

Multivariate Data Analysis with **TMVA**

Peter Speckmayer (*) (CERN)

LCD Seminar, CERN, April 14, 2010

(*) On behalf of the present core developer team: A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. v. Toerne, H. Voss
And the contributors: Tancredi Carli (CERN, Switzerland), Asen Christov (Universität Freiburg, Germany), Krzysztof Danielowski (IFJ and AGH/UJ, Krakow, Poland), Dominik Dannheim (CERN, Switzerland), Sophie Henrot-Versille (LAL Orsay, France), Matthew Jachowski (Stanford University, USA), Kamil Kraszewski (IFJ and AGH/UJ, Krakow, Poland), Attila Krasznahorkay Jr. (CERN, Switzerland, and Manchester U., UK), Maciej Kruk (IFJ and AGH/UJ, Krakow, Poland), Yair Mahalalel (Tel Aviv University, Israel), Rustem Ospanov (University of Texas, USA), Xavier Prudent (LAPP Annecy, France), Arnaud Robert (LPNHE Paris, France), Doug Schouten (S. Fraser University, Canada), Fredrik Tegenfeldt (Iowa University, USA, until Aug 2007), Jan Therhaag (Universität Bonn, Germany), Alexander Voigt (CERN, Switzerland), Kai Voss (University of Victoria, Canada), Marcin Wolter (IFJ PAN Krakow, Poland), Andrzej Zemla (IFJ PAN Krakow, Poland).

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

Outline

- Introduction:
 - the reasons why we need “sophisticated” data analysis algorithms
 - the classification/(regression) problem
 - what is Multivariate Data Analysis and Machine Learning
 - a little bit of statistics
- Classifiers in TMVA
 - Cuts
 - Kernel Methods and Likelihood Estimators
 - Linear Fisher Discriminant
 - Neural Networks
 - Support Vector Machines
 - Boosted Decision Trees
 - General boosting
 - Category classifier
- TMVA
 - Using TMVA
 - Toy examples

Literature / Software packages

... a short/biased selection

- **Literature**
 - T.Hastie, R.Tibshirani, J.Friedman, “The Elements of Statistical Learning”, Springer 2001
 - C.M.Bishop, “Pattern Recognition and Machine Learning”, Springer 2006
- **Software packages for Multivariate Data Analysis/Classification**
 - individual classifier software
 - e.g. “JETNET” C.Peterson, T. Rognvaldsson, L.Loennblad
- **attempts to provide “all inclusive” packages**
 - StatPatternRecognition: I.Narsky, arXiv: physics/0507143
 - <http://www.hep.caltech.edu/~narsky/spr.html>
 - TMVA: Höcker, Speckmayer, Stelzer, Therhaag, v.Toerne, Voss, arXiv: physics/0703039
 - <http://tmva.sf.net> or every ROOT distribution (not necessarily the latest TMVA version though)
 - WEKA: <http://www.cs.waikato.ac.nz/ml/weka/>
 - Huge data analysis library available in “R”: <http://www.r-project.org/>
- **Conferences: PHYSTAT, ACAT, CHEP...**

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

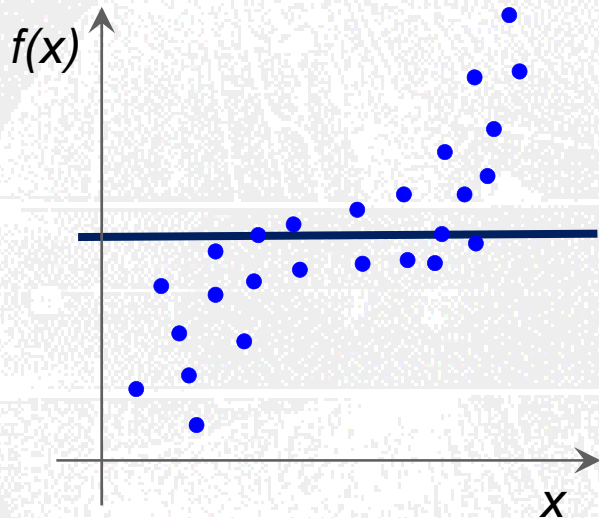
e.g. MiniBooNE: NIMA 543 (2005), or D0 single top: Phys.Rev. D78, 012005 (2008)

Regression

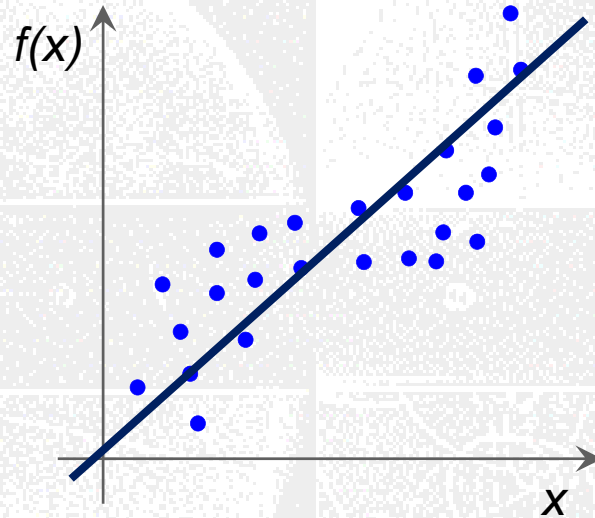
■ How to estimate a “functional behaviour” from a set of measurements?

