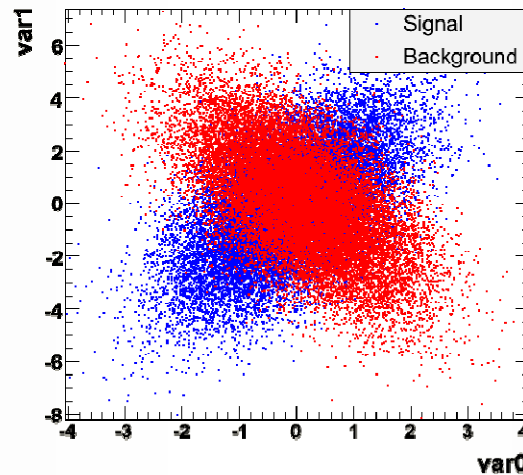
More Toys: Linear-, Cross-, Circular Correlations

- Illustrate the behaviour of linear and nonlinear classifiers

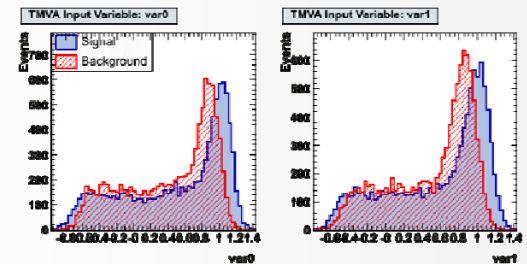
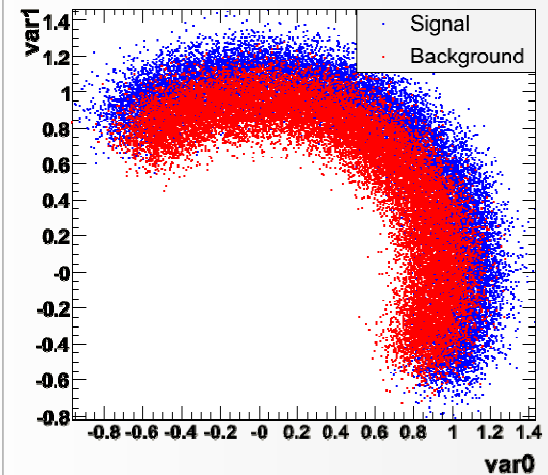
Linear correlations
(same for signal and background)



Linear correlations
(opposite for signal and background)

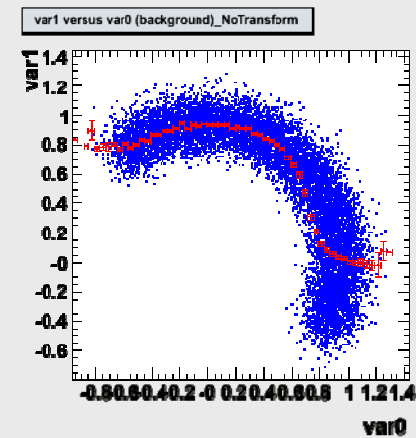
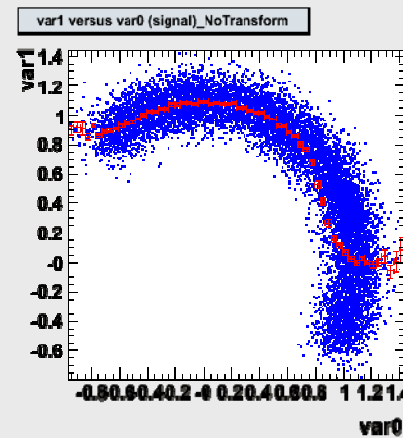
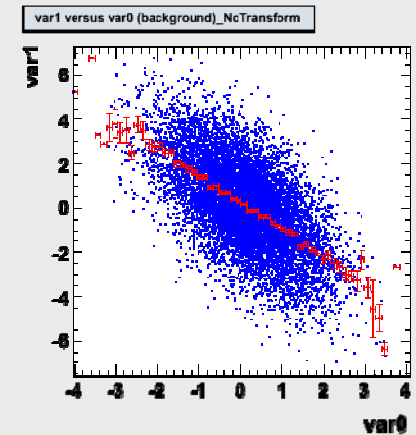
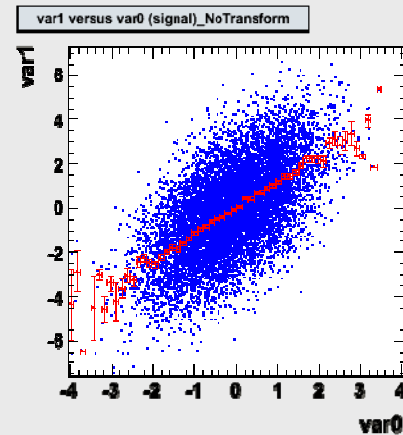
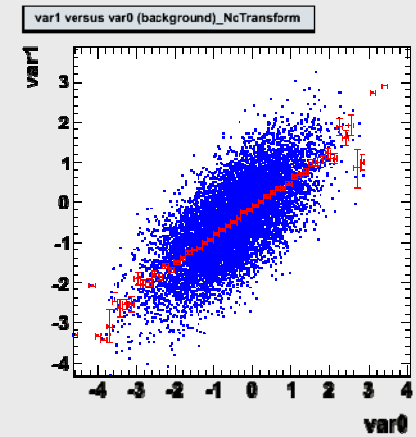
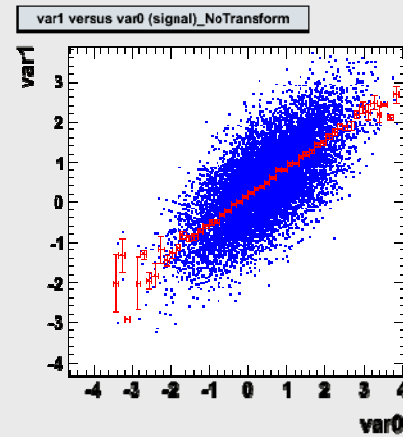


Circular correlations
(same for signal and background)



- How does linear decorrelation affect strongly nonlinear cases ?

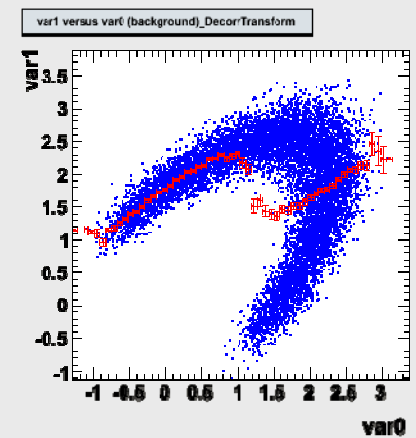
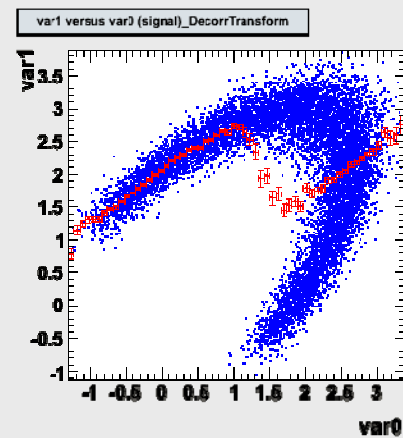
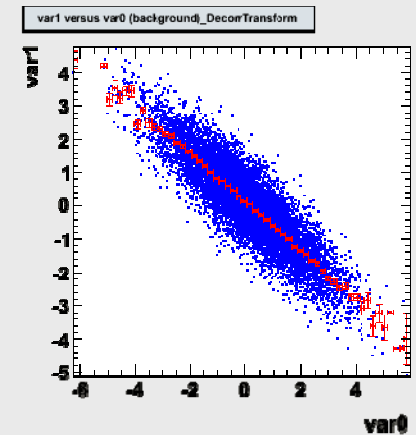
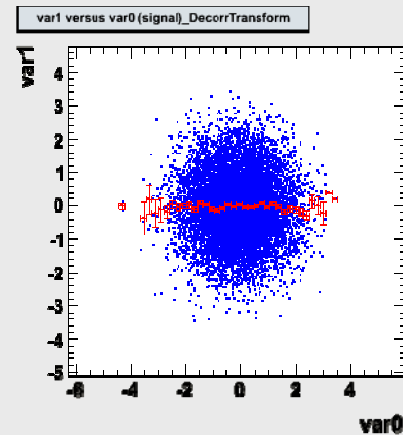
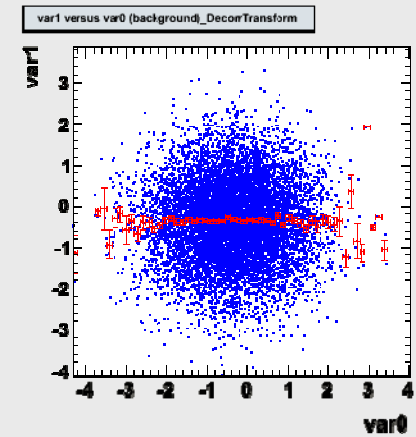
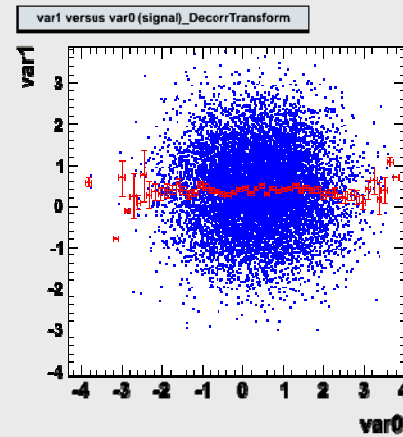
Original correlations



