Multivariate Data Analysis with *T*MVA

Andreas Hoecker(*) (CERN)

Statistical Tools Workshop, DESY, Germany, June 19, 2008

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

And the contributors: A. Christov, Or Cohen, Kamil Kraszewski, Krzysztof Danielowski, S. Henrot-Versillé, M. Jachowski, A. Krasznahorkay Jr., Maciej Kruk, Y. Mahalalel, R. Ospanov, X. Prudent, A. Robert, F. Tegenfeldt, K. Voss, M. Wolter, A. Zemla

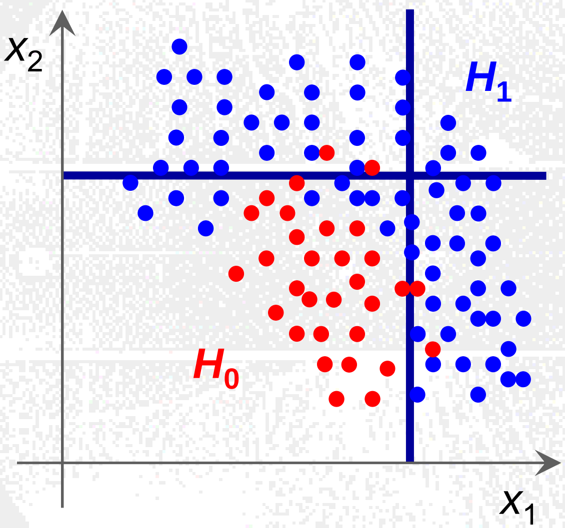
See acknowledgments on page 43

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

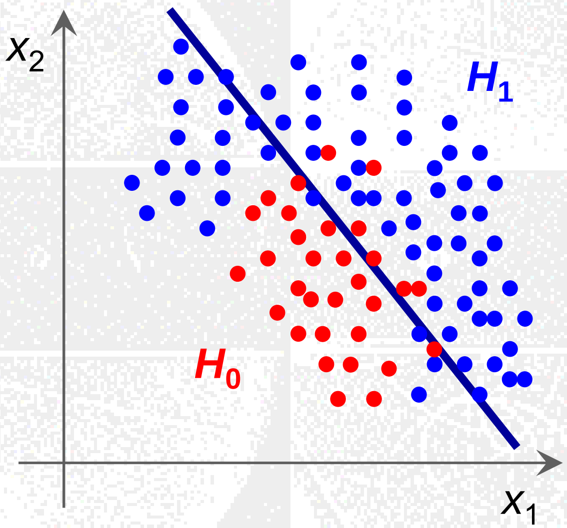
Event Classification

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

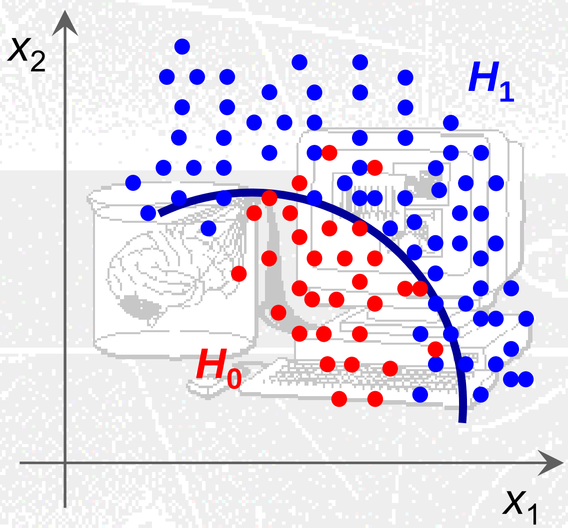
Rectangular cuts?



A linear boundary?



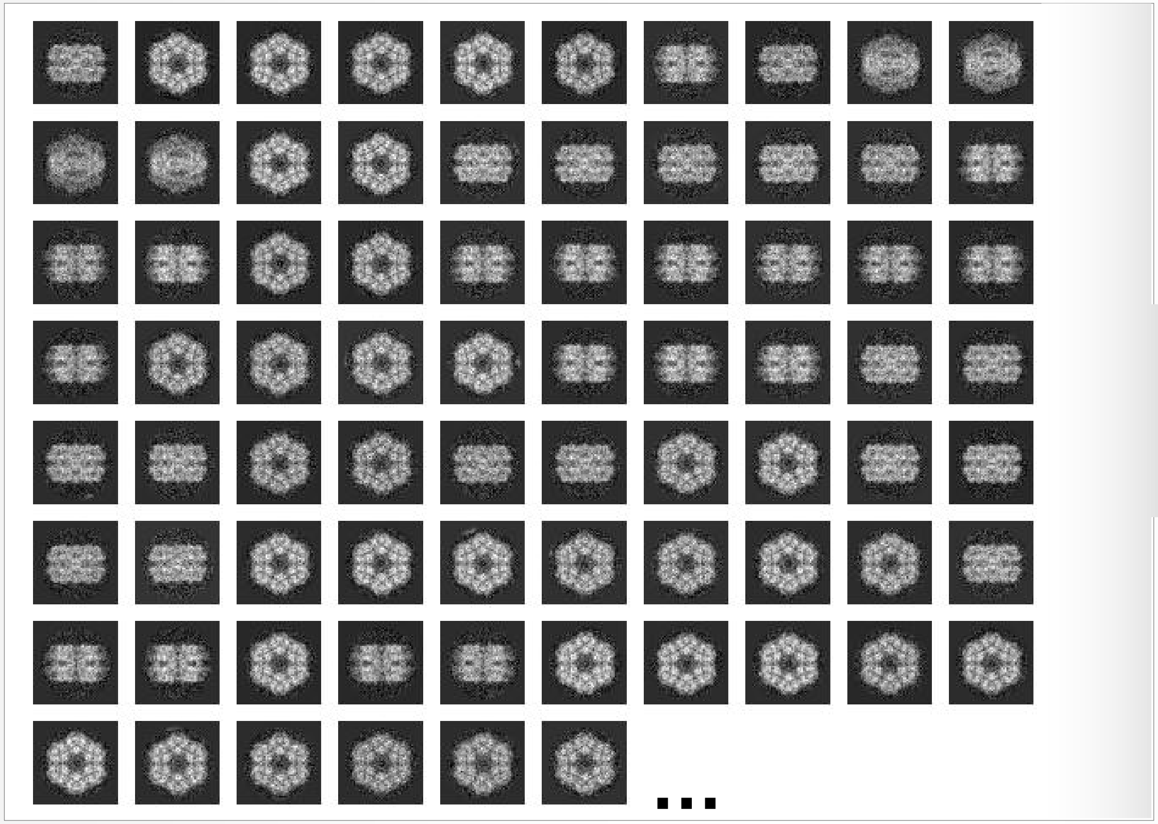
A nonlinear one?



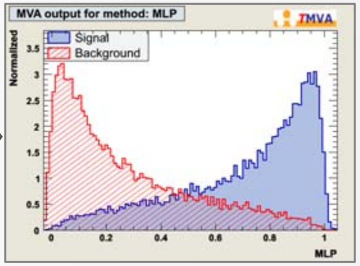
- How can we decide this in an optimal way ? → Let the machine learn it !

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*



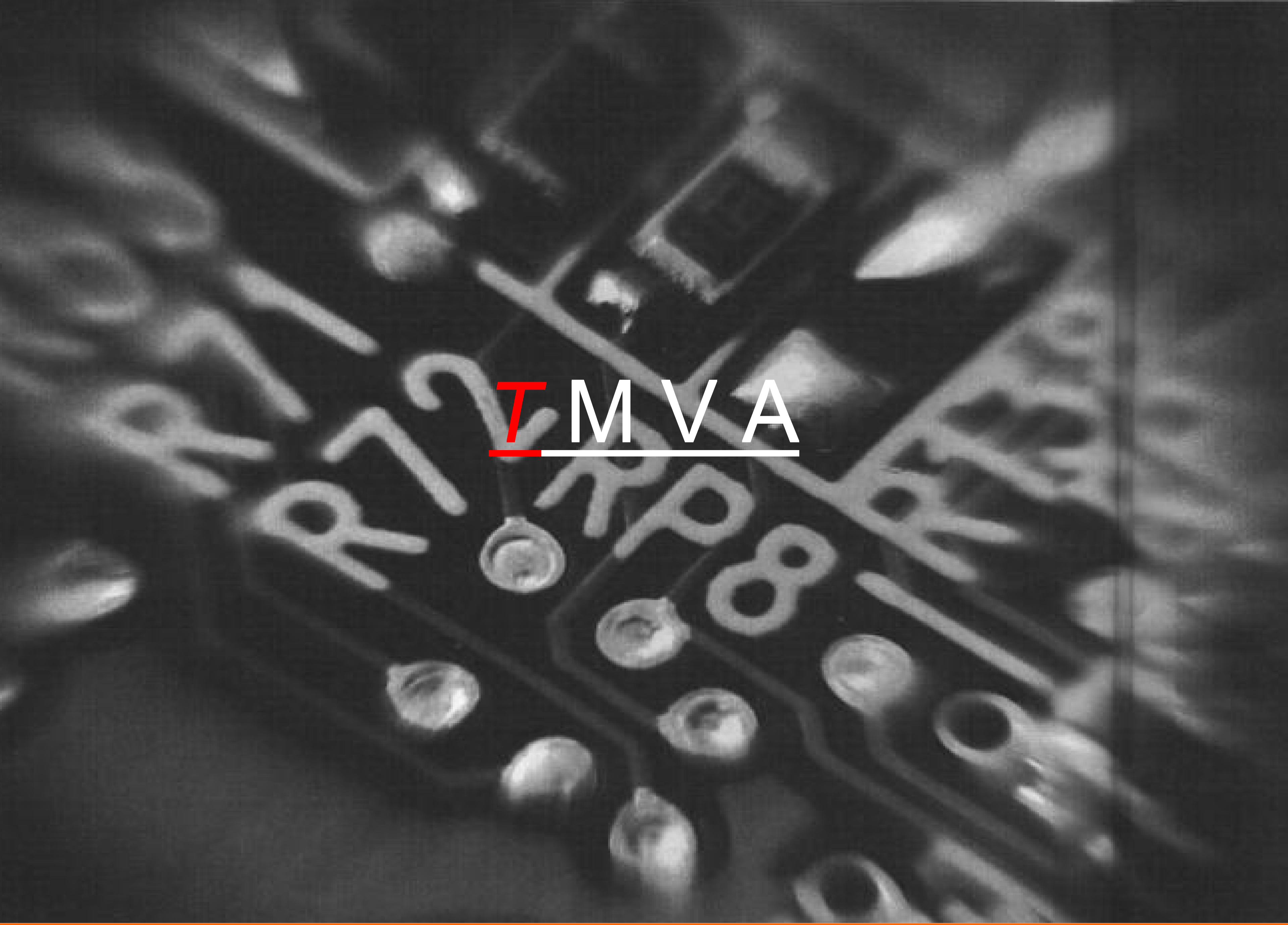
$y(H_0) \rightarrow 0$, $y(H_1) \rightarrow 1$



MV regression is also interesting !

In work for TMVA !

TMVA



What is *T*MVA

- ROOT: is the analysis framework used by most (HEP)-physicists
- Idea: rather than just implementing new MVA techniques and making them available in ROOT (*i.e.*, like TMulitLayerPercetron does):
  - Have one common platform / interface for all MVA classifiers
  - Have common data pre-processing capabilities
  - Train and test all classifiers on same data sample and evaluate consistently
  - Provide common analysis (ROOT scripts) and application framework
  - Provide access with and without ROOT, through macros, C++ executables or python

Outline of this talk

- The *T*MVA project
- Quick survey of available classifiers and processing steps
- Evaluation tools

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 4 core developers, and 16 active contributors
  - >2400 downloads since March 2006 (not accounting cvs checkouts and ROOT users)
- Written in C++, relying on core ROOT functionality
- Integrated and distributed with ROOT since ROOT v5.11/03

The TMVA Classifiers

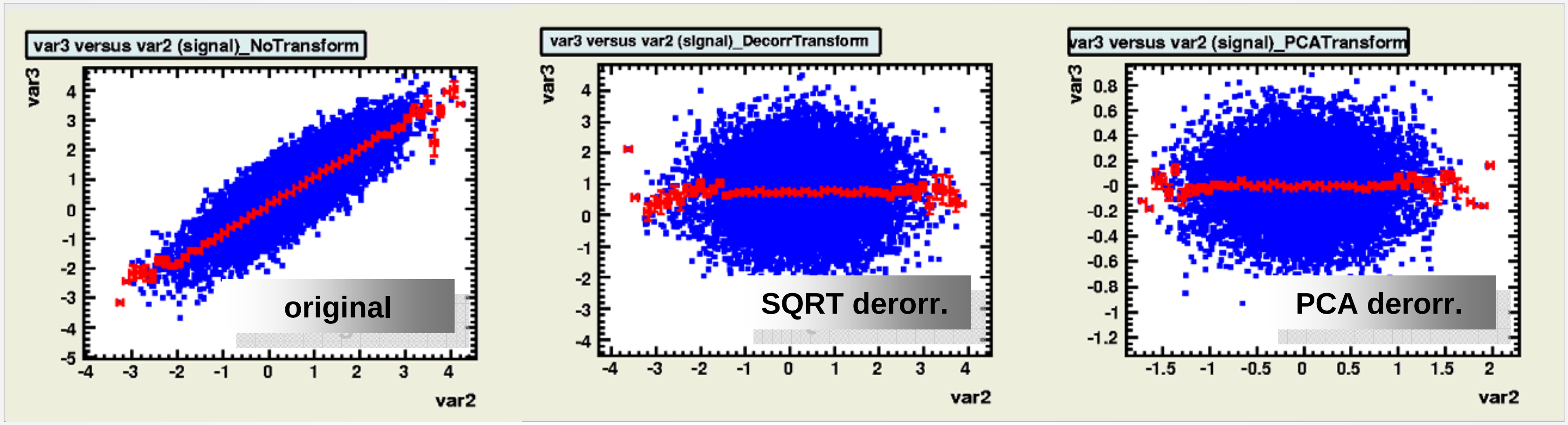
- Currently implemented classifiers :
  - Rectangular cut optimisation
  - Projective and multidimensional likelihood estimator
  - k-Nearest Neighbor algorithm
  - Fisher and H-Matrix discriminants
  - Function discriminant
  - Artificial neural networks (3 *multilayer perceptron* implementations)
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

- Currently implemented data preprocessing stages:
  - Decorrelation
  - Principal Value Decomposition
  - Transformation to uniform and Gaussian distributions (*coming soon*)

Data Preprocessing: Decorrelation

- Commonly realised for all methods in *T*MVA (centrally in **`DataSet`** class)

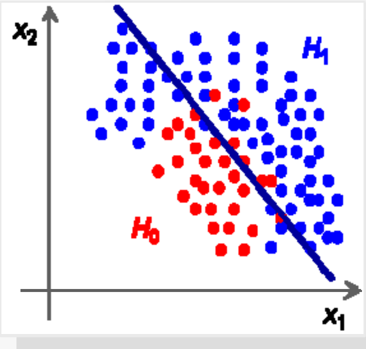
- Removal of linear correlations by rotating input variables
  - using the "square-root" of the correlation matrix
  - using the Principal Component Analysis

- 

Rectangular Cut Optimisation

- Simplest method: cut in rectangular variable volume

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

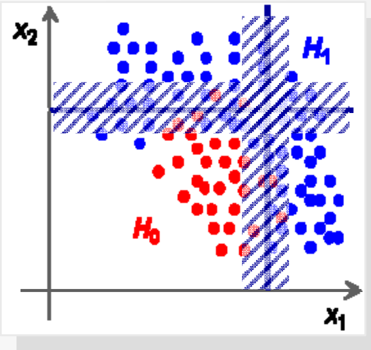


- Technical challenge: how to find optimal cuts ?
  - MINUIT fails due to non-unique solution space
  - *T*MVA uses: **Monte Carlo sampling**, **Genetic Algorithm**, **Simulated Annealing**
  - Huge speed improvement of volume search by sorting events in binary tree

- 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



PDE introduces fuzzy logic

Likelihood ratio for event i_{event}

PDFs

discriminating variables

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

Species: signal, background types

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

PDE Approach: Estimating PDF Kernels

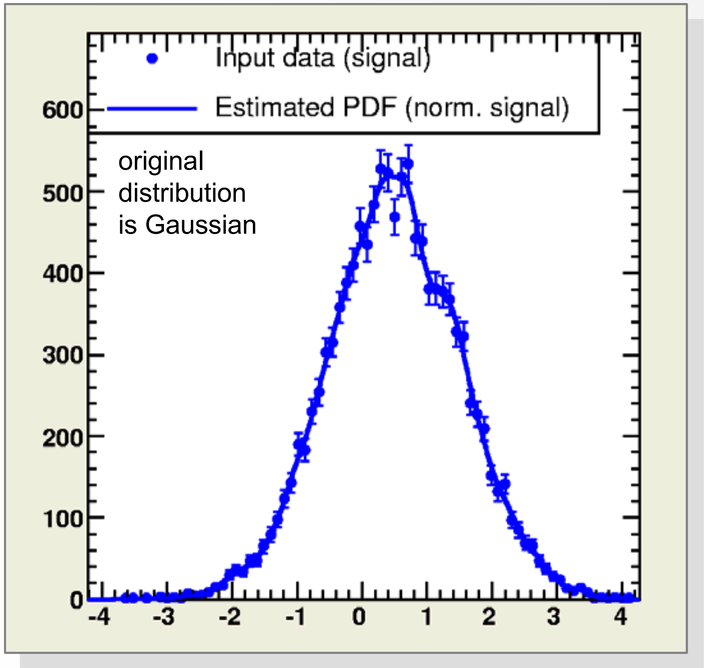
- Technical challenge: how to estimate the PDF shapes
  - **3 ways:** **parametric fitting (function)** | **nonparametric fitting** | **event counting**