- Energy deposit in a the calorimeter, distance between overlapping photons, ...
- Entry location of the particle in the calorimeter or on a silicon pad, ...

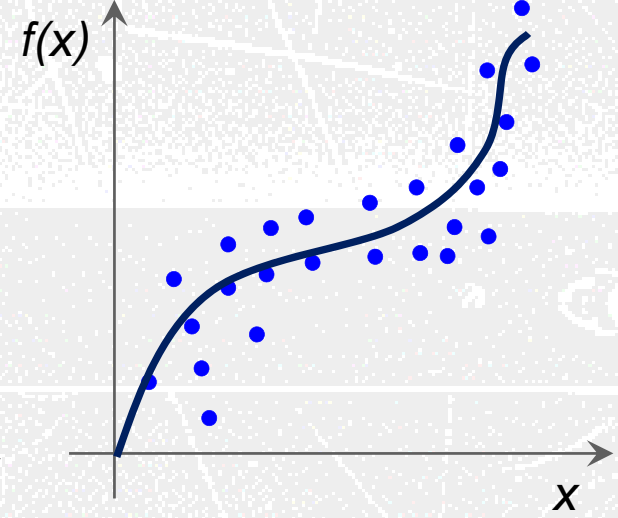
Constant ?



Linear function ?



Nonlinear ?

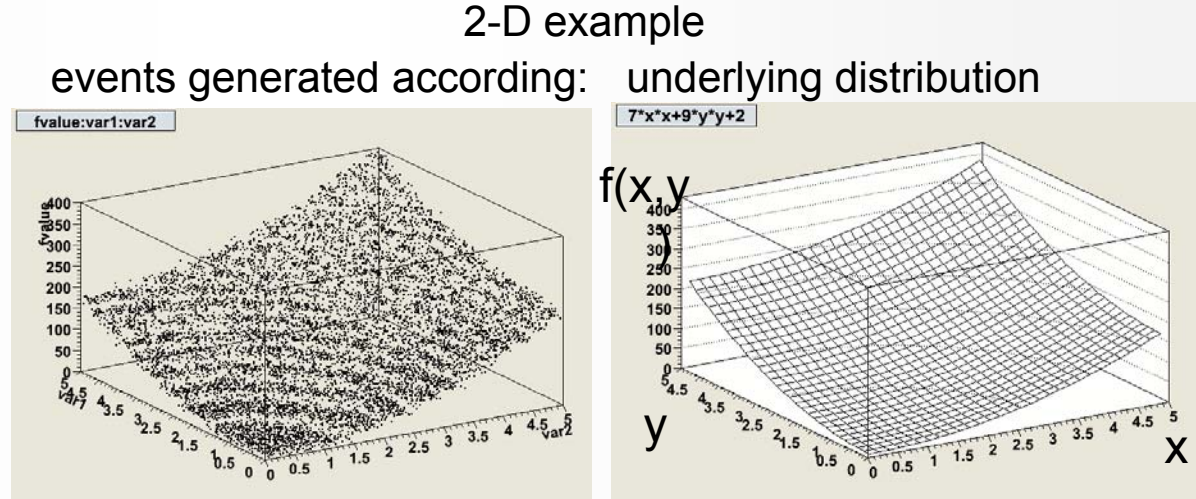
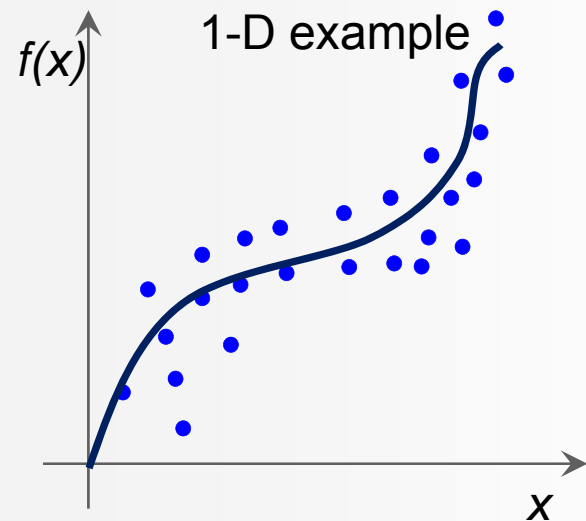


■ Seems trivial? → human eye has good pattern recognition

■ What if we have many input variables?