- How does linear decorrelation affect strongly nonlinear cases ?

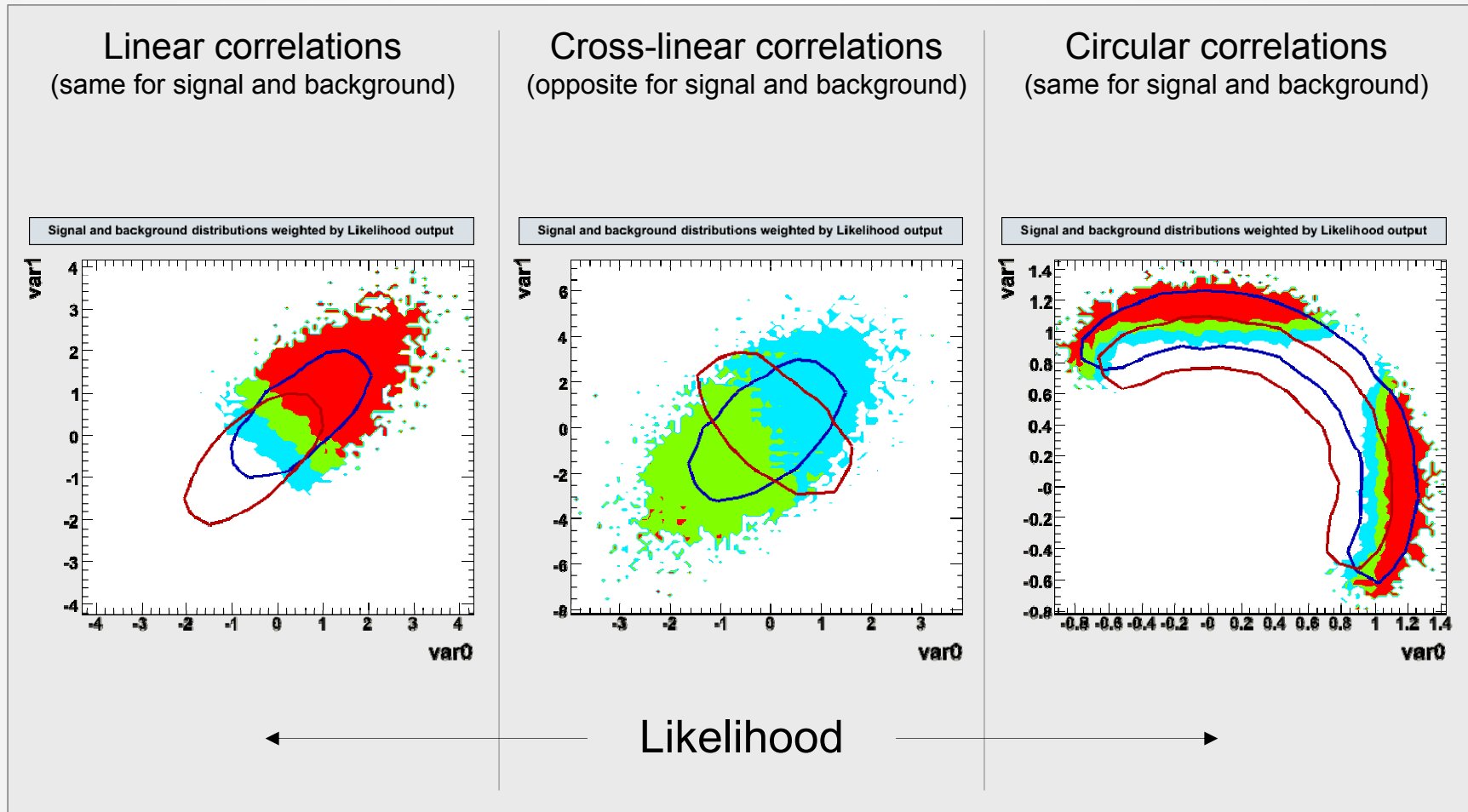
Original correlations

SQRT decorrelation



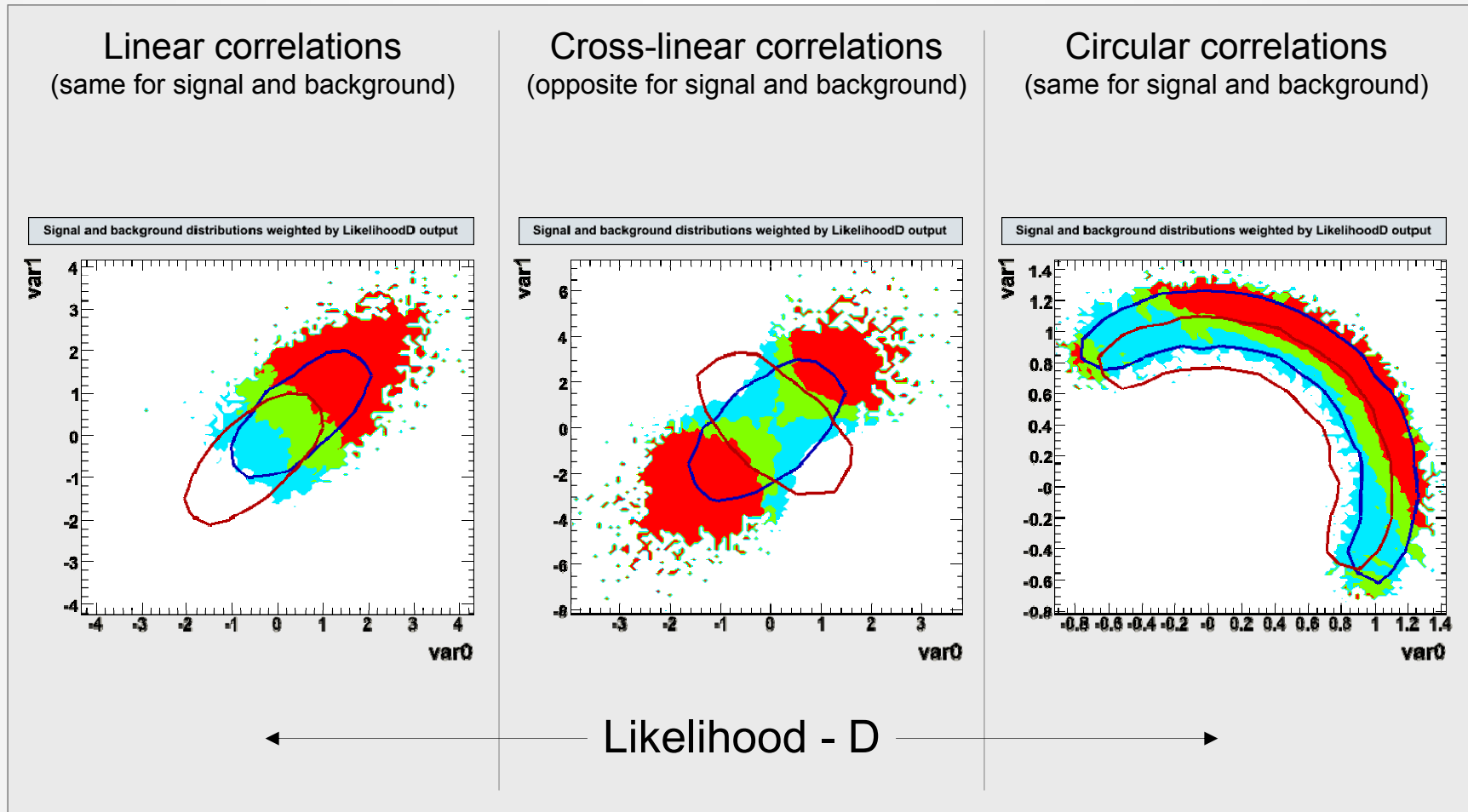
Weight Variables by Classifier Output

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