Difficult to automate for arbitrary PDFs

Easy to automate, can create artefacts/suppress information

Automatic, unbiased, but suboptimal

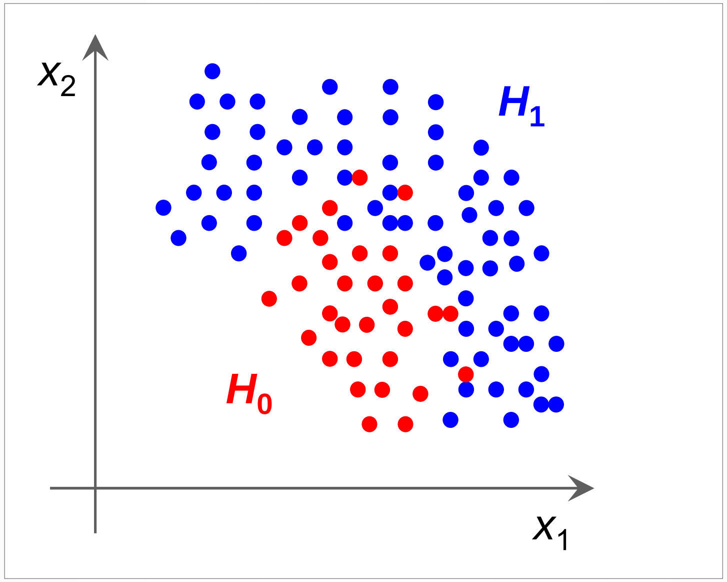
- We have chosen to implement nonparametric fitting in *T*MVA
  - Binned shape interpolation using spline functions and adaptive smoothing
  - Unbinned adaptive kernel density estimation (KDE) with Gaussian smearing
  - *T*MVA performs automatic validation of goodness-of-fit



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"

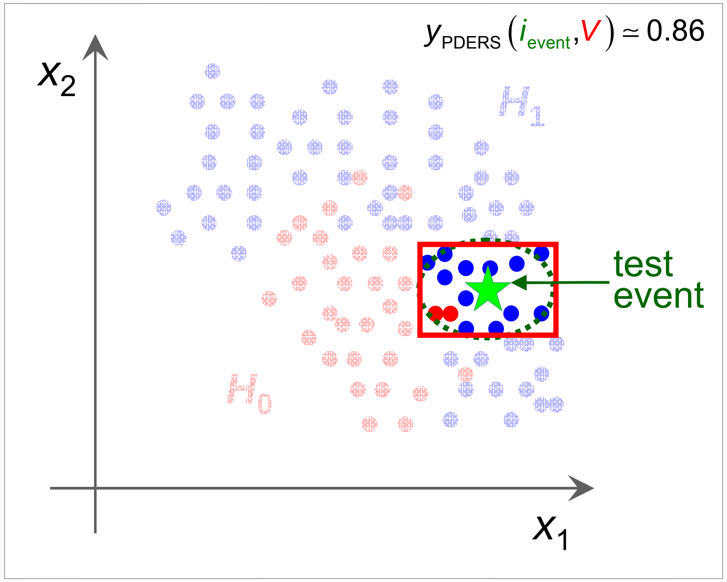
Carli-Koblitz, NIM A501, 576 (2003)



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)



- 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

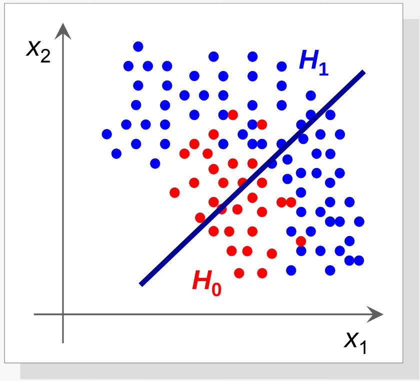
Multidimensional PDE Approach

k-Nearest Neighbor

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

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:

"Fisher coefficients"

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

  - 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)

Function discriminant analysis (FDA)

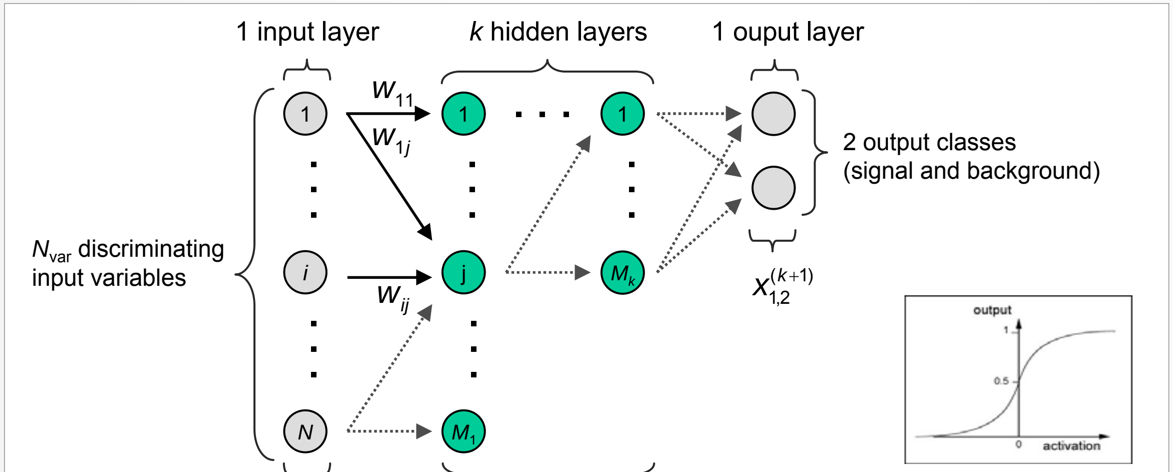
Fit any user-defined function of input variables requiring that signal events return $\rightarrow$1 and background $\rightarrow$0

- Parameter fitting: Genetics Alg., MINUIT, MC and combinations
- Easy reproduction of Fisher result, but can add nonlinearities
- Very transparent discriminator

Nonlinear Analysis: Artificial Neural Networks

- Achieve nonlinear classifier response by “activating” output nodes using nonlinear weights

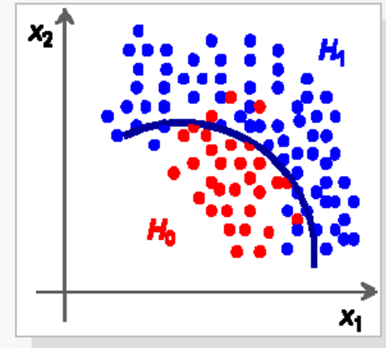
Feed-forward Multilayer Perceptron



(“Activation” function)

$x_{i=1..N_{var}}^{(0)}$

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

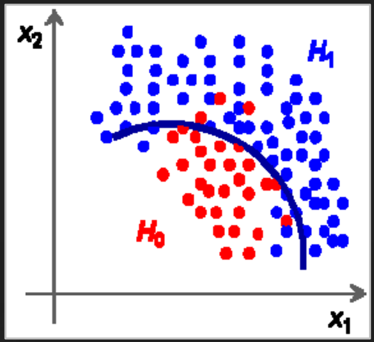


Weight adjustment using analytical back-propagation

- Three different implementations in TMVA (all are Multilayer Perceptrons)
  - **TMlpANN:** Interface to ROOT’s MLP implementation
  - **MLP:** TMVA’s own MLP implementation for increased speed and flexibility
  - **CFMlpANN:** ALEPH’s Higgs search ANN, translated from FORTRAN

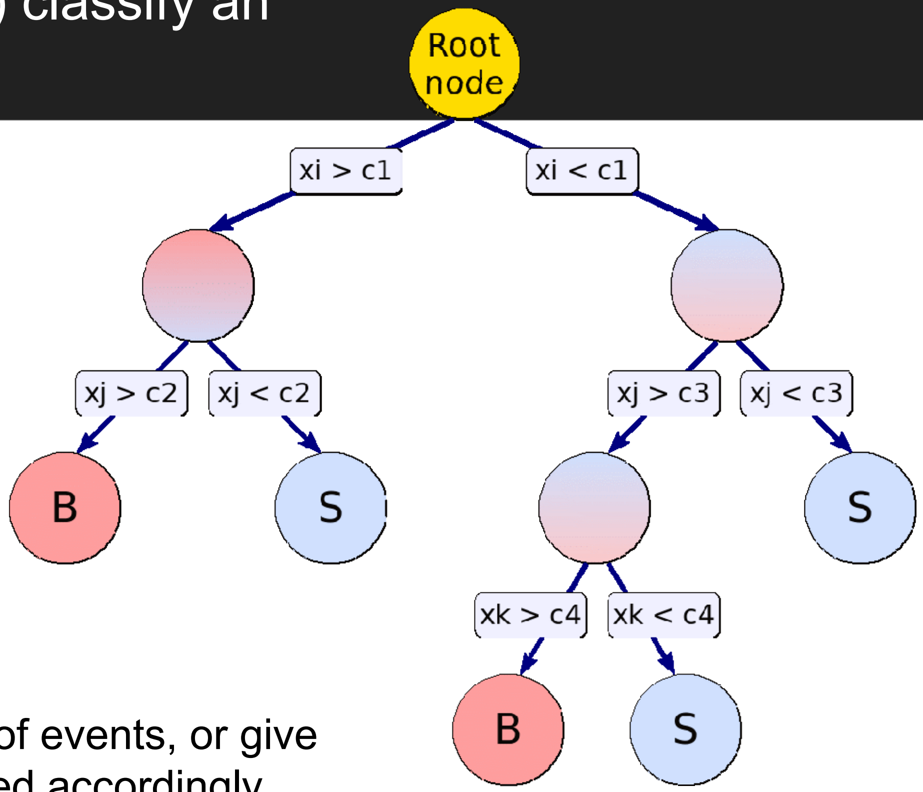
Decision Trees

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



Decision Trees

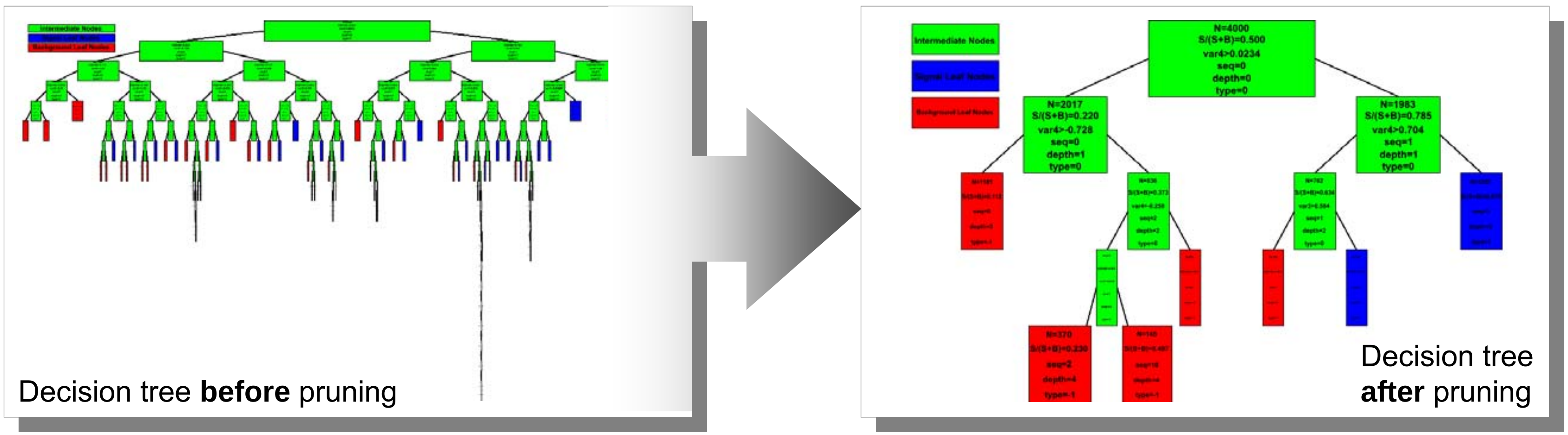
- Sequential application of cuts splits the data into nodes, where the final nodes (leafs) classify an event as signal or background



- 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

Decision Trees

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



Decision tree **before** pruning

Decision tree **after** pruning

- Bottom-up "pruning" of a decision tree
  - Remove statistically insignificant nodes to reduce tree overtraining

Boosted Decision Trees (BDT)

- Data mining with decision trees is popular in science (so far mostly outside of HEP)
  - Advantages:
    - 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 → r_m=1 if all cuts satisfied, =0 otherwise)

normalised discriminating event variables

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

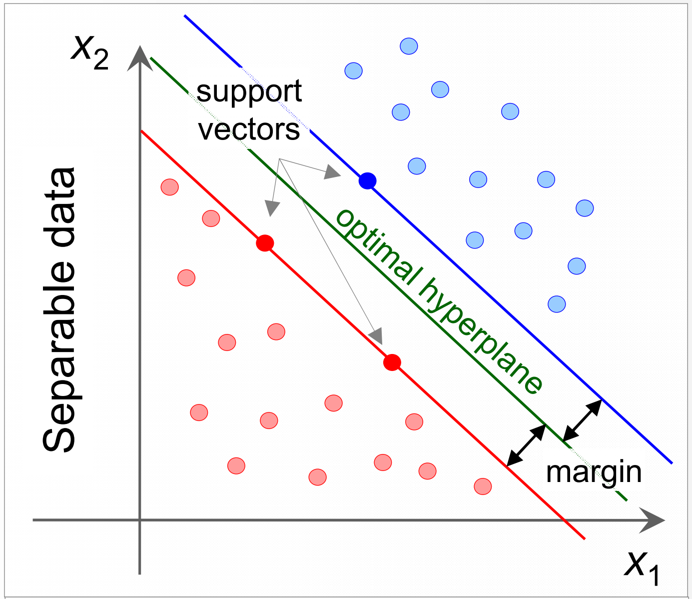
- The problem to solve is
  - Create rule ensemble: use forest of decision trees
  - Fit coefficients a_m, b_k: gradient direct regularization minimising *Risk* (Friedman et al.)

- Pruning removes topologically equal rules" (same variables in cut sequence)

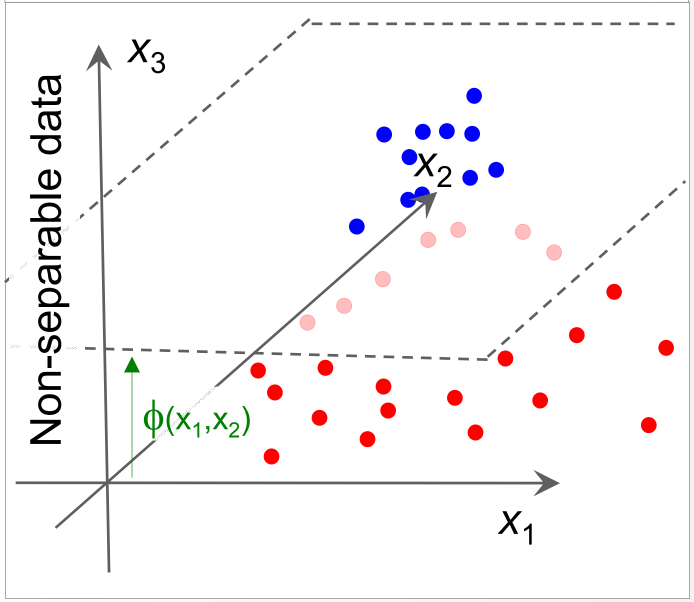
One of the elementary cellular automaton rules (Wolfram 1983, 2002). It specifies the next color in a cell, depending on its color and its immediate neighbors. Its rule outcomes are encoded in the binary representation $30=00011110_2$.

Support Vector Machine (SVM)

- Linear case: 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 dim. space where a linear boundary can fully separate the data
  - Explicit transformation not required: use kernel functions to approximate scalar products between transformed vectors in the higher dim. space
  - Choose Kernel and fit the hyperplane using the techniques developed for linear case

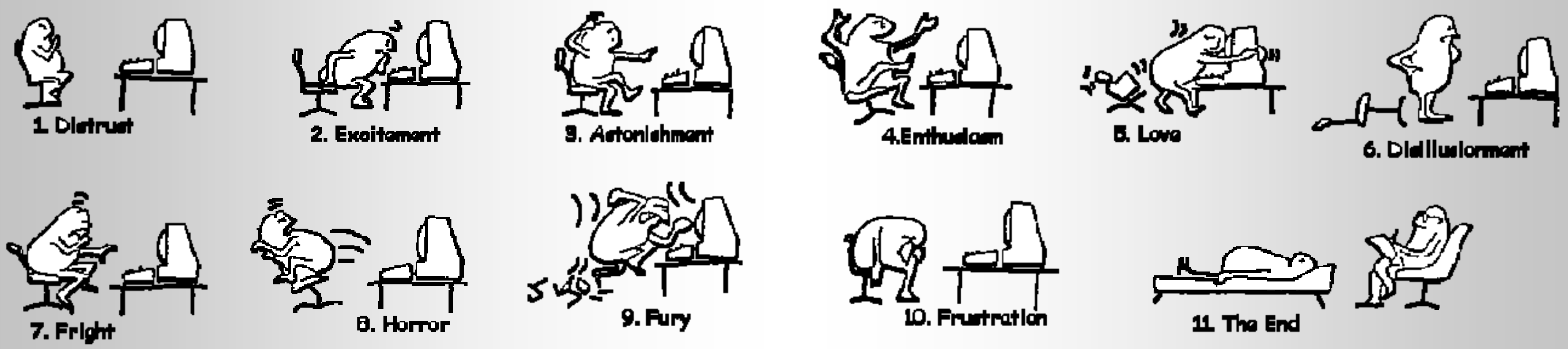


Using *T*MVA

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

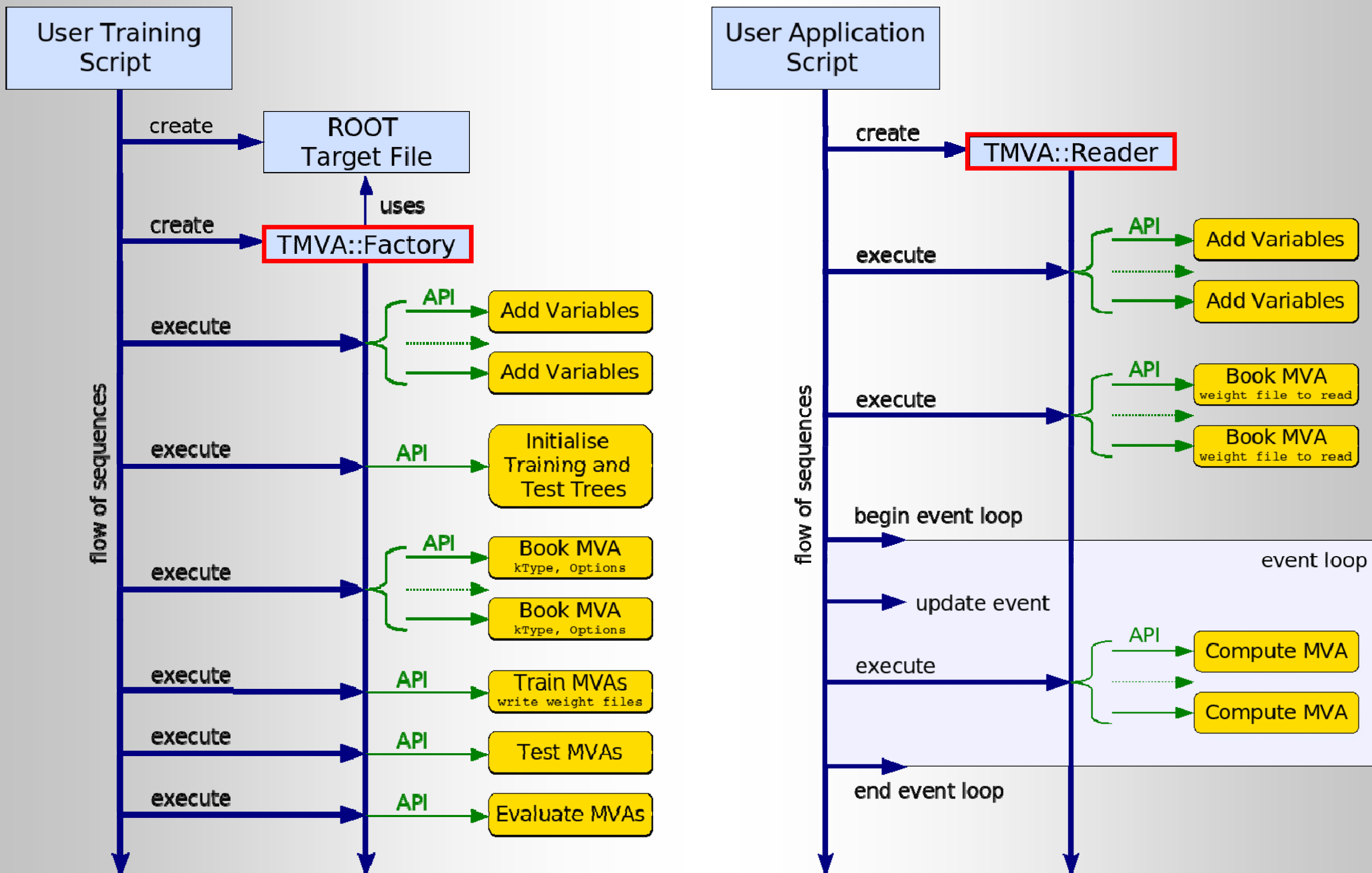
1. *Training phase*: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. *Application phase*: using selected trained classifiers to classify unknown data samples

➡ Illustration of these steps with toy data samples



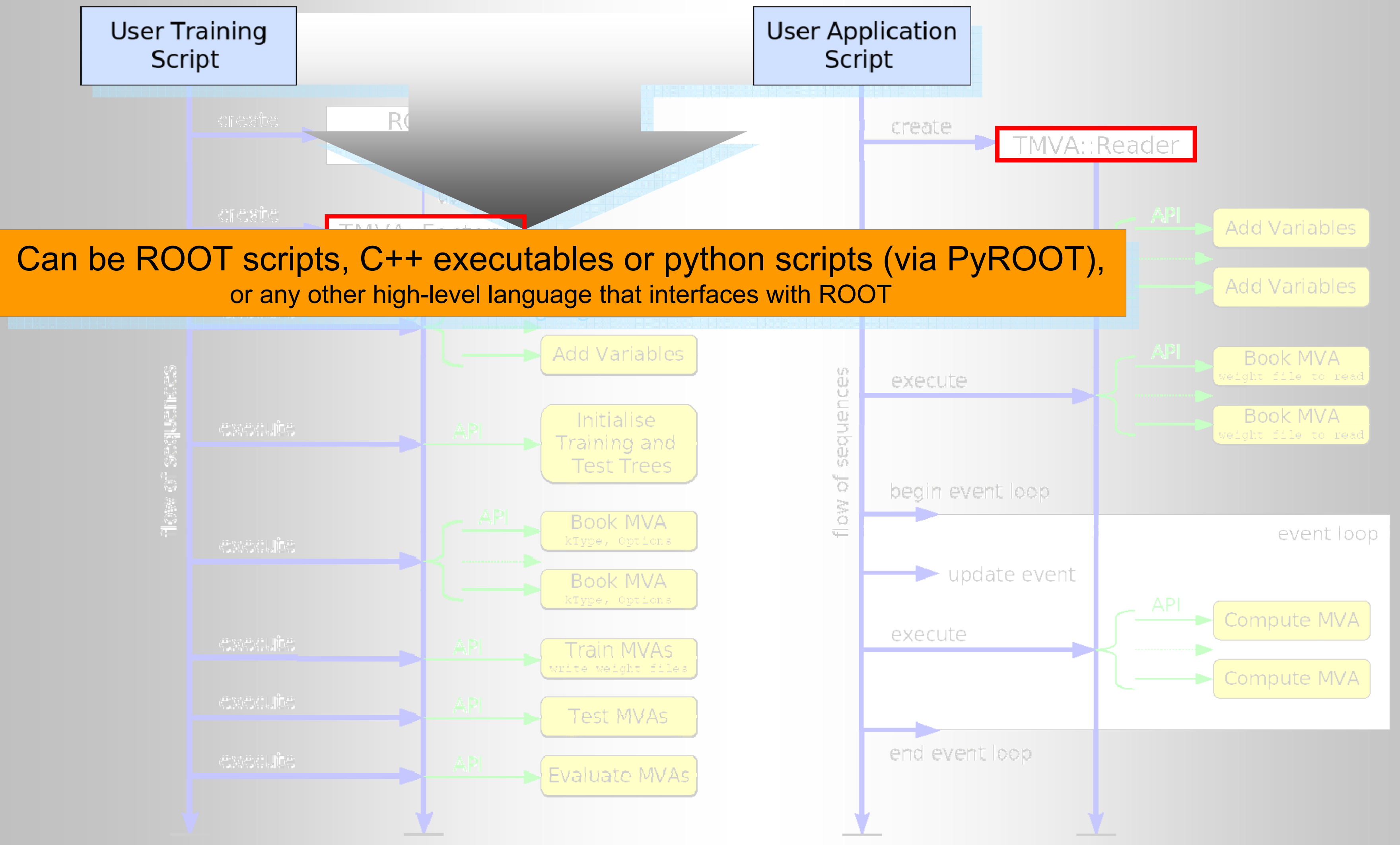
→ *T*MVA tutorial

Code Flow for *Training* and *Application* Phases



→ TMVA tutorial

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

→ *T*MVA tutorial

A Simple Example for *Training*

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

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

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

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

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

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

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

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

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

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

← create *Factory*

← give training/test trees

← register input variables

← select MVA methods

← train, test and evaluate

→ TMVA tutorial

A Simple Example for an *Application*

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

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

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

  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);

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

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

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

← create *Reader*

← register the variables

← book classifier(s)

← prepare event loop

← compute input variables

← calculate classifier output

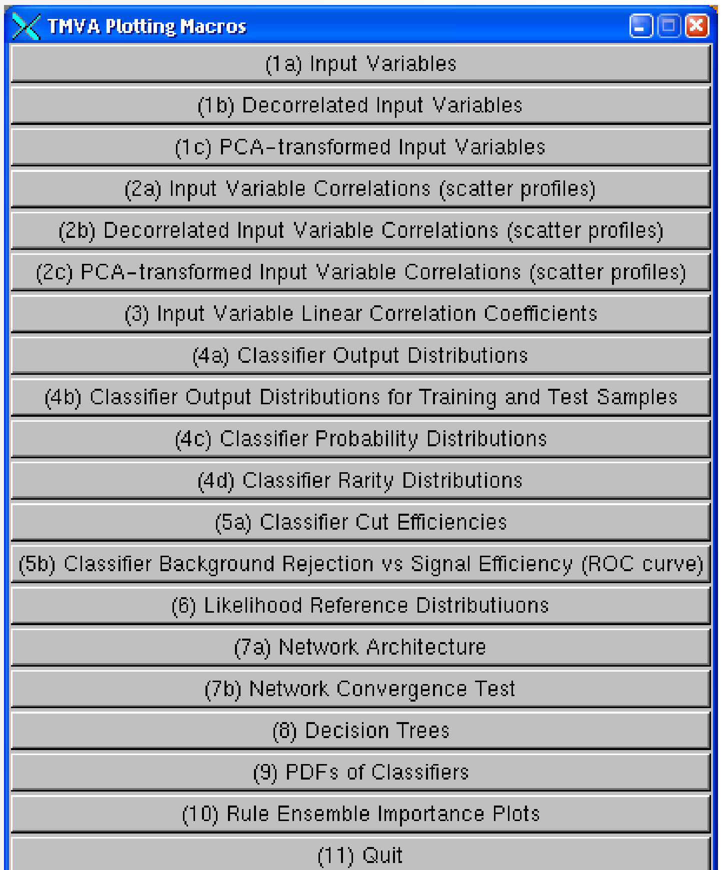
→ *T*MVA tutorial

Data Preparation

- Data input format: ROOT TTree or ASCII
- Supports selection of any subset or combination or function of available variables
- Supports application of pre-selection cuts (possibly independent for signal and bkg)
- Supports global event weights for signal or background input files
- Supports use of any input variable as individual event weight
- Supports various methods for splitting into training and test samples:
  - Block wise
  - Randomly
  - Periodically (*i.e*. periodically 3 testing ev., 2 training ev., 3 testing ev, 2 training ev. ….)
  - User defined training and test trees
- Preprocessing of input variables (*e.g.*, decorrelation)

MVA Evaluation Framework

- TMVA is not only a collection of classifiers, but an MVA framework
- After training, TMVA provides ROOT evaluation scripts (through GUI)



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

Correlation scatters and linear coefficients for S & B

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

Classifier *Rarity* distribution

Classifier significance with optimal cuts

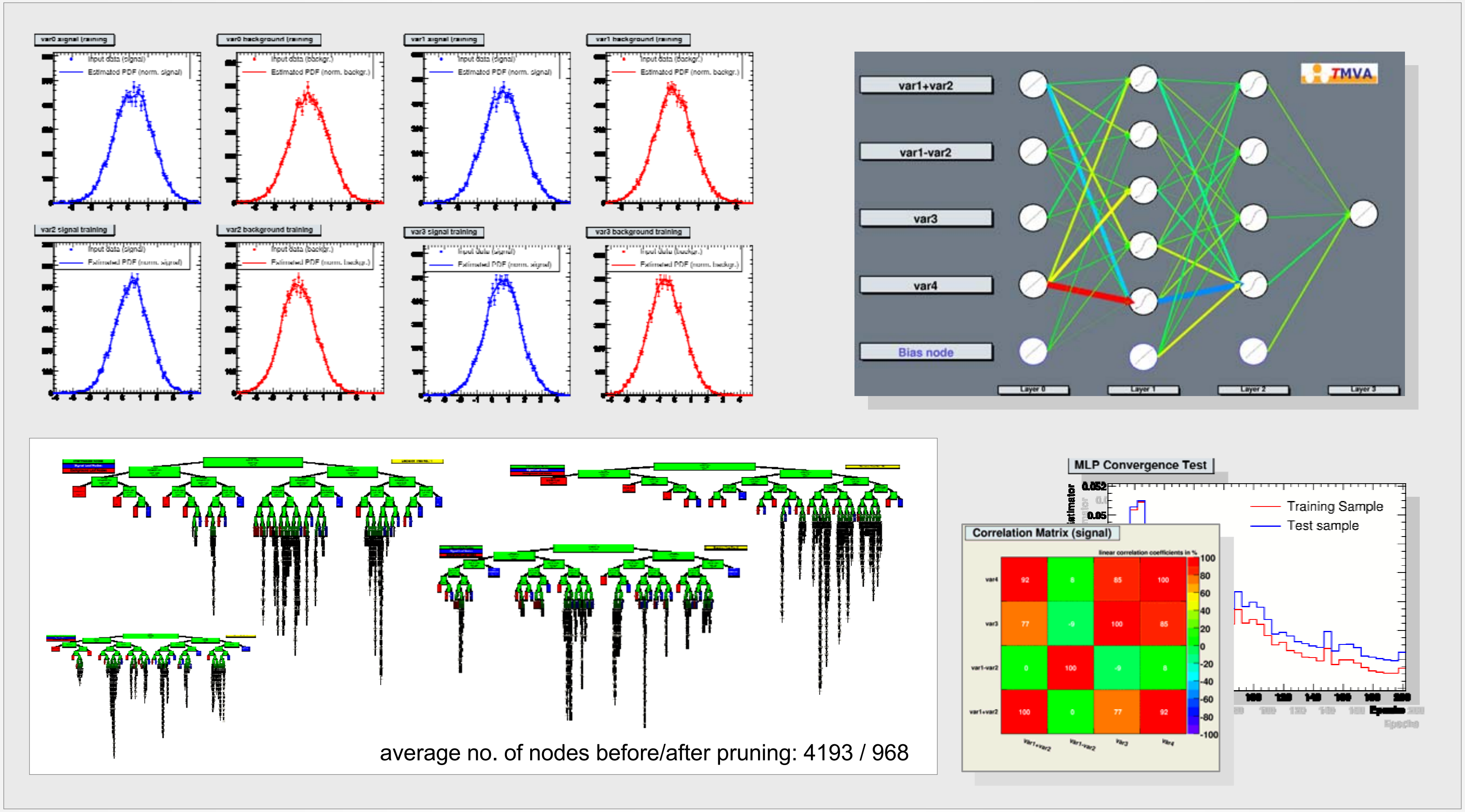
B rejection versus S efficiency

Classifier-specific plots:
- Likelihood reference distributions
- Classifier PDFs (for probability output and Rarity)
- Network architecture, weights and convergence
- Rule Fitting analysis plots