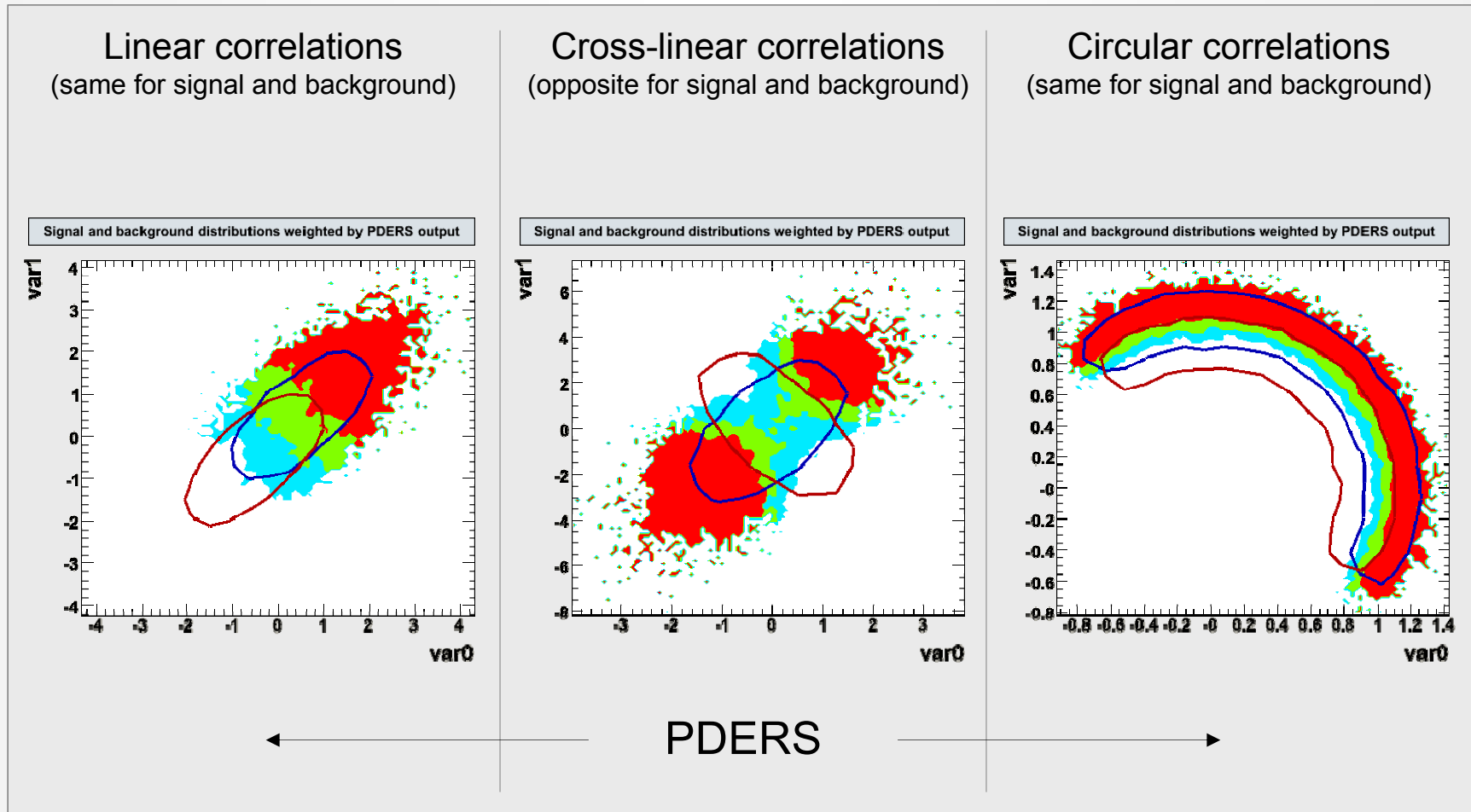
Weight Variables by Classifier Output

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



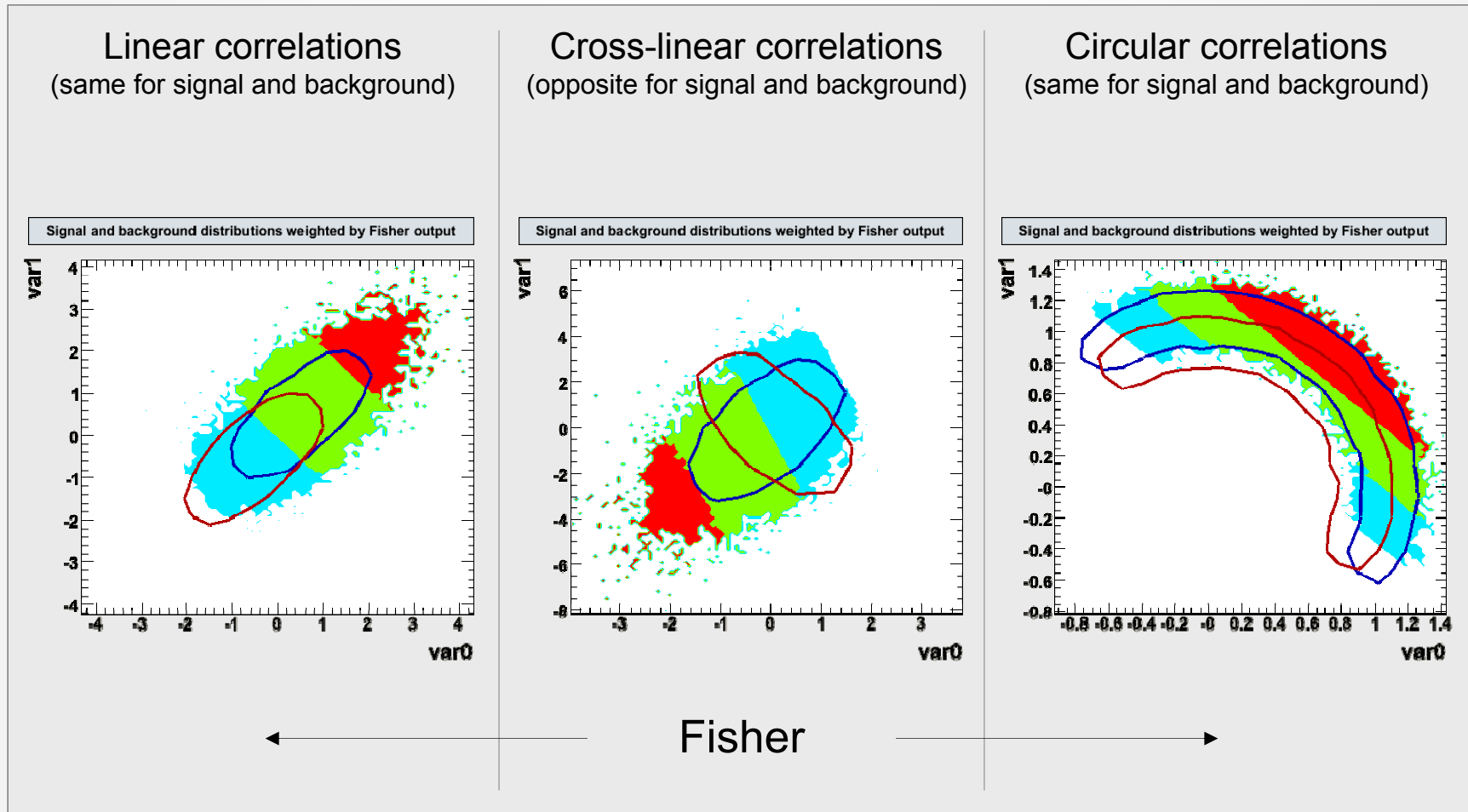
Weight Variables by Classifier Output

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



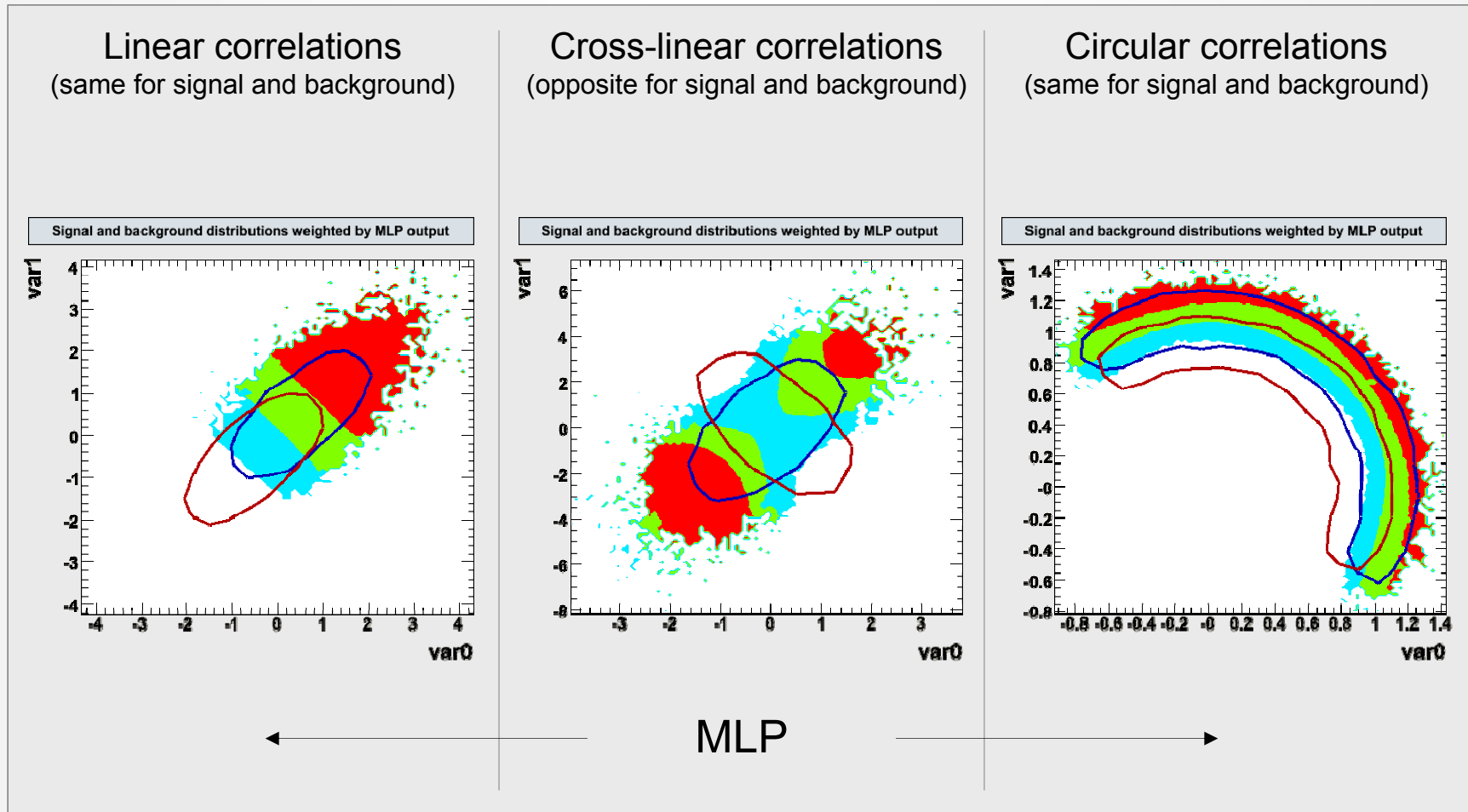
Weight Variables by Classifier Output

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



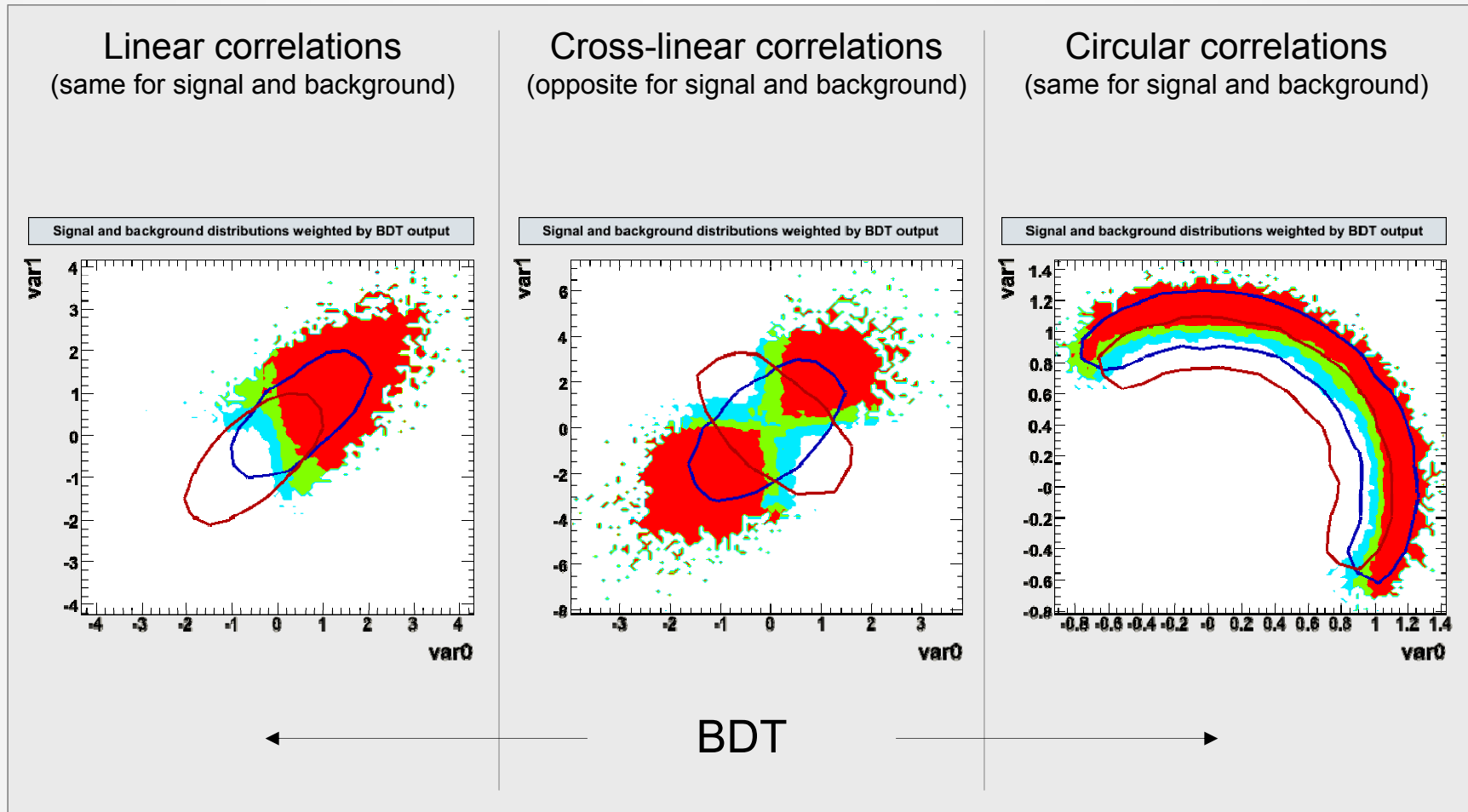
Weight Variables by Classifier Output

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