- Visualise decision trees

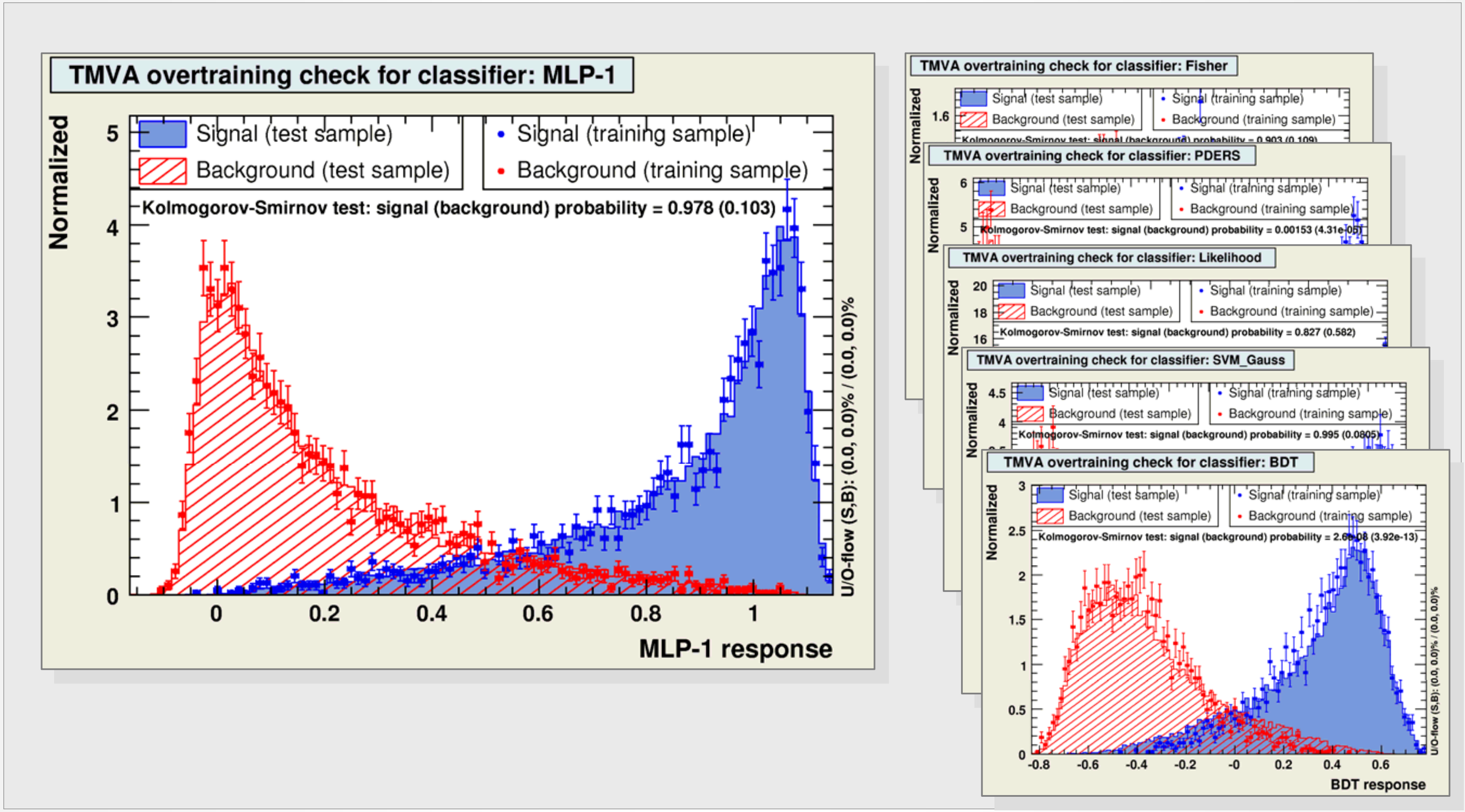
Evaluating the Classifier Training (I)

- Projective likelihood PDFs, MLP training, BDTs, …



Evaluating the Classifier Training (II)

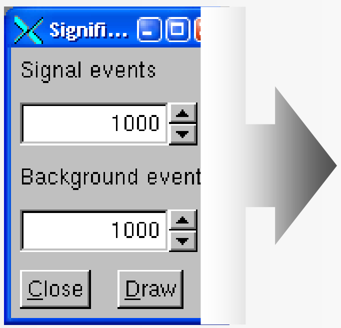
- Classifier output distributions for test *and* training samples …

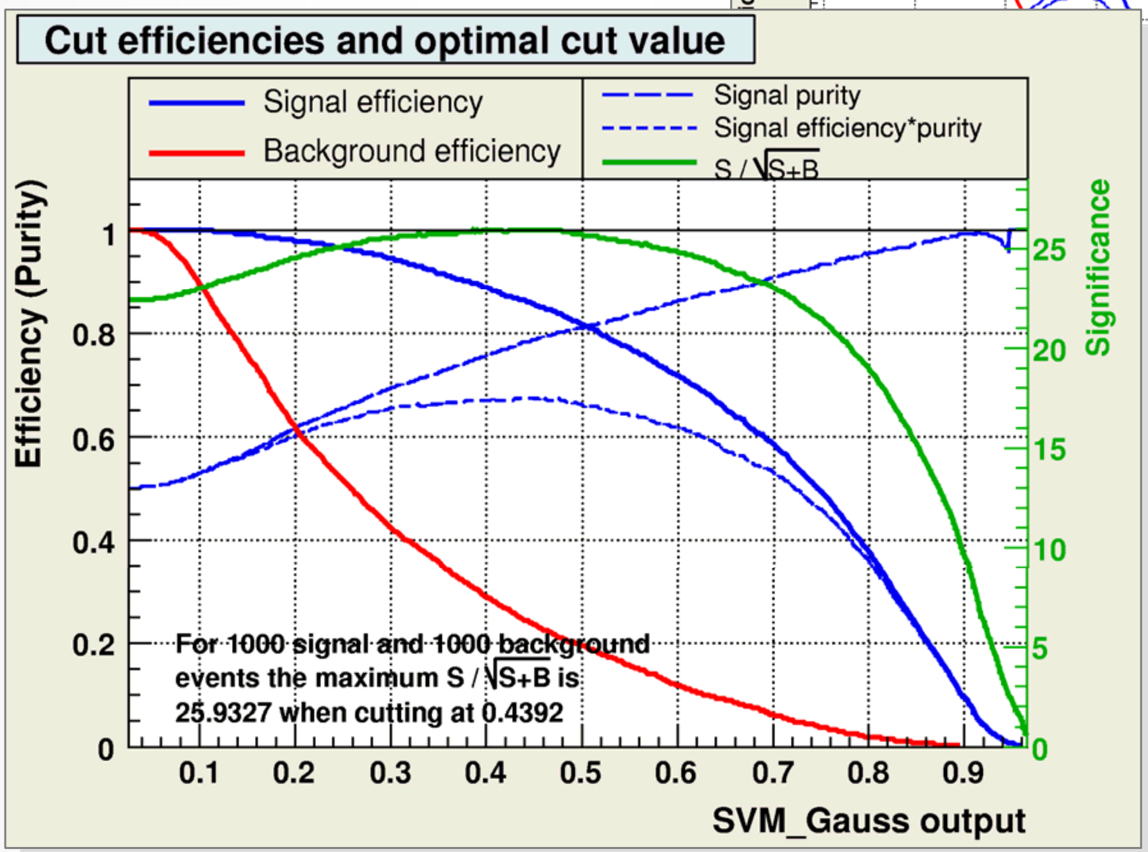


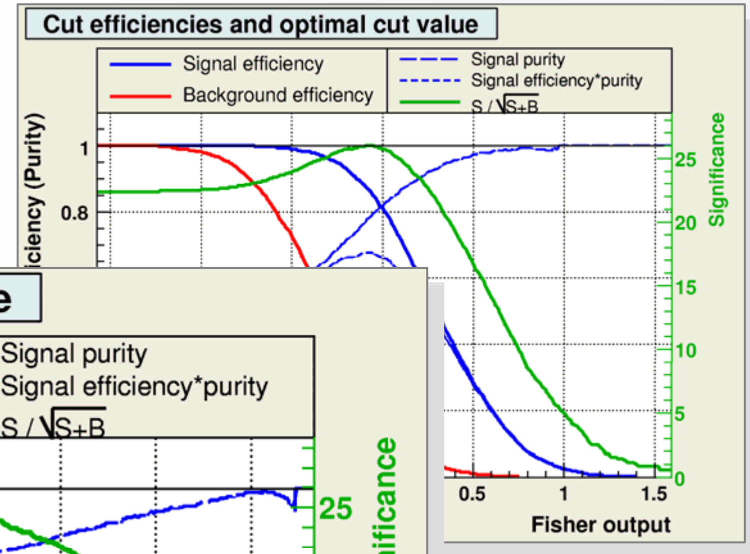
Evaluating the Classifier Training (III)

- Optimal cut for each classifiers …

  Determine the optimal cut (working point) on a classifier output



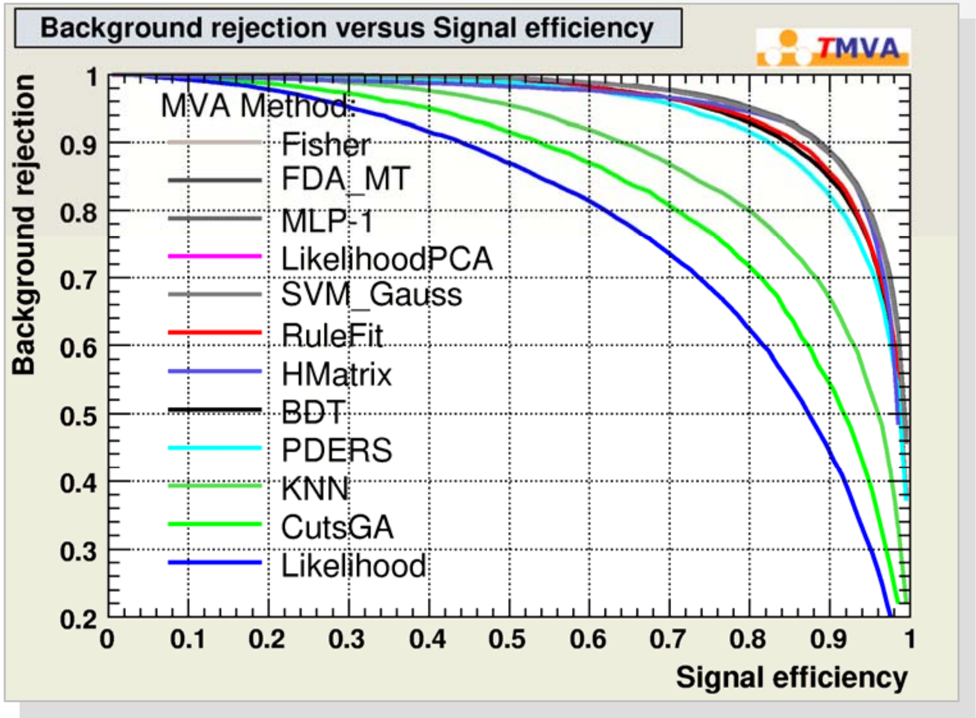


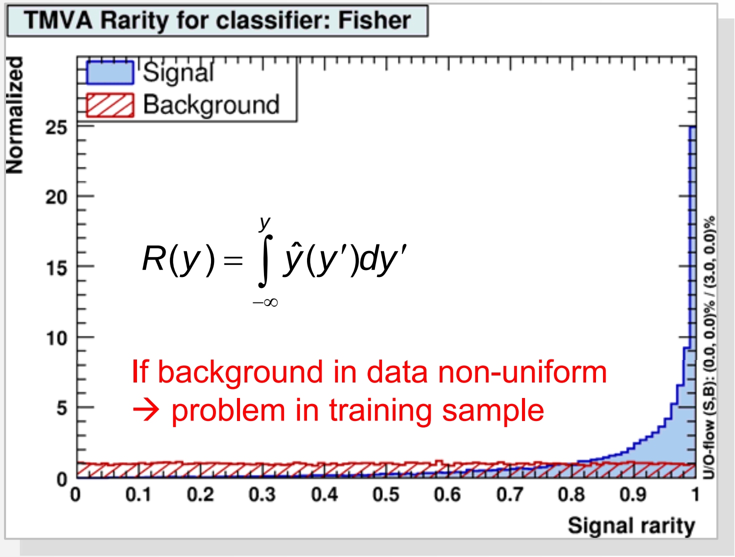


Evaluating the Classifier Training (IV)

- Background rejection versus signal efficiencies …

Best plot to compare classifier performance





An elegant variable is the *Rarity*: transforms to uniform background. Height of signal peak direct measure of classifier performance

Evaluating the Classifiers (taken from TMVA output…)

Input Variable Ranking

Better variable

```
--- Fisher       : Ranking result (top variable is best ranked)
--- Fisher       : ---------------------------------------------
--- Fisher       : Rank : Variable   : Discr. power
--- Fisher       : ---------------------------------------------
--- Fisher       :    1 : var4       : 2.175e-01
--- Fisher       :    2 : var3       : 1.718e-01
--- Fisher       :    3 : var1       : 9.549e-02
--- Fisher       :    4 : var2       : 2.841e-02
--- Fisher       : ---------------------------------------------
```

- How discriminating is a variable ?

Classifier correlation and overlap

```
--- Factory      : Inter-MVA overlap matrix (signal):
--- Factory      : ------------------------------
--- Factory      :             Likelihood  Fisher
--- Factory      : Likelihood:     +1.000  +0.667
--- Factory      :     Fisher:     +0.667  +1.000
--- Factory      : ------------------------------
```

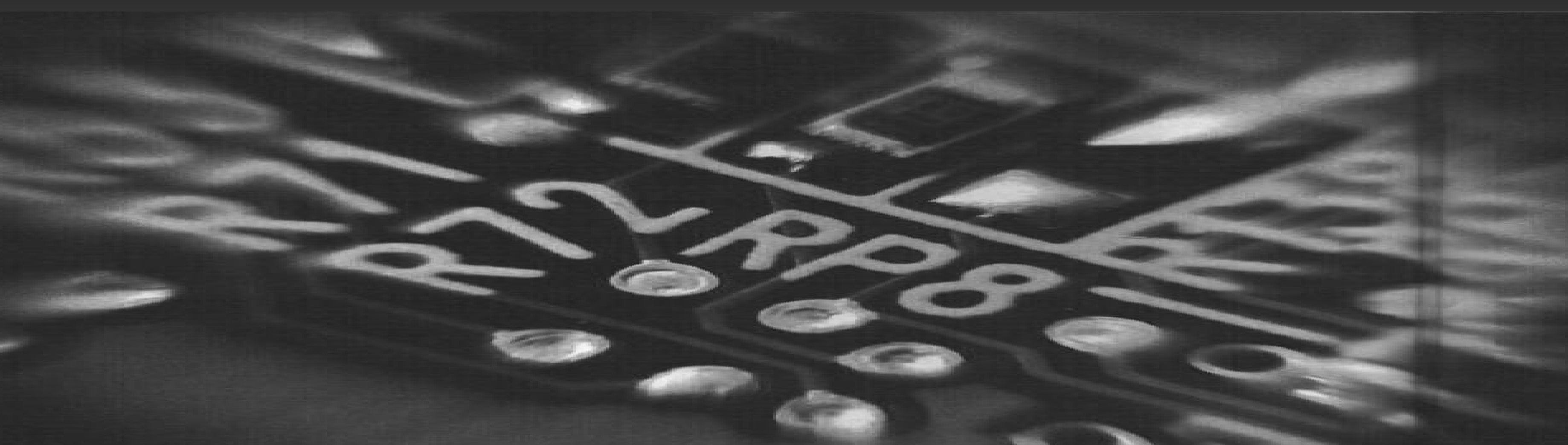
- Do classifiers select the same events as signal and background ? If not, there is something to gain !

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): @B=0.01	@B=0.10	@B=0.30	Area	Sepa-ration:	Signifi-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…)

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

MVA Methods:	Signal efficiency at bkg eff. (error): @B=0.01	@B=0.10	@B=0.30	Area	Separation:	Significance:
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

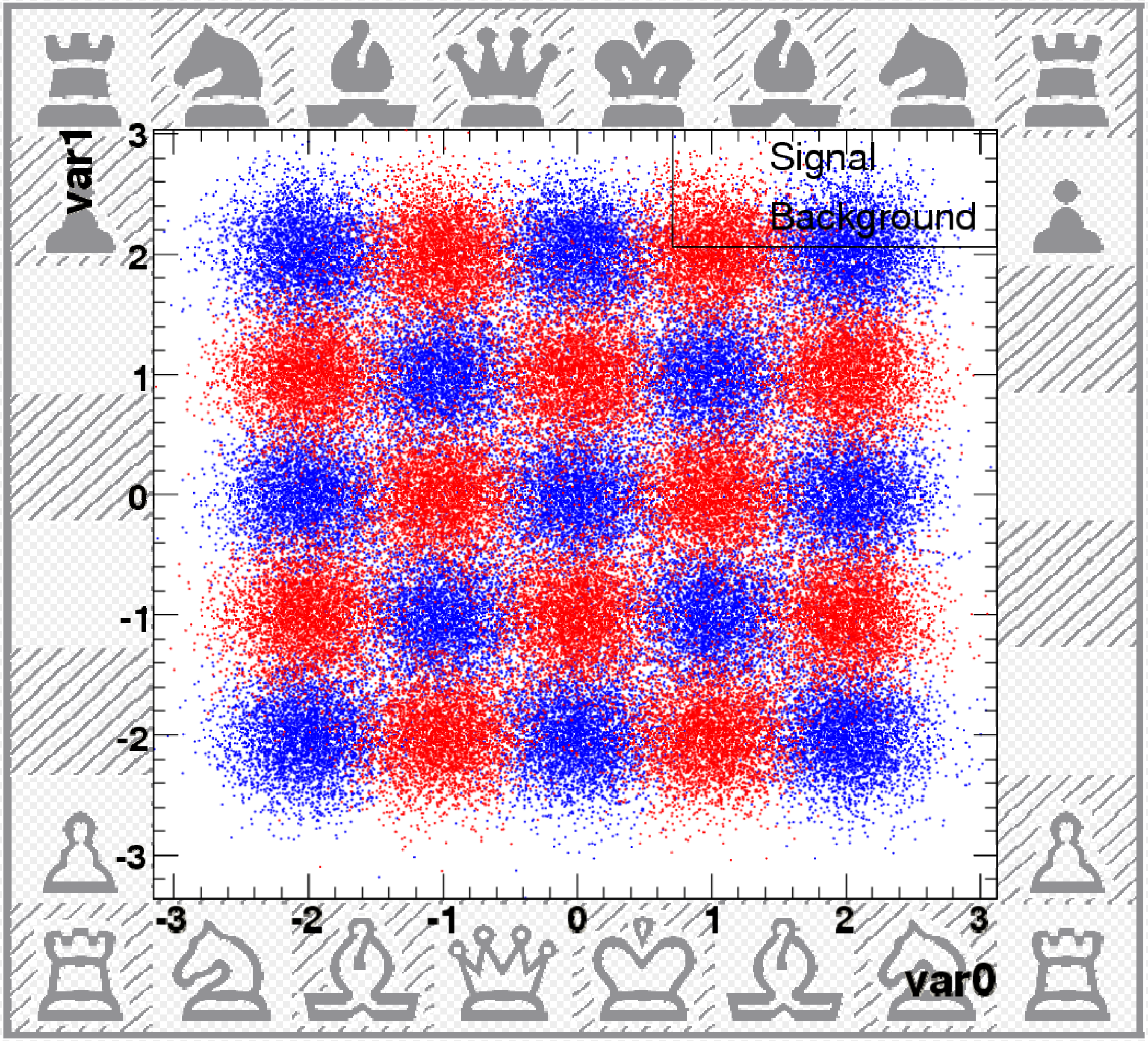
Better classifier

Testing efficiency compared to training efficiency (overtraining check)

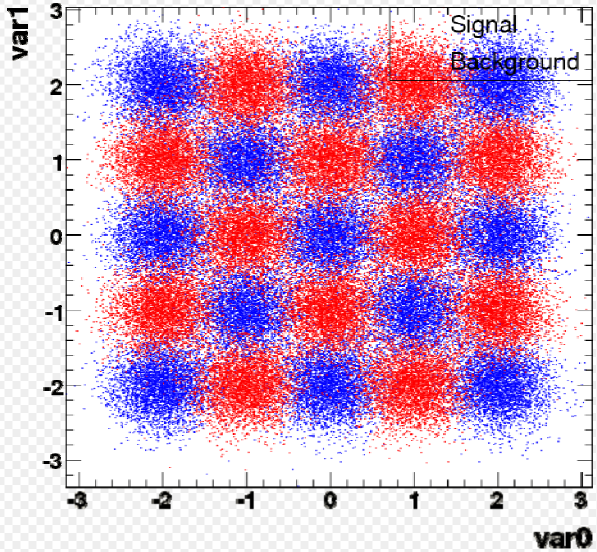
MVA Methods:	Signal efficiency: from test sample (from traing sample) @B=0.01	@B=0.10	@B=0.30
Fisher	0.268 (0.275)	0.653 (0.658)	0.873 (0.873)
MLP	0.266 (0.278)	0.656 (0.658)	0.873 (0.873)
LikelihoodD	0.259 (0.273)	0.649 (0.657)	0.871 (0.872)
PDERS	0.223 (0.389)	0.628 (0.691)	0.861 (0.881)
RuleFit	0.196 (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)

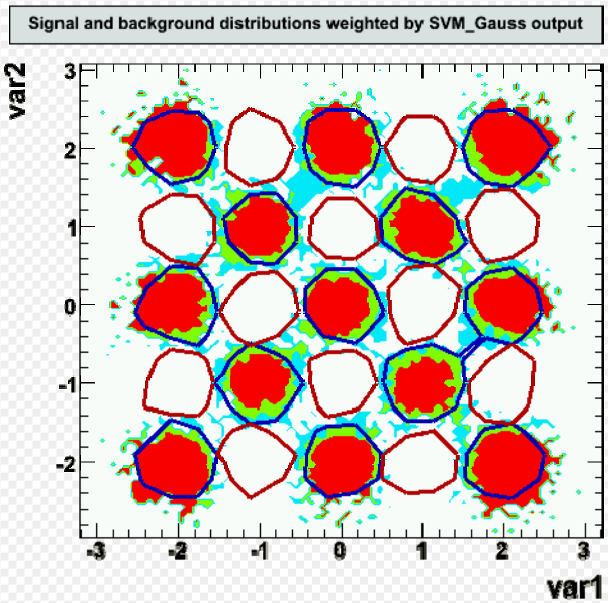
Check for over-training

The “Schachbrett” Toy

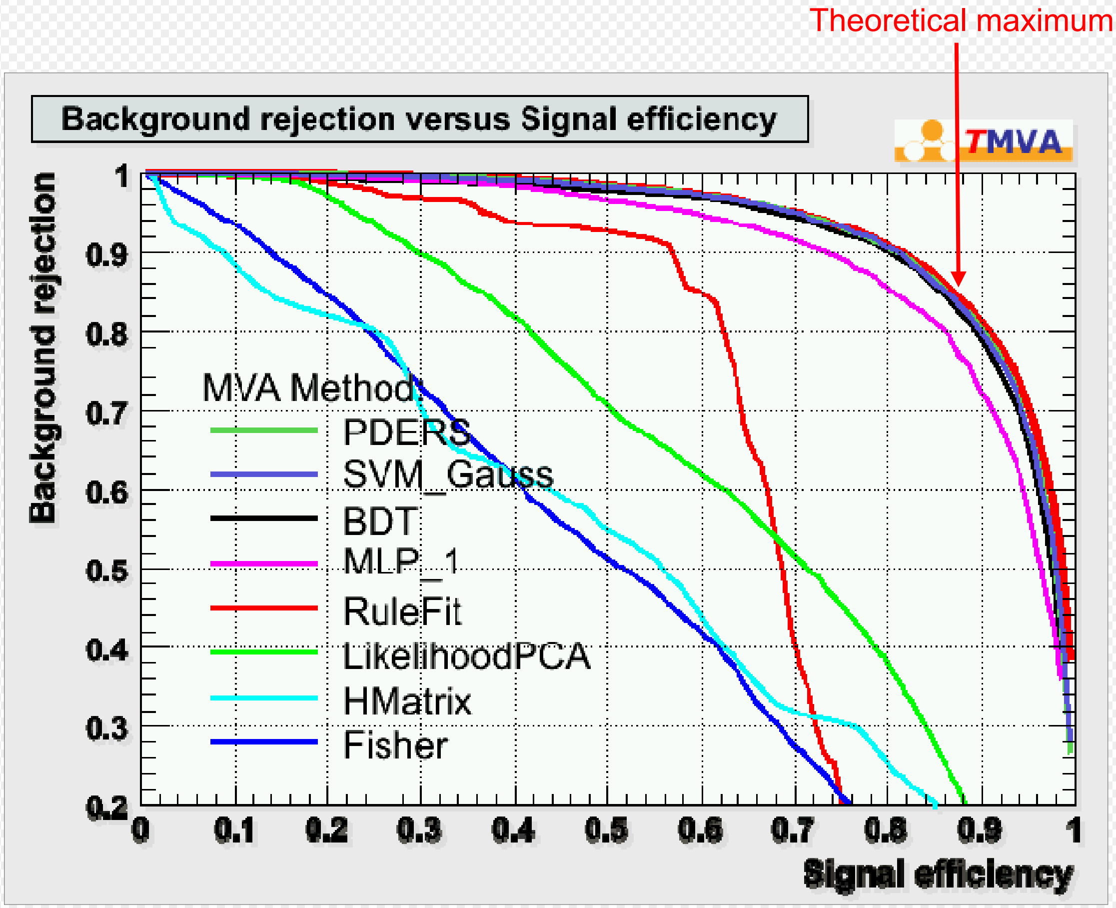


The “Schachbrett” Toy





- 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		Cuts	Likeli-hood	PDERS / k-NN	H-Matrix	Fisher	MLP	BDT	RuleFit	SVM
Perfor-mance	no / linear correlations	😐	☺	☺	😐	☺	☺	😐	☺	☺
	nonlinear correlations	😐	☹	☺	☹	☹	☺	☺	😐	☺
Speed	Training	☹	☺	☺	☺	☺	😐	☹	😐	☹
	Response	☺	☺	☹ / 😐	☺	☺	☺	😐	😐	😐
Robust-ness	Overtraining	☺	😐	😐	☺	☺	☹	☹	😐	😐
	Weak input variables	☺	☺	☹	☺	☺	😐	😐	😐	😐
Curse of dimensionality		☹	☺	☹	☺	☺	😐	☺	😐	😐
Transparency		☺	☺	😐	☺	☺	☹	☹	☹	☹

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

Outlook

Primary development from last 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
- Categorisation: use any combination of input variables and classifiers in any phase space region
- Code is ready – now in testing mode. Dispatched soon hopefully...

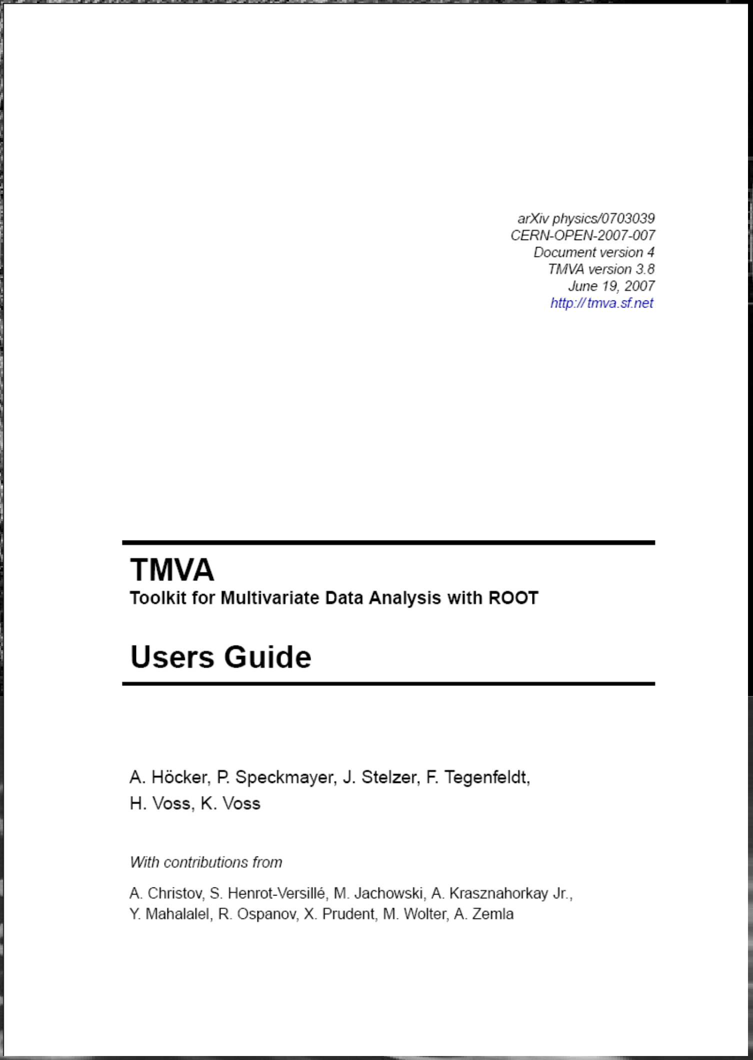
This summer: Extend TMVA to multivariate regression

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

- *T*MVA 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

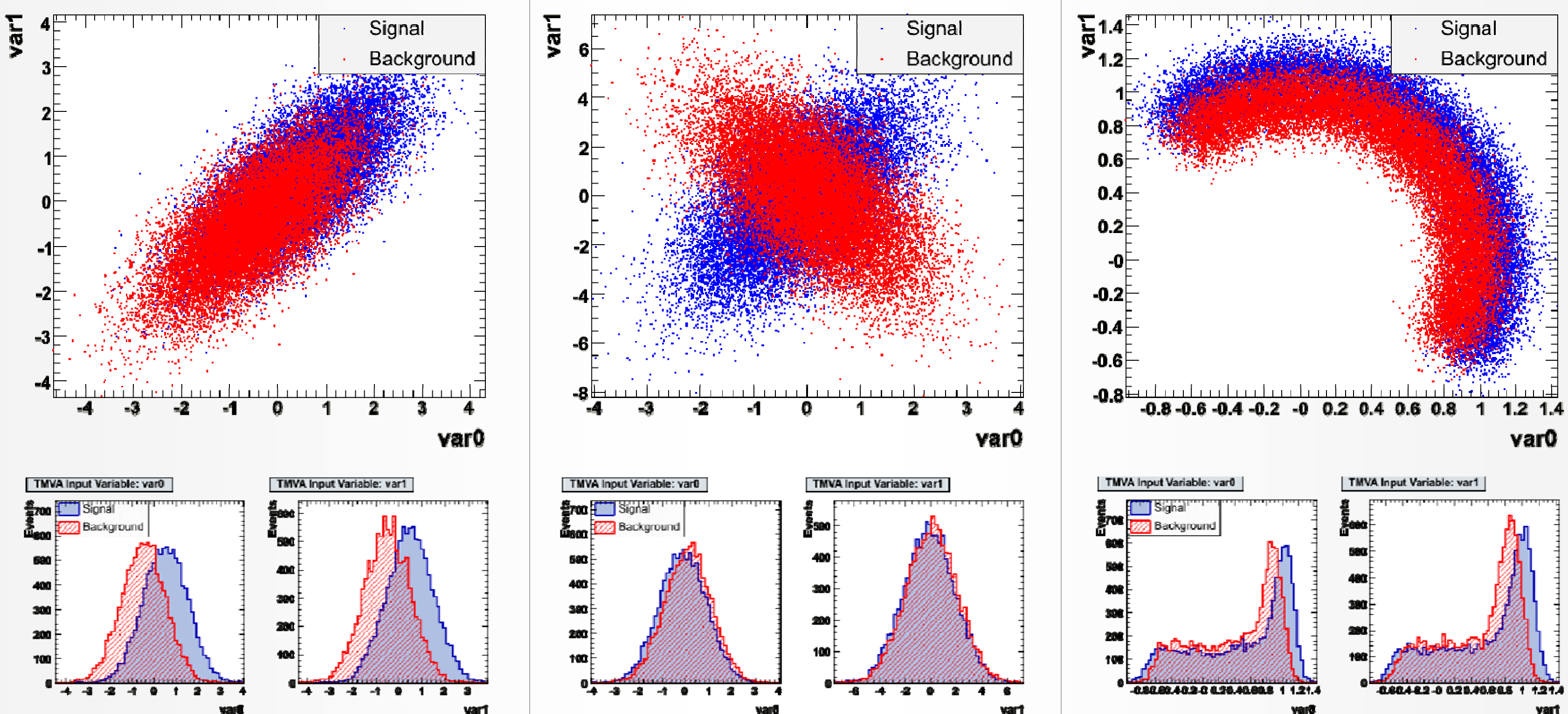
Acknowledgments: The fast development of TMVA would not have been possible without the contribution and feedback from many developers and users to whom we are indebted. We thank in particular the CERN Summer students Matt Jachowski (Stanford) for the implementation of TMVA's new MLP neural network, Yair Mahalalel (Tel Aviv) and three genius Krakow mathematics students for significant improvements 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

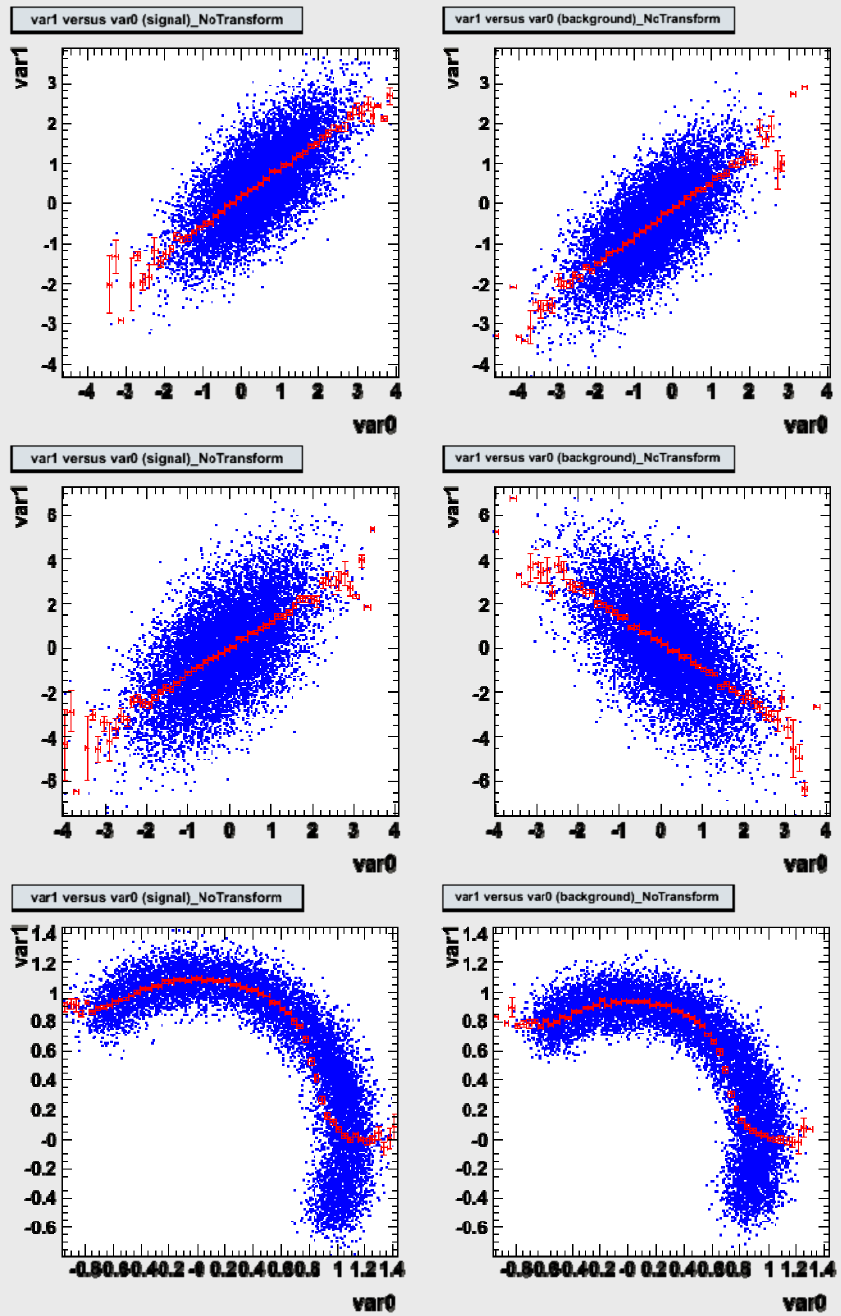
More Toys: Linear-, Cross-, Circular Correlations