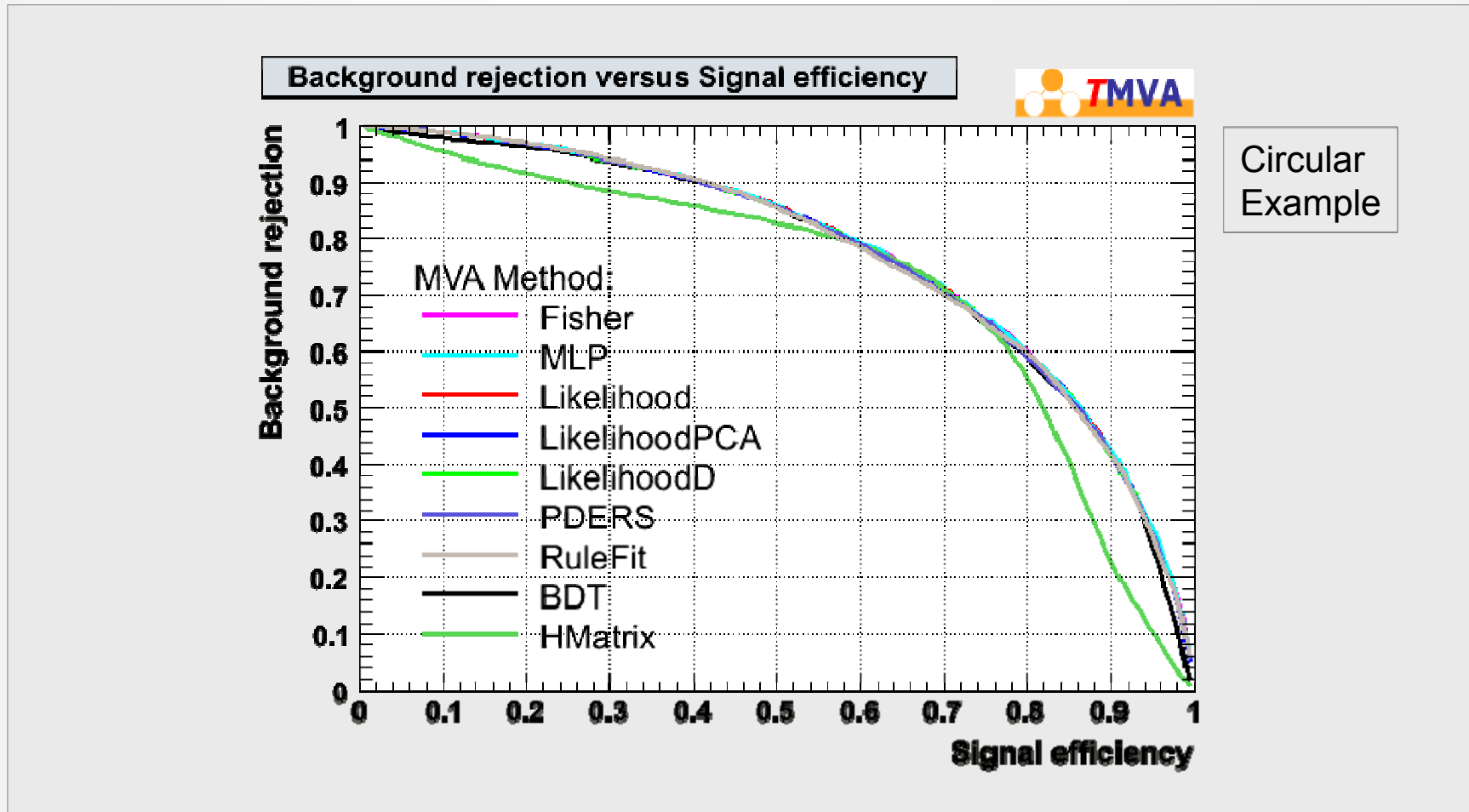
Weight Variables by Classifier Output

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



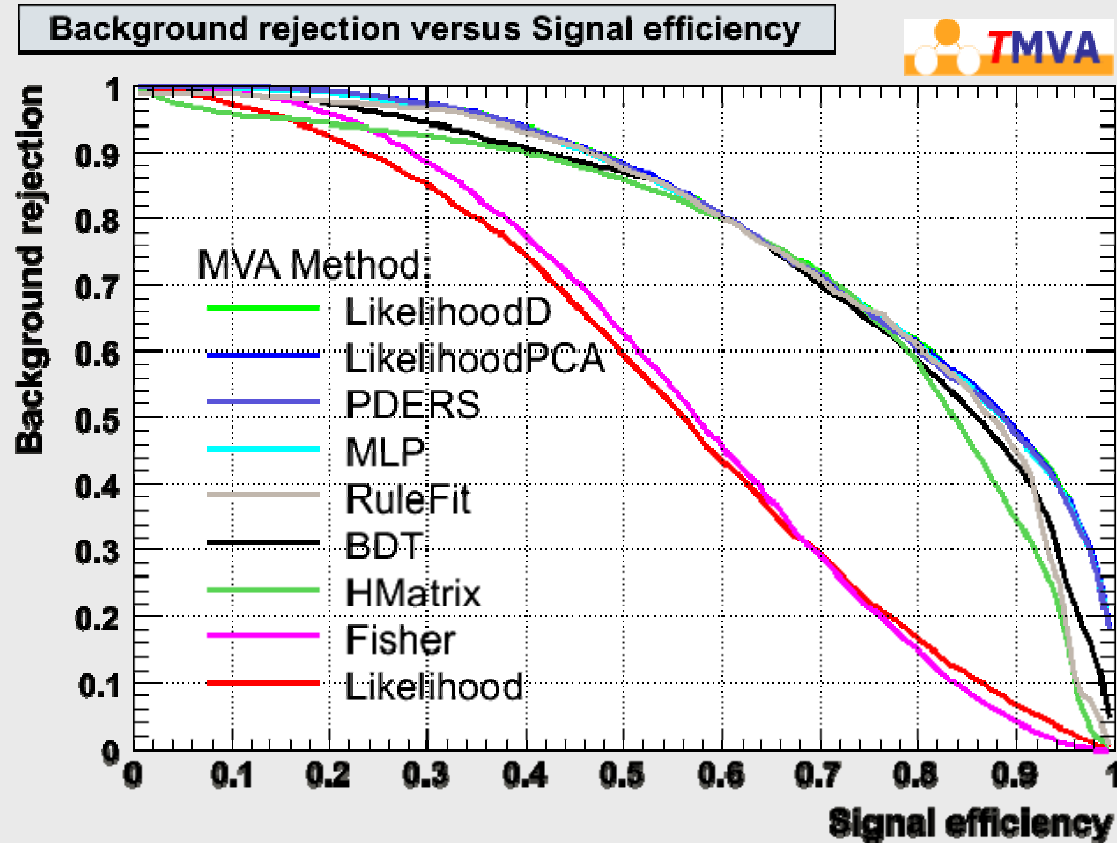
Final Classifier Performance

- Background rejection versus signal efficiency curve:



Final Classifier Performance

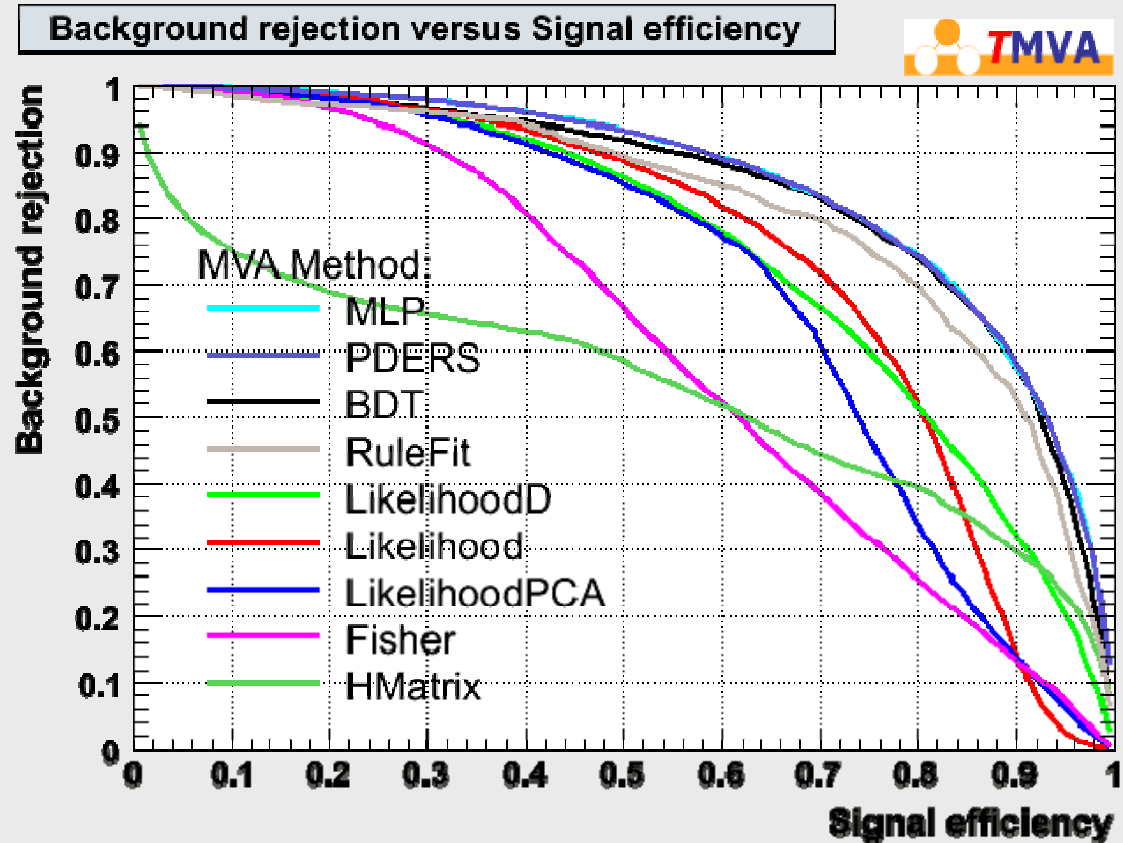
- Background rejection versus signal efficiency curve:



Circular Example

Final Classifier Performance

- Background rejection versus signal efficiency curve:

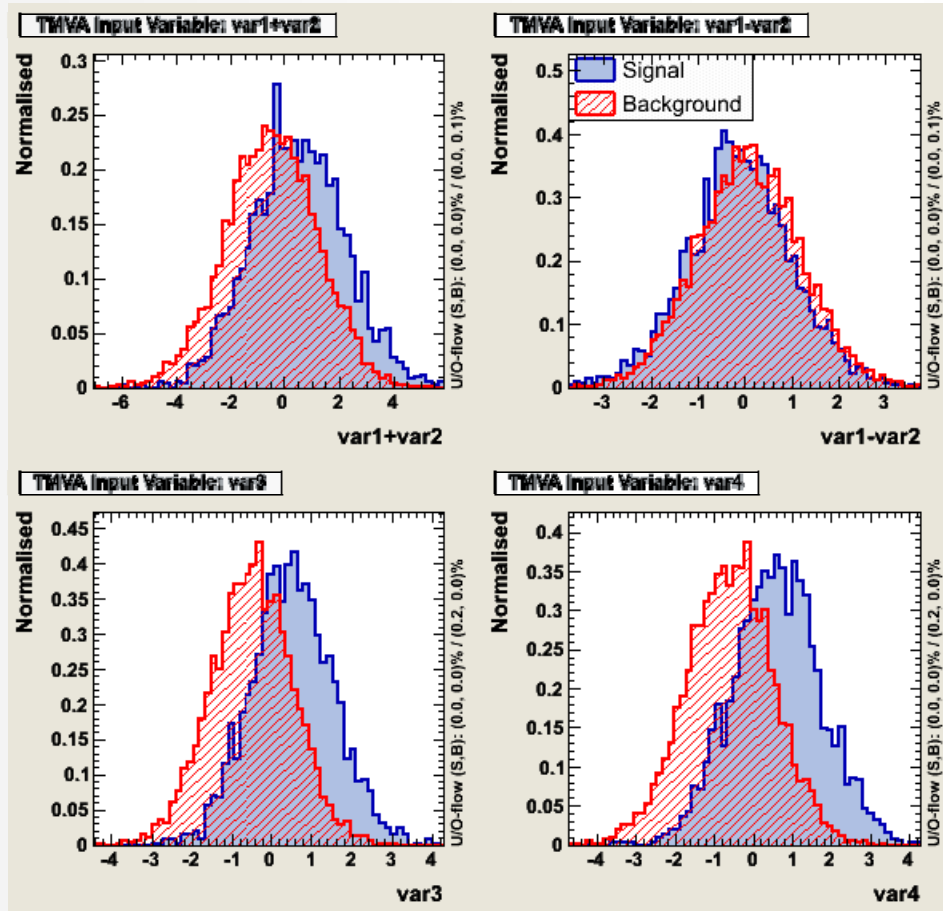


Circular Example

Some words on systematics

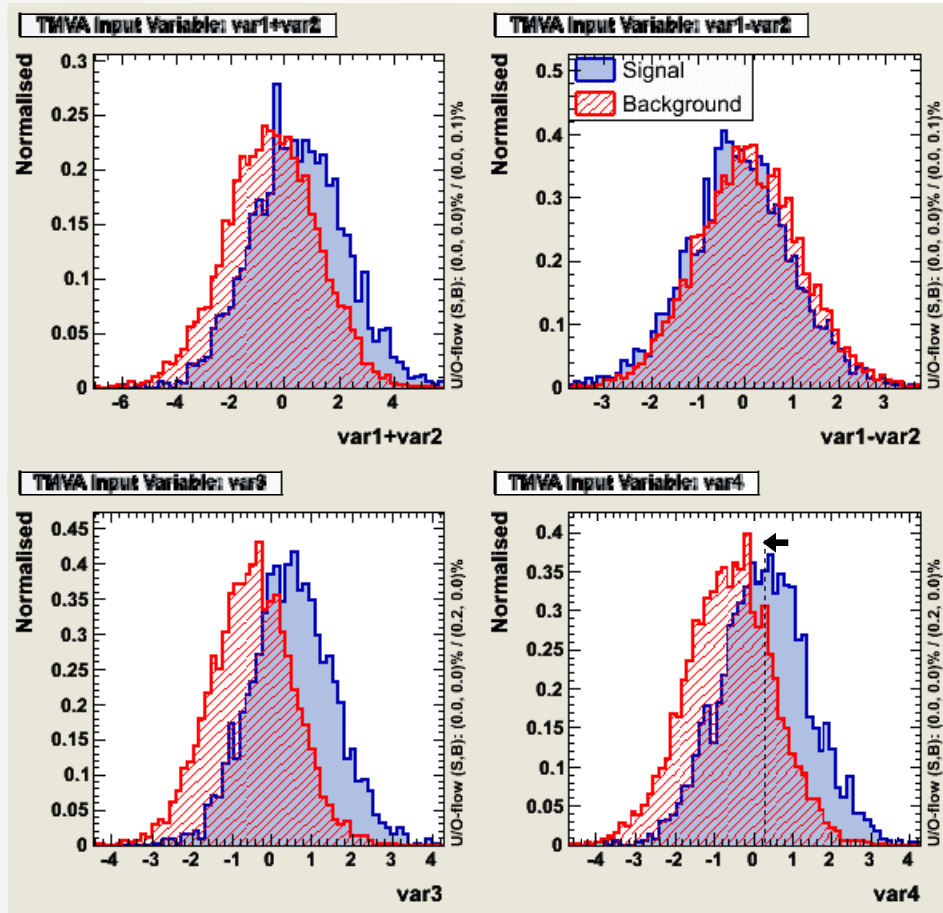
Treatment of Systematic Uncertainties

- Assume strongest variable “var4” suffers from systematic uncertainty



Treatment of Systematic Uncertainties

- Assume strongest variable “var4” suffers from systematic uncertainty



“Calibration uncertainty” may shift the central value and hence worsen the discrimination power of “var4”