- 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)

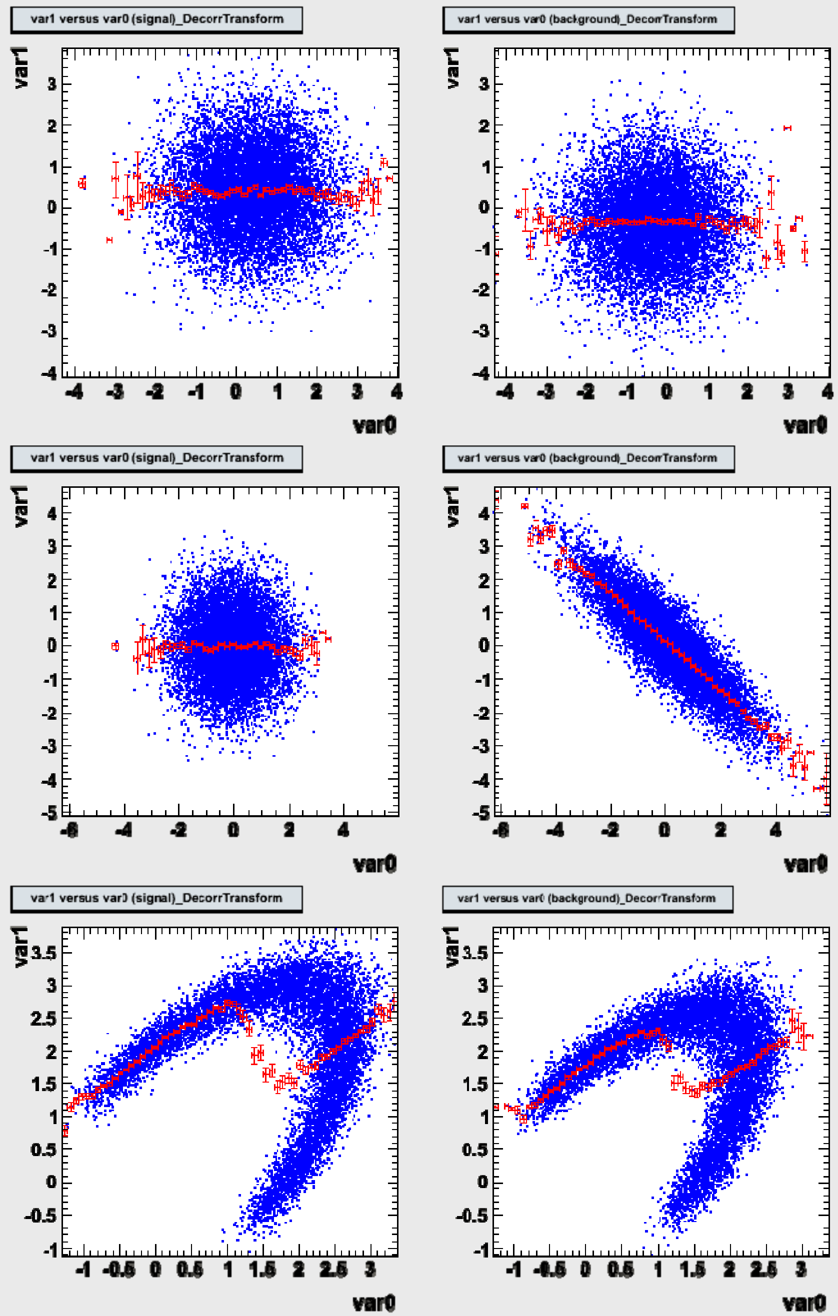


- How does linear decorrelation affect strongly nonlinear cases ?



Original correlations

- How does linear decorrelation affect strongly nonlinear cases ?



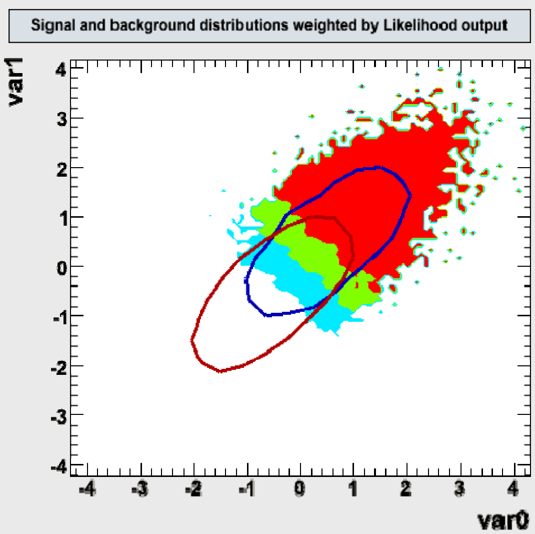
Original correlations

SQRT decorrelation

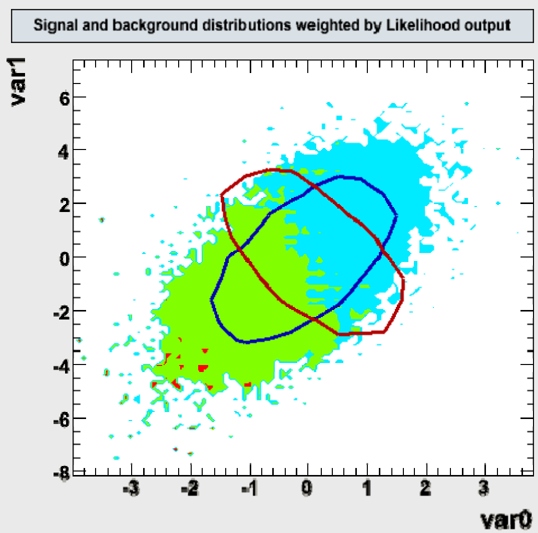
Weight Variables by Classifier Output

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

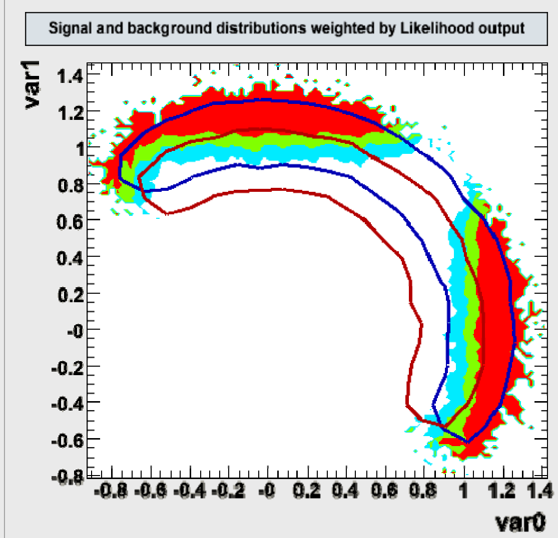
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



Circular correlations
(same for signal and background)

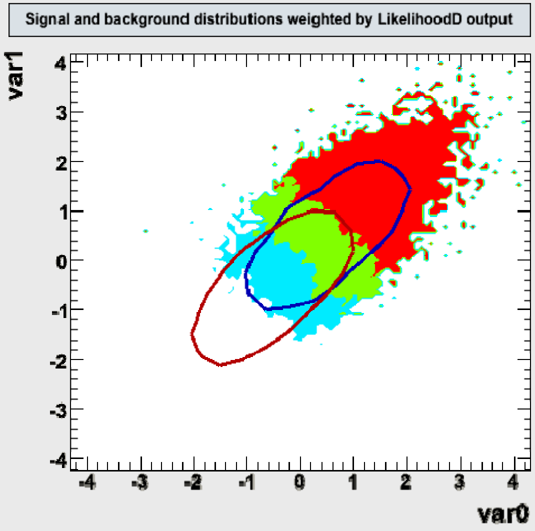


← Likelihood →

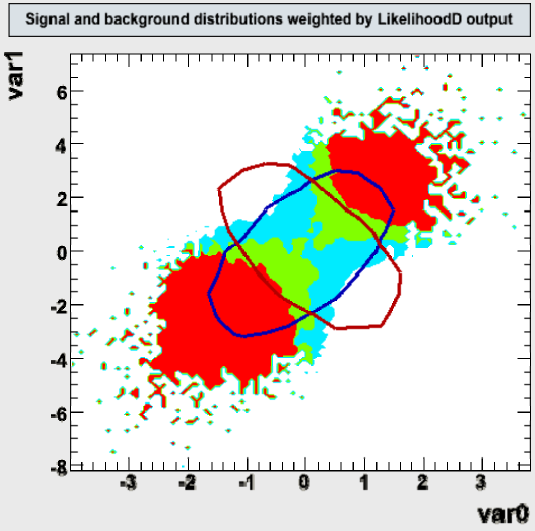
Weight Variables by Classifier Output

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

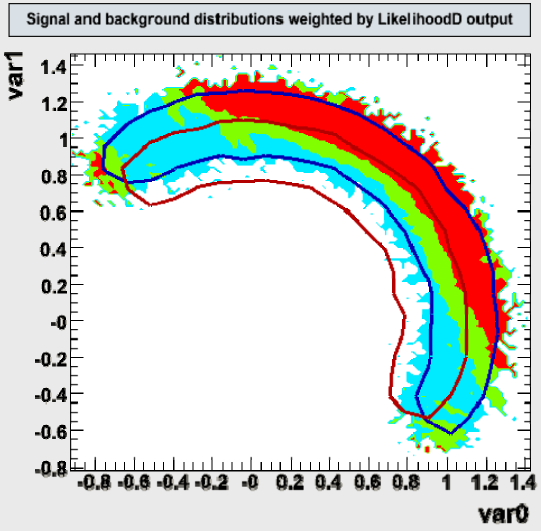
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



Circular correlations
(same for signal and background)

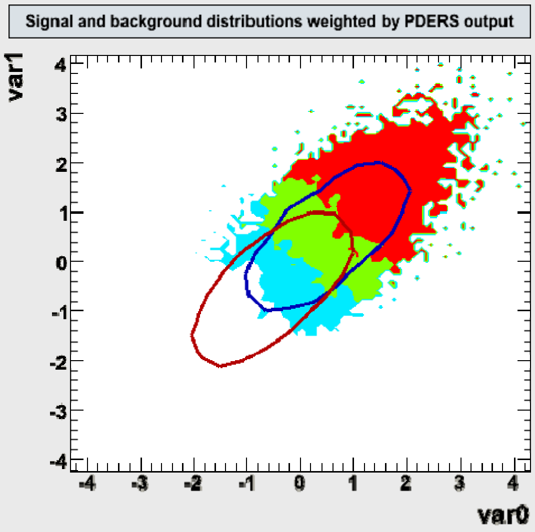


Likelihood - D

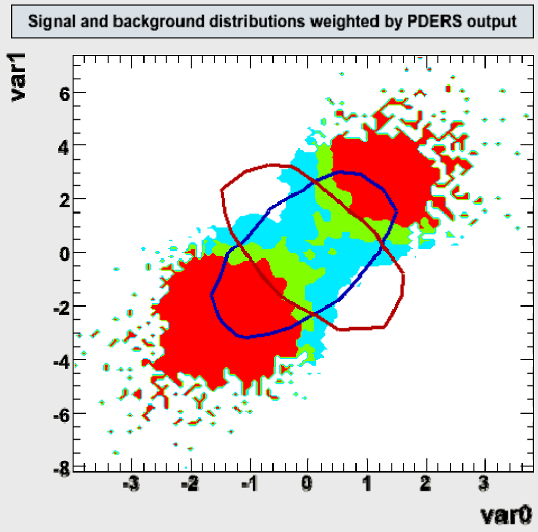
Weight Variables by Classifier Output

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

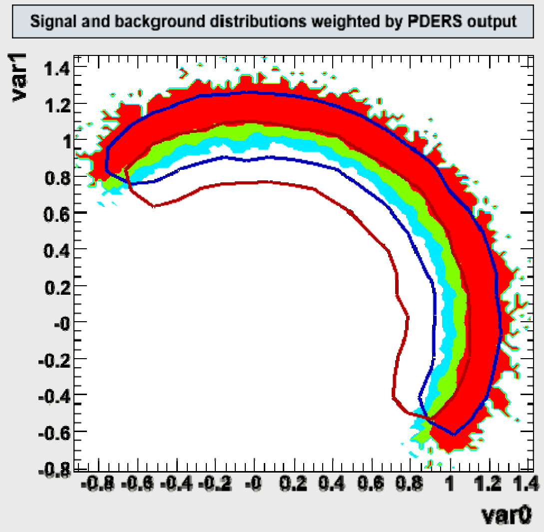
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



Circular correlations
(same for signal and background)

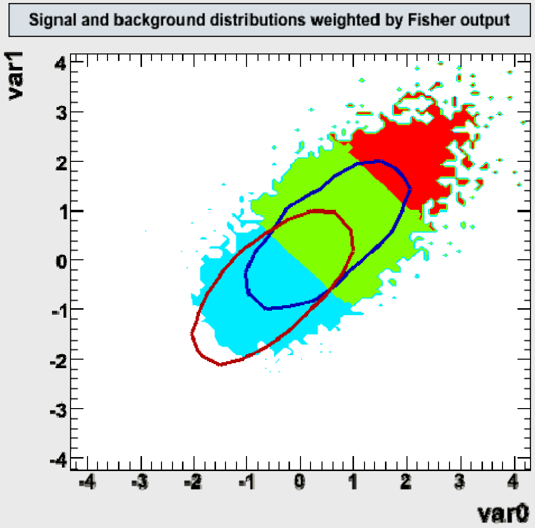


PDERS

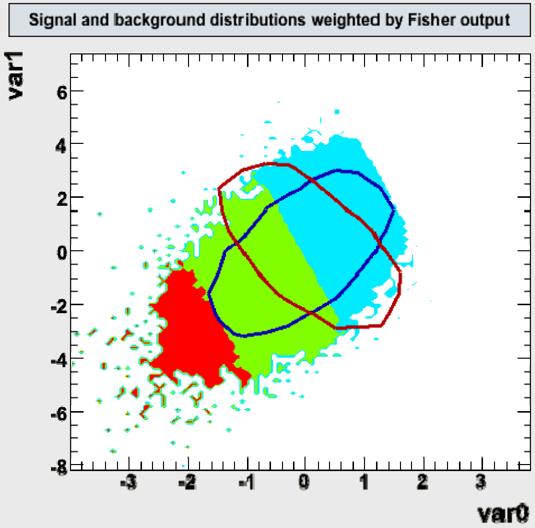
Weight Variables by Classifier Output

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

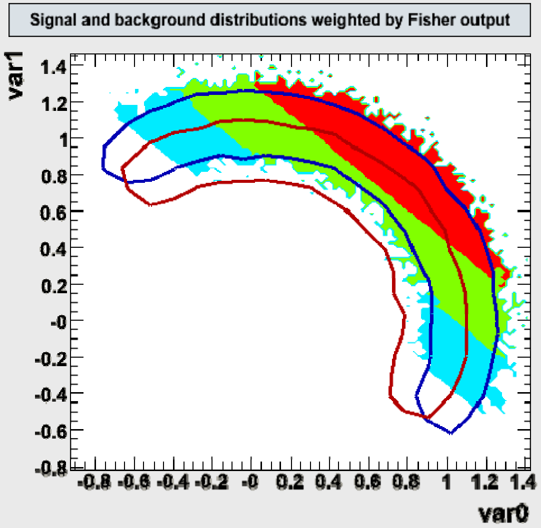
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



Circular correlations
(same for signal and background)



Fisher

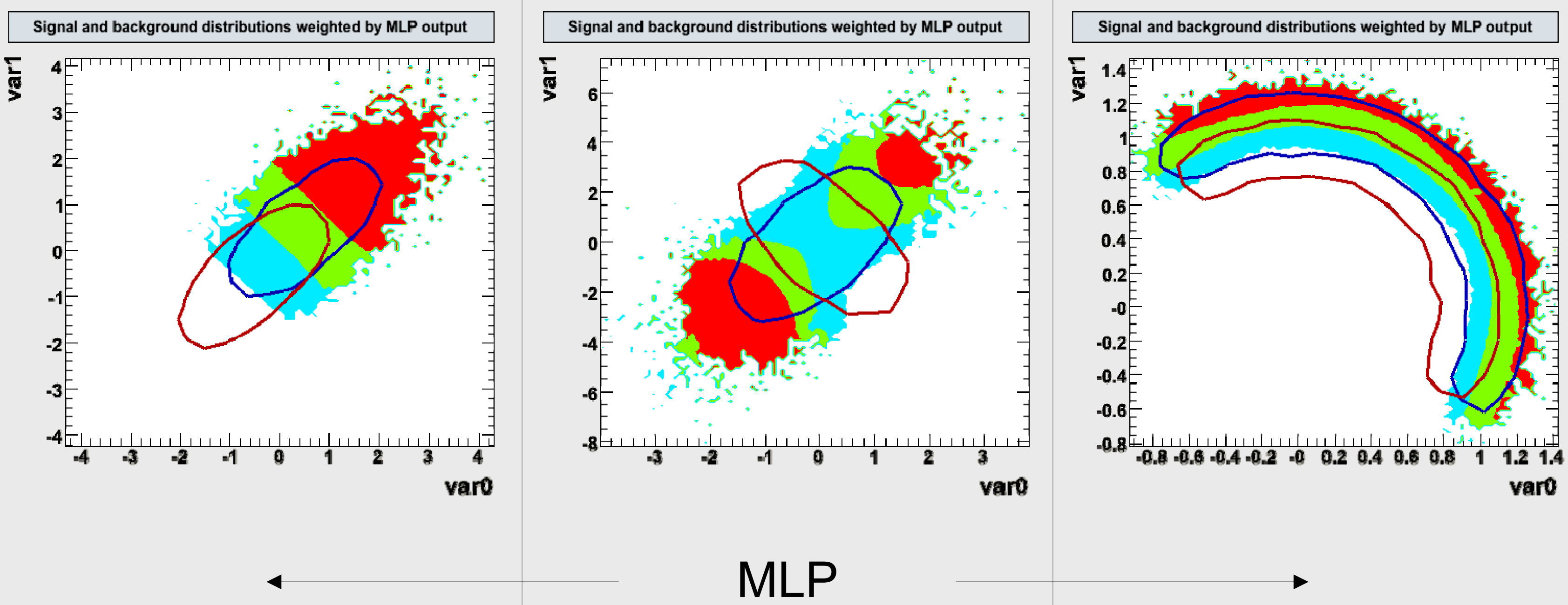
Weight Variables by Classifier Output

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

Linear correlations
(same for signal and background)