Treatment of Systematic Uncertainties

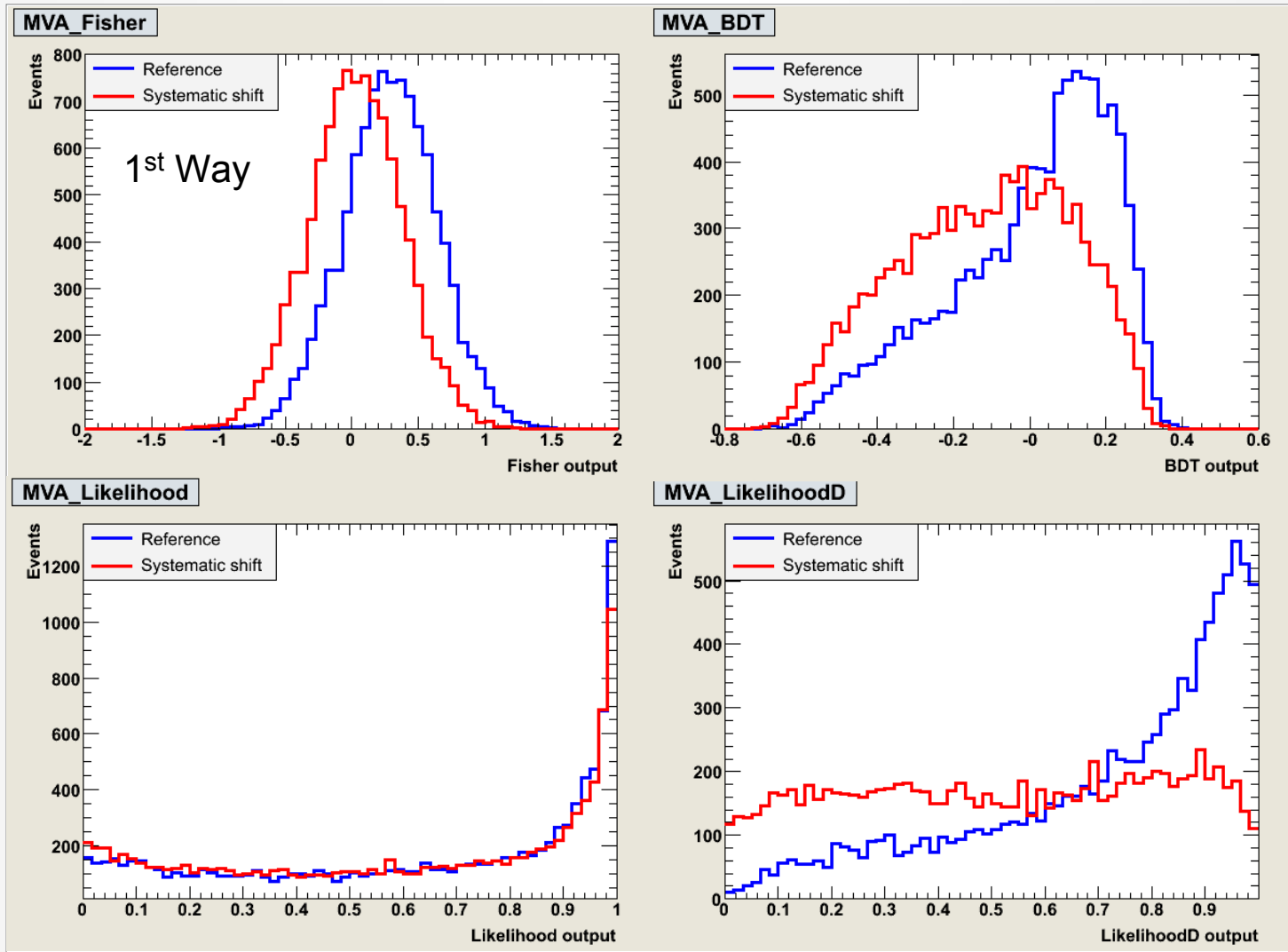
■ Assume strongest variable “var4” suffers from systematic uncertainty

➡ (at least) Two ways to deal with it:

1. Ignore the systematic in the training, and evaluate systematic error on classifier output
 - Drawbacks:
 - “var4” appears stronger in training than it might be → suboptimal performance
 - Classifier response will strongly depend on “var4”
 2. Train with shifted (= weakened) “var4”, and evaluate systematic error on classifier output
 - Cures previous drawbacks
- ➡ If classifier output distributions can be validated with data control samples, the second drawback is mitigated, but not the first one (the performance loss) !

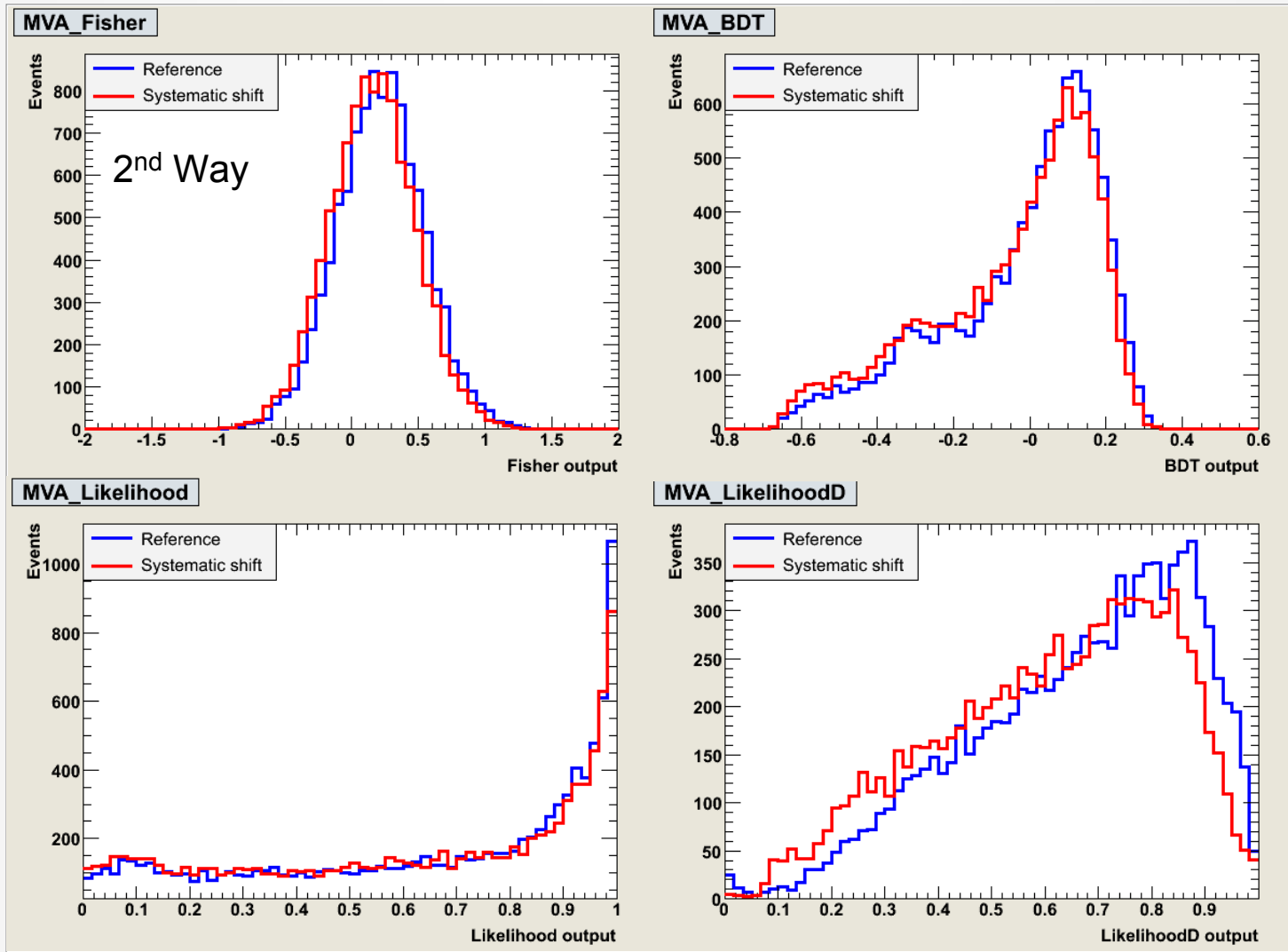
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