Cross-linear correlations
(opposite for signal and background)

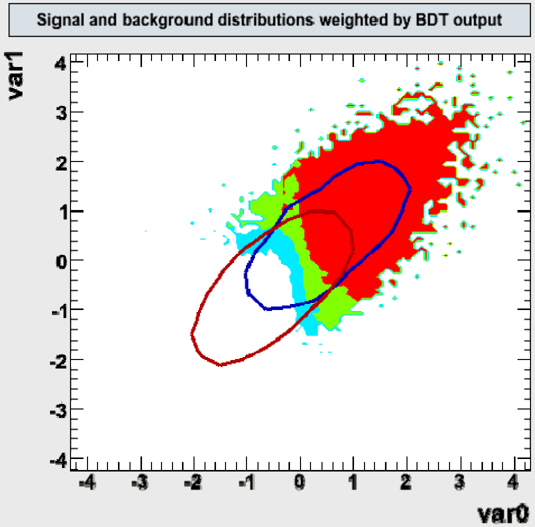
Circular correlations
(same for signal and background)

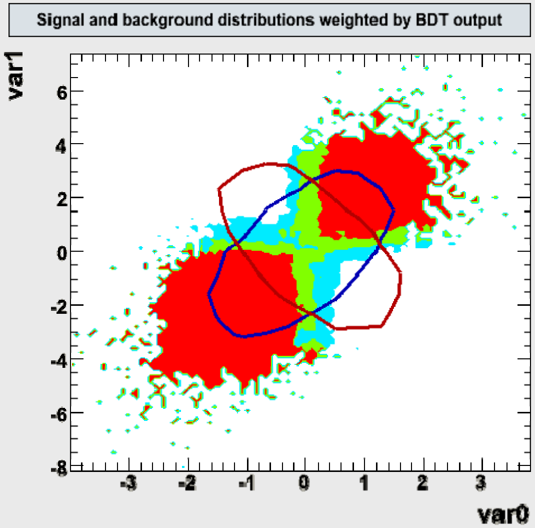


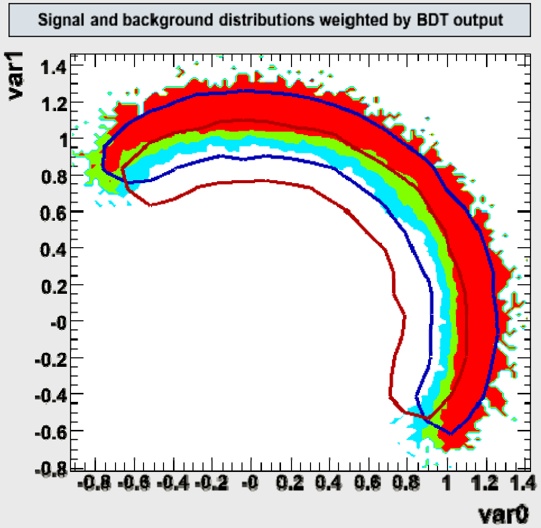
Weight Variables by Classifier Output

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

Linear correlations (same for signal and background)	Cross-linear correlations (opposite for signal and background)	Circular correlations (same for signal and background)



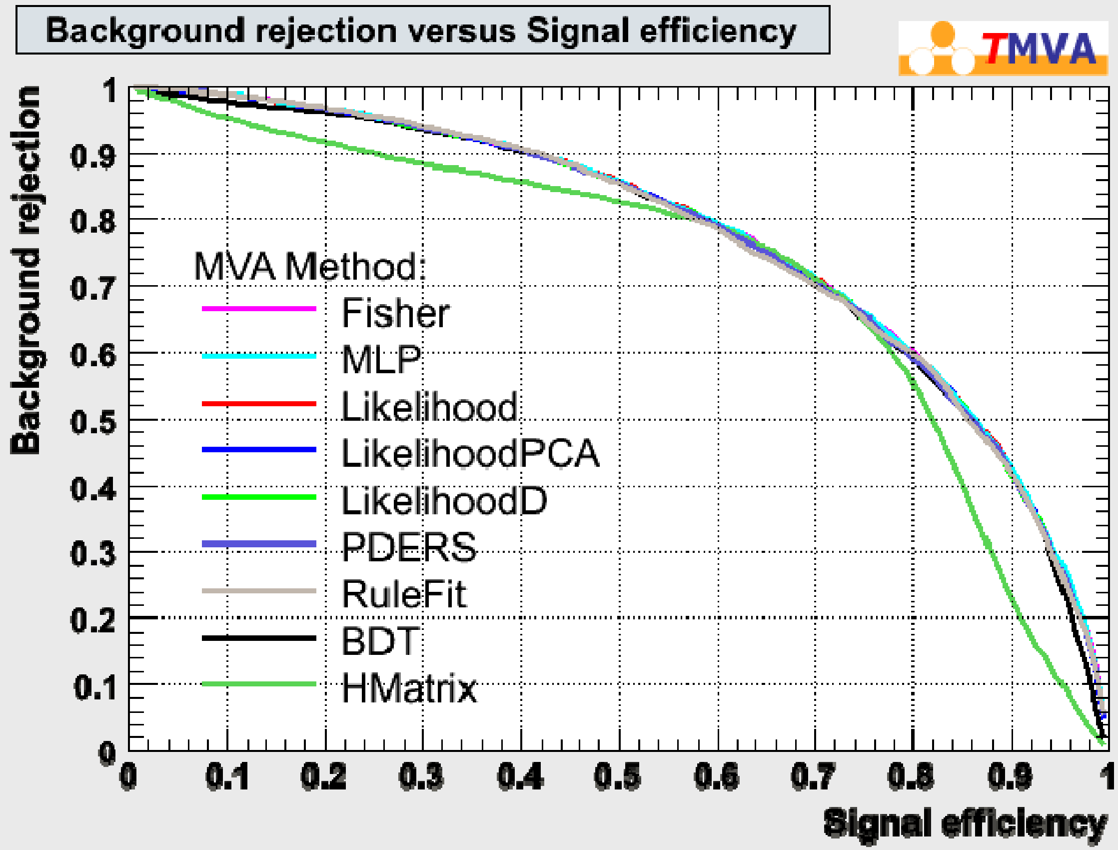




BDT

Final Classifier Performance

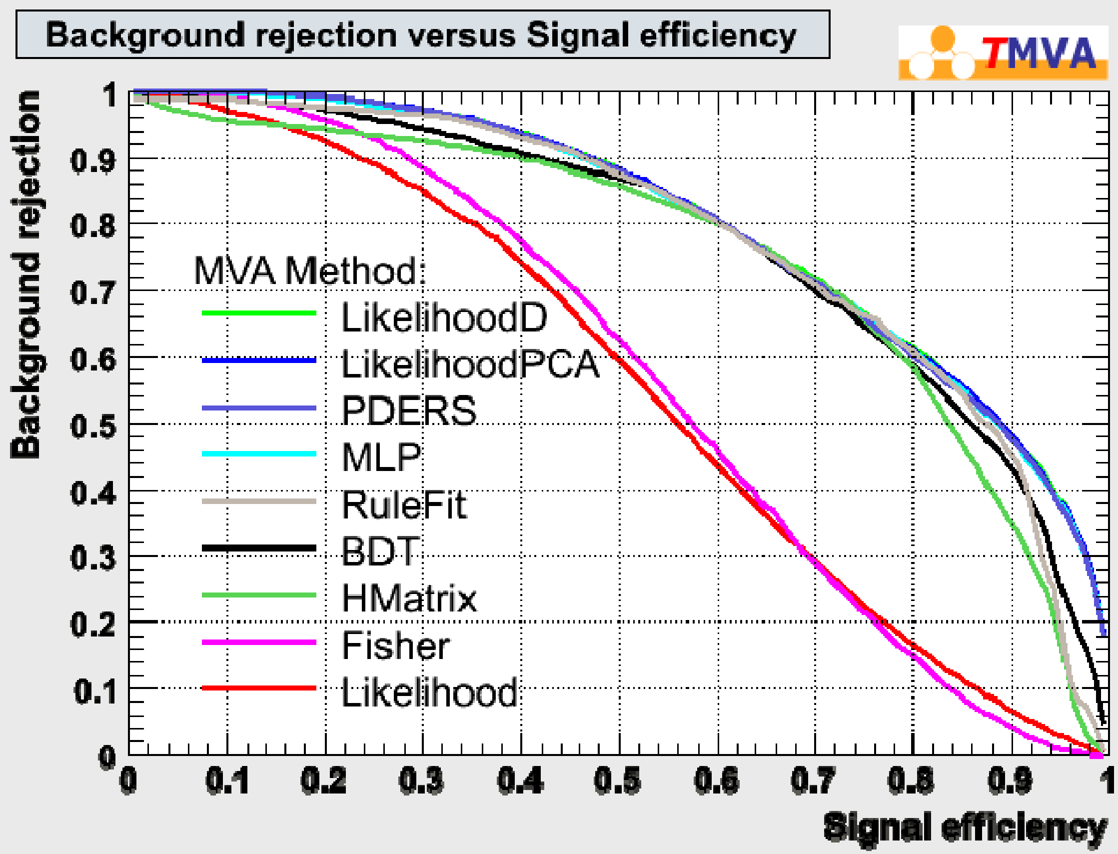
- Background rejection versus signal efficiency curve:



Circular Example

Final Classifier Performance

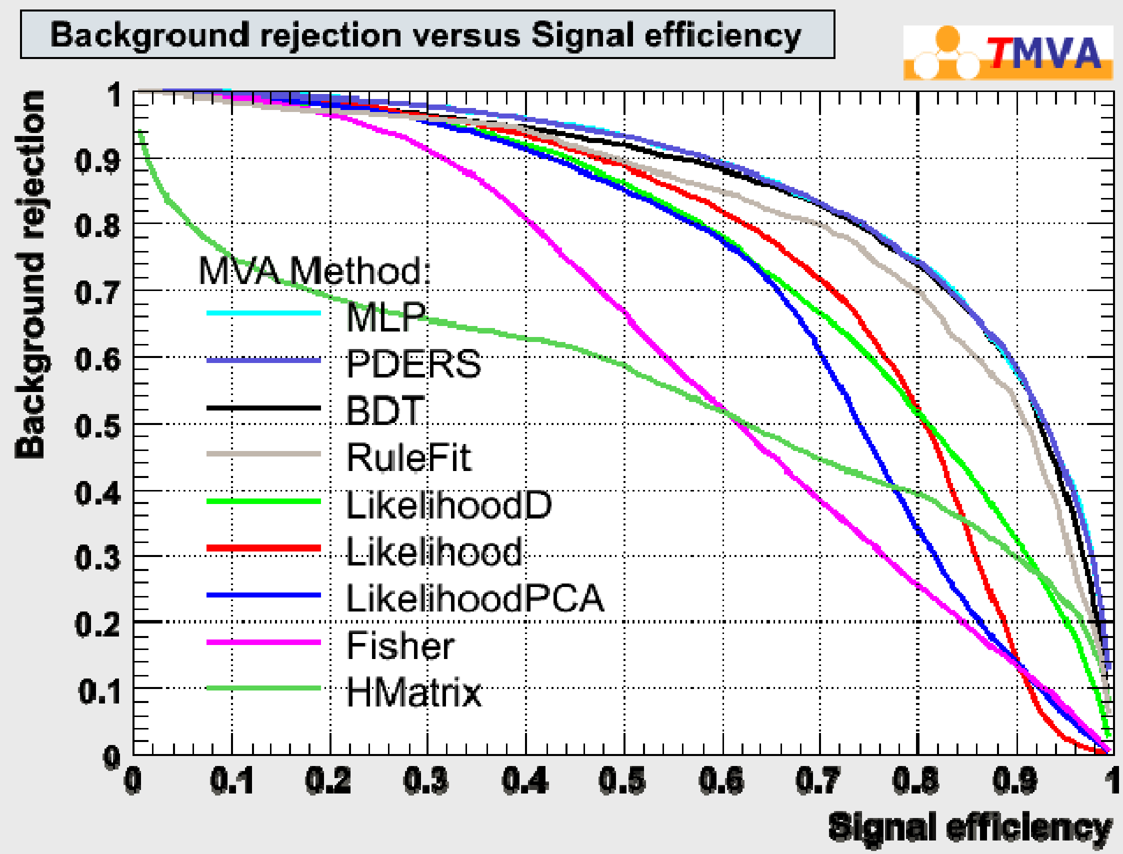
- Background rejection versus signal efficiency curve:



Circular Example

Final Classifier Performance

- Background rejection versus signal efficiency curve:

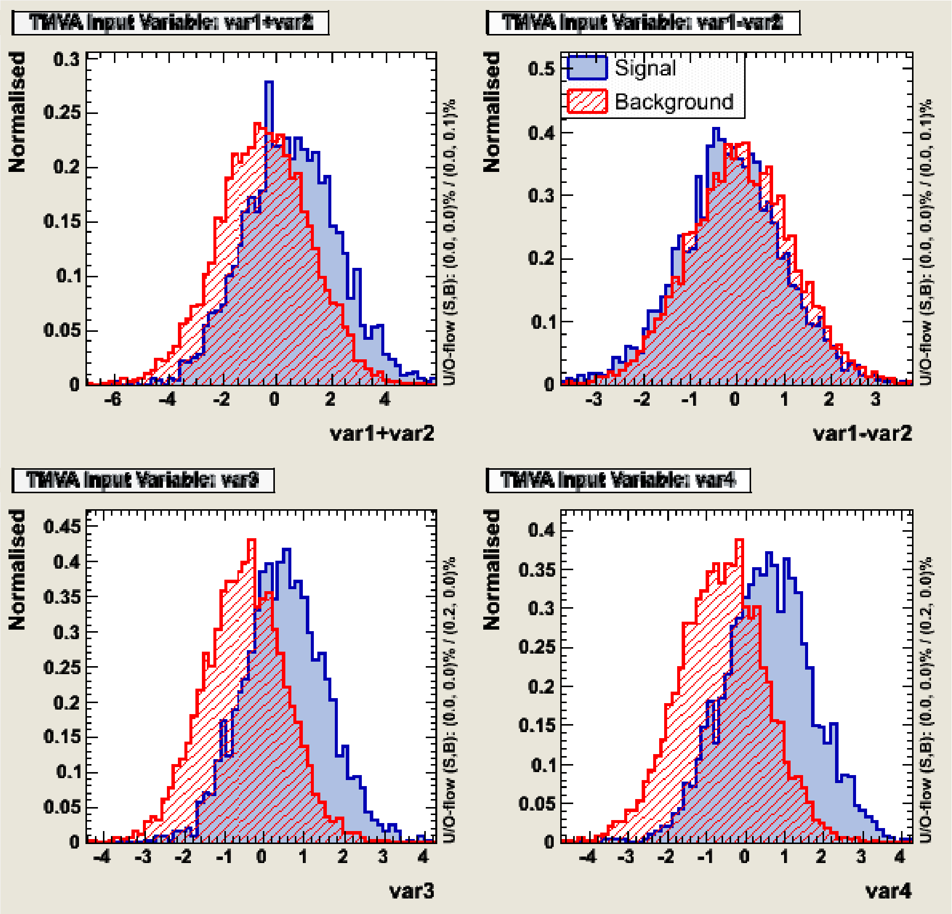


Circular Example

Some words on systematics

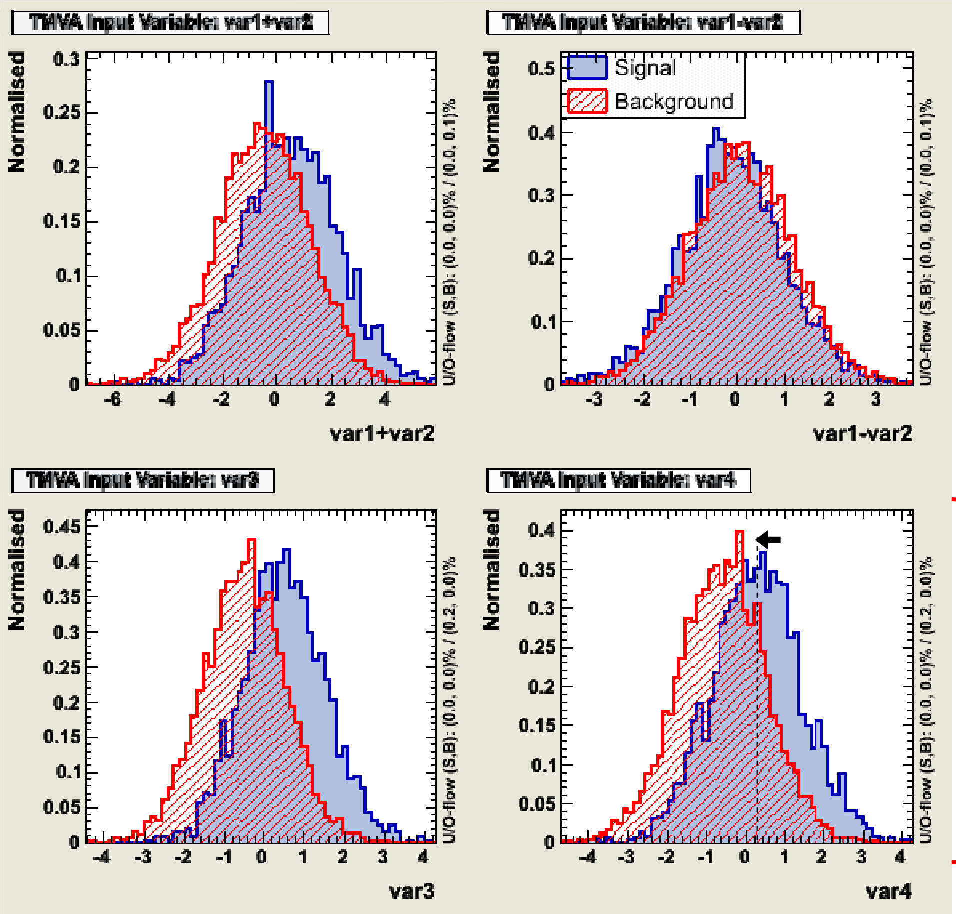
Treatment of Systematic Uncertainties

- 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"