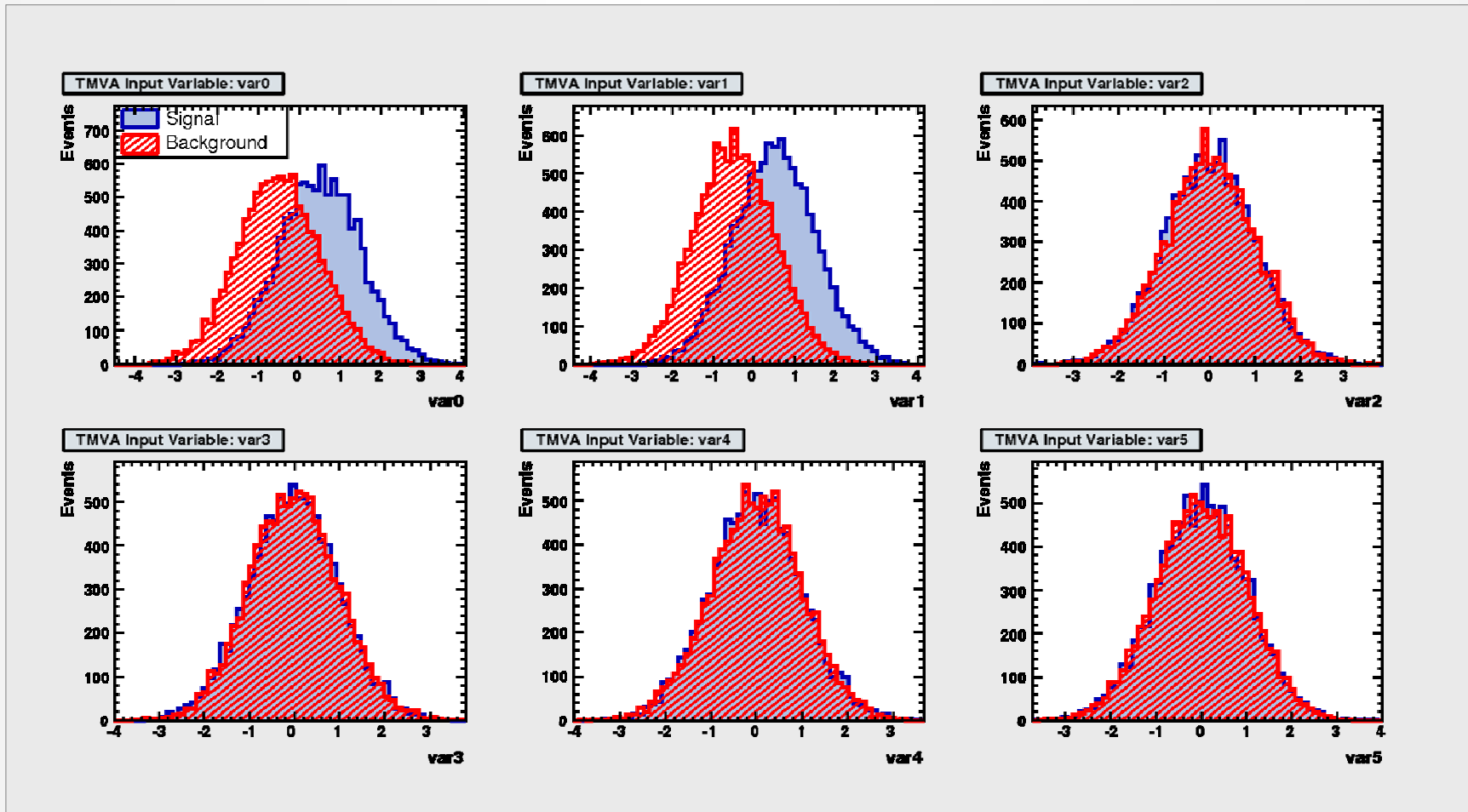
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



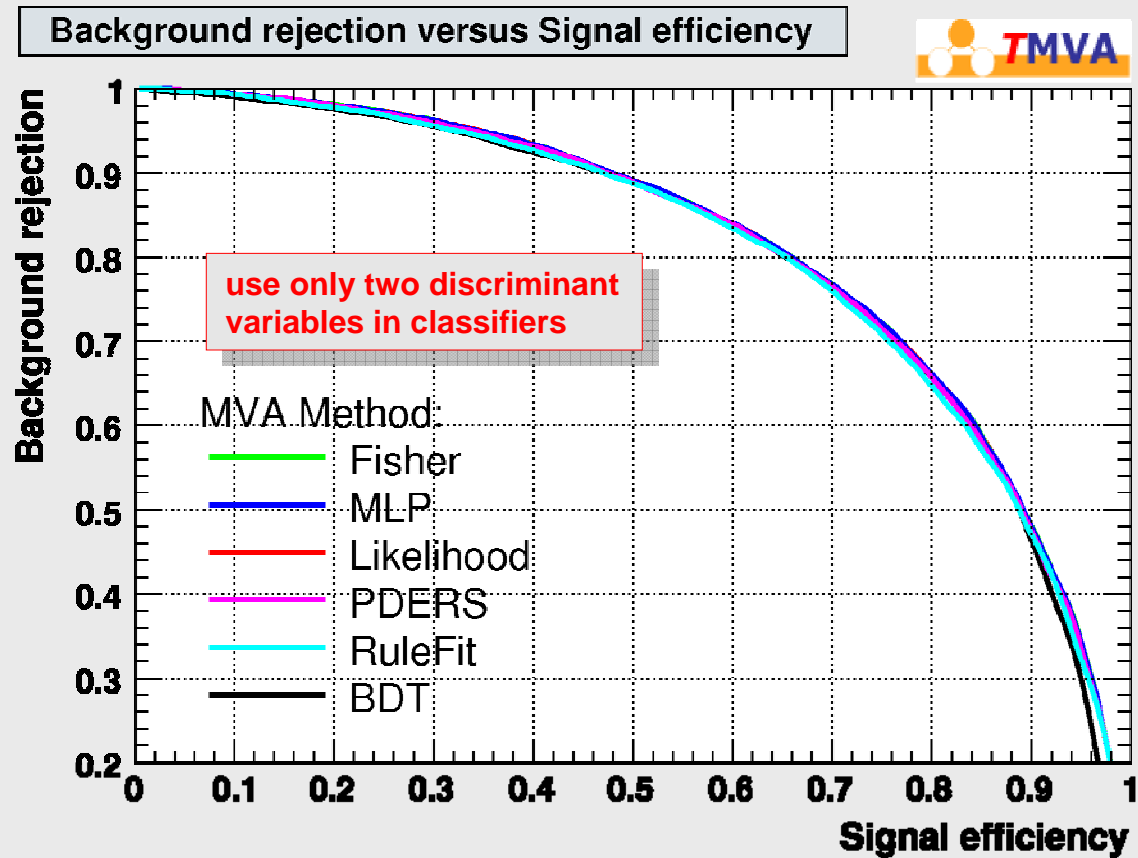
Stability with Respect to Irrelevant Variables

- 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 ?