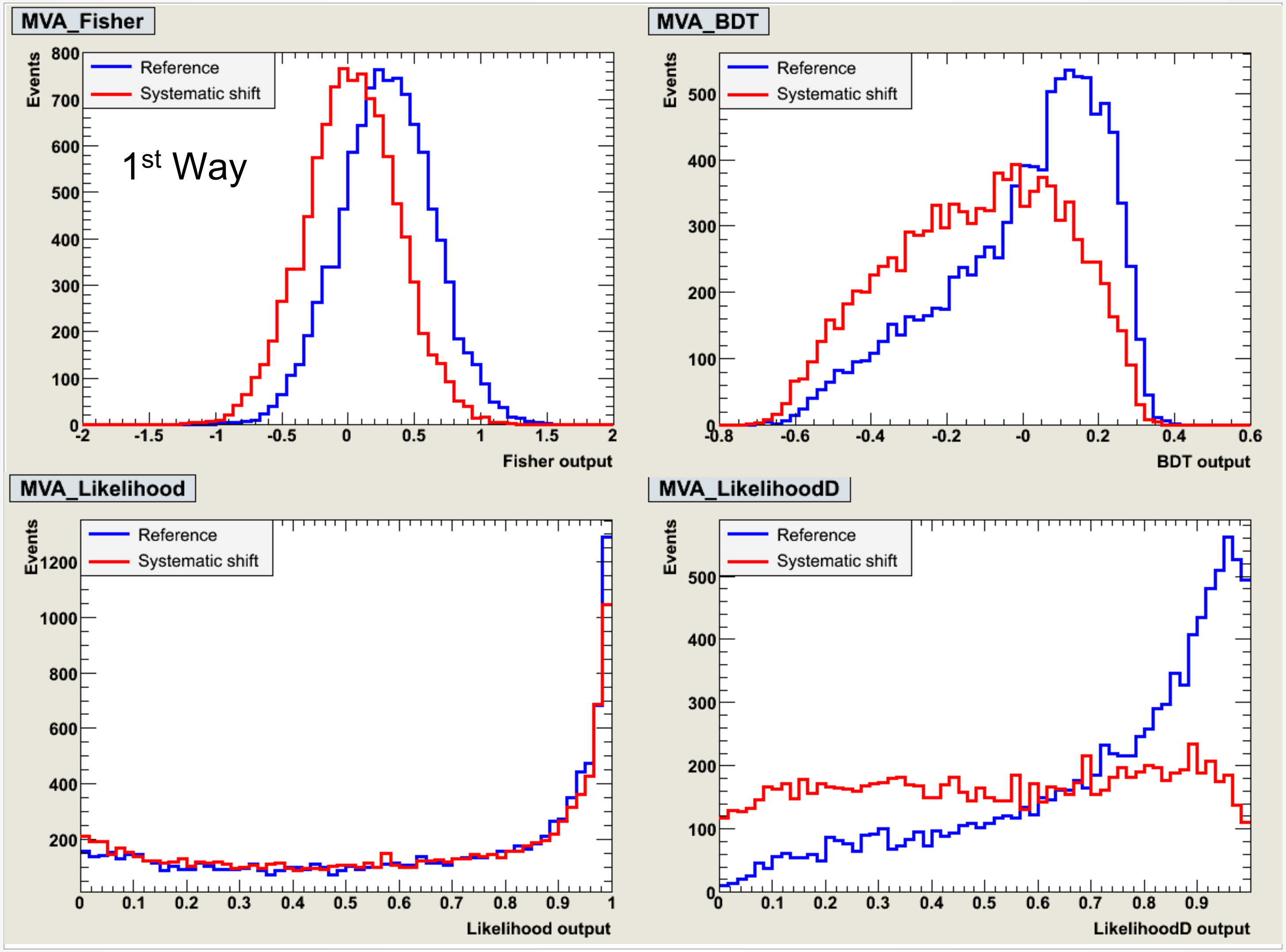
2. Train with shifted (= weakened) "var4", and evaluate systematic error on classifier output

– 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) !

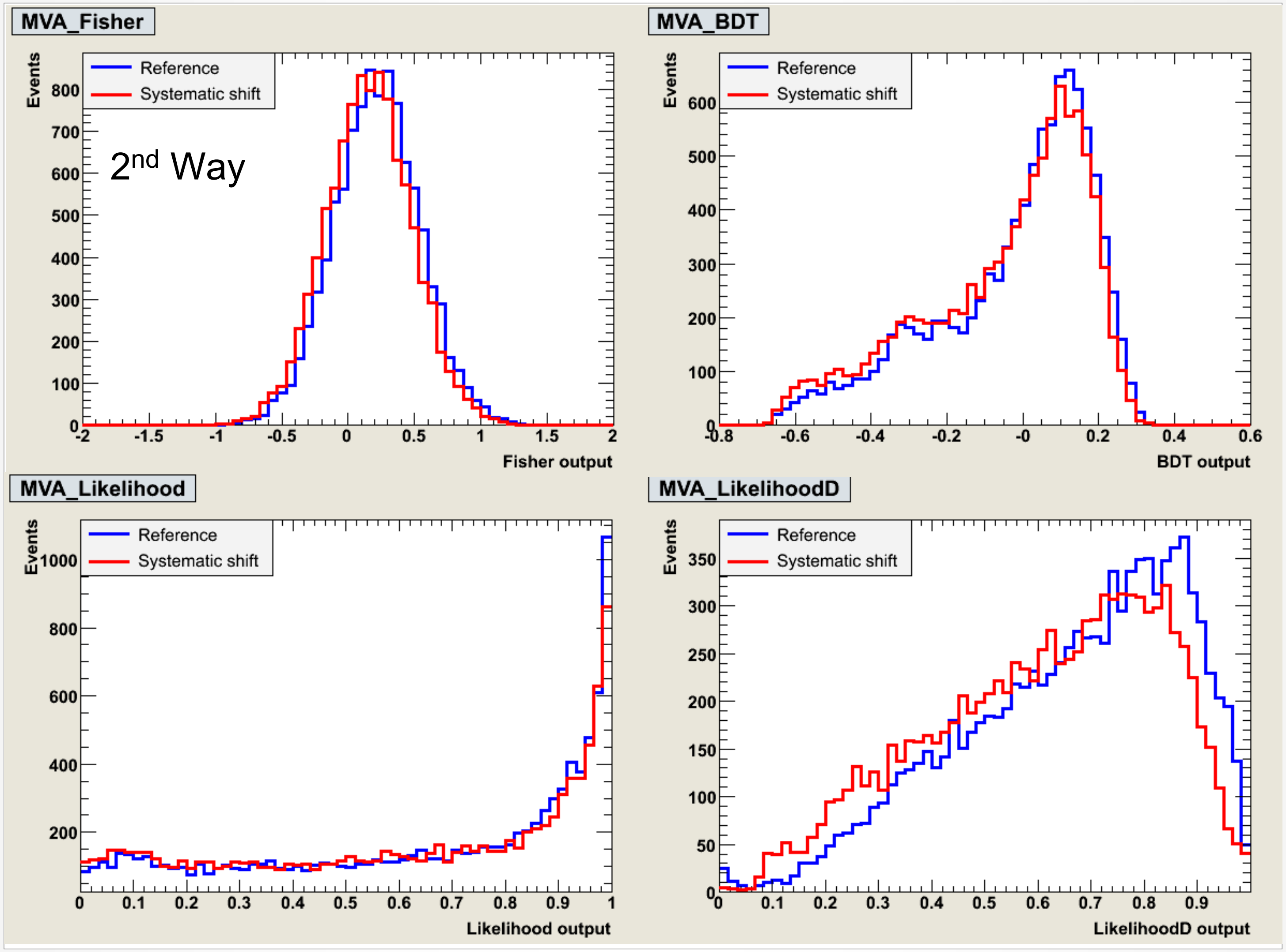
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



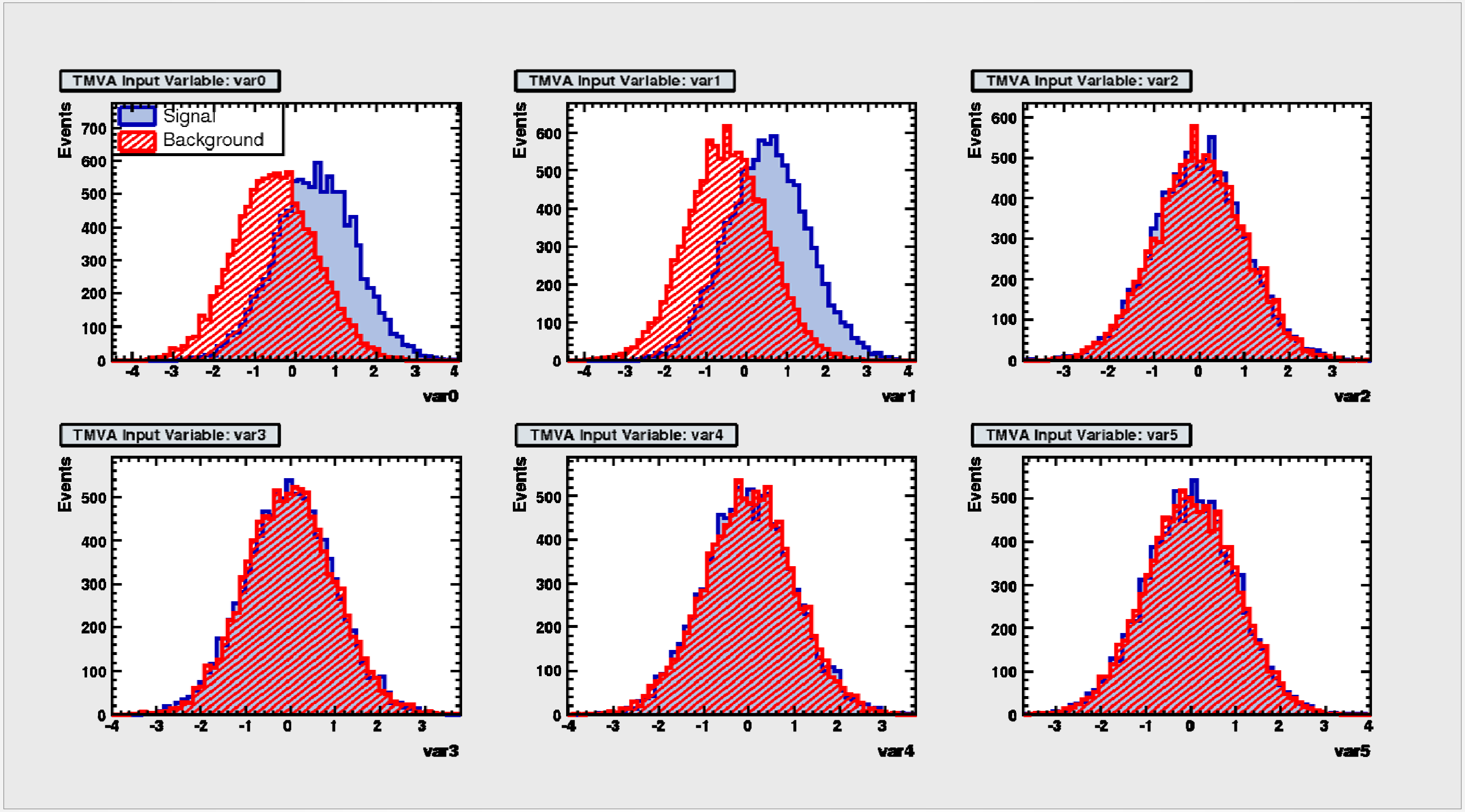
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



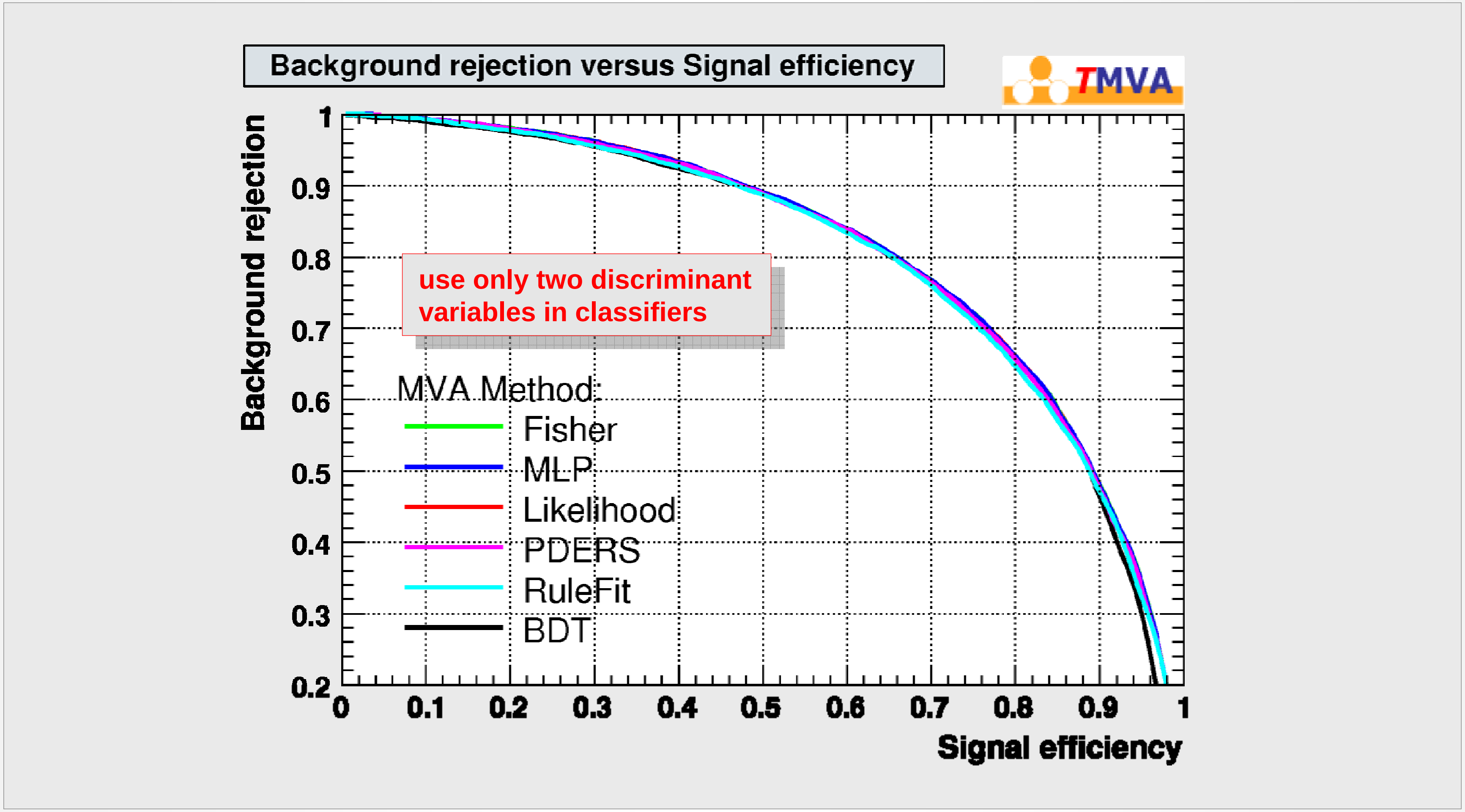
Stability with Respect to Irrelevant Variables

- 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 ?