Regression → model functional behaviour

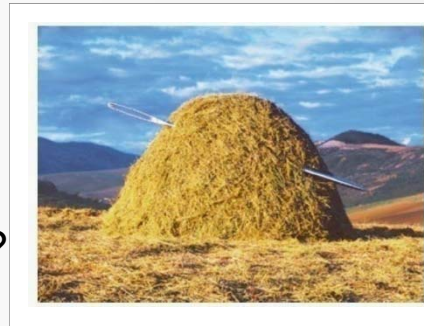
- Assume for example “D”-variables that somehow characterize the shower in your calorimeter.
 - Monte Carlo or testbeam
 - data sample with measured cluster observables
 - + known particle energy
 - = calibration function (energy == surface in D+1 dimensional space)



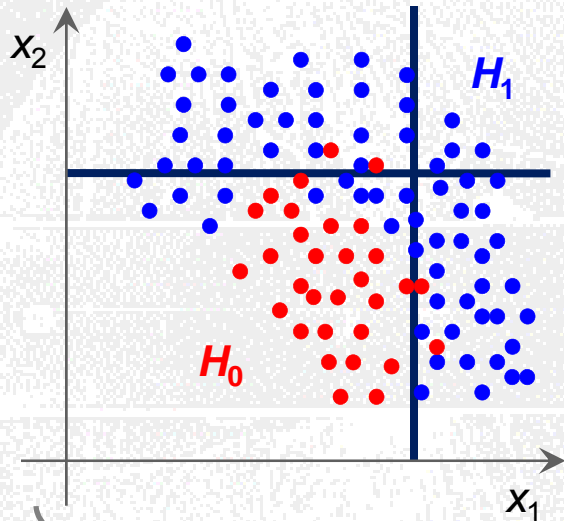
- better known: (linear) regression → fit a known analytic function
 - e.g. the above 2-D example → reasonable function would be: $f(x) = ax^2+by^2+c$
- what if we don't have a reasonable “model” ? → need something more general:
 - e.g. piecewise defined splines, kernel estimators, decision trees to approximate $f(x)$

Event Classification

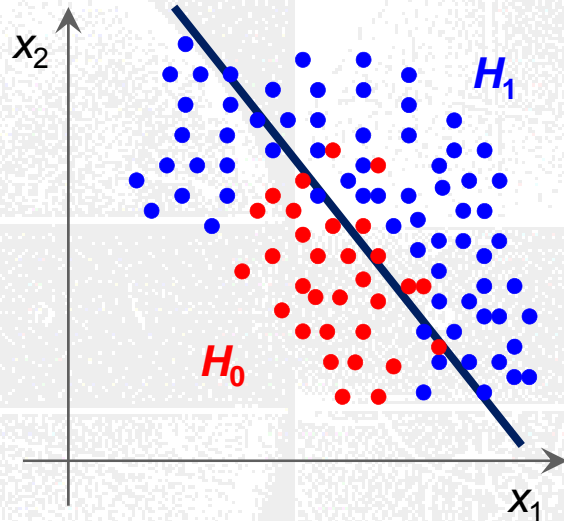
- Suppose data sample with two types of events: H_0 , H_1
 - We have found discriminating input variables x_1, x_2, \dots
 - What decision boundary should we use to select events of type H_1 ?



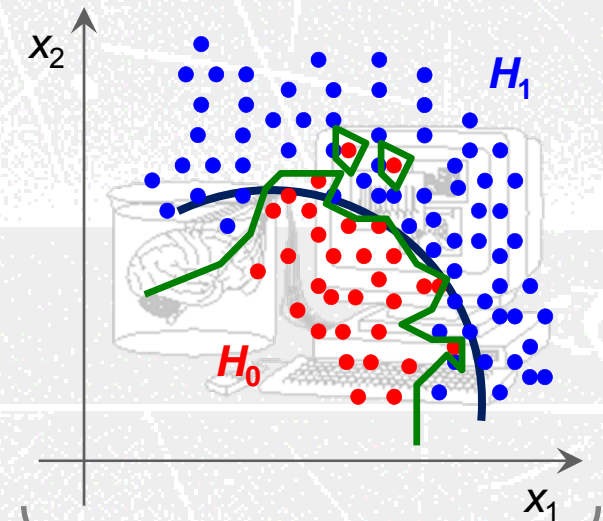
Rectangular cuts?



A linear boundary?



A nonlinear one?



Low variance (stable), high bias methods

High variance, small bias methods

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

Multivariate Classification

R^N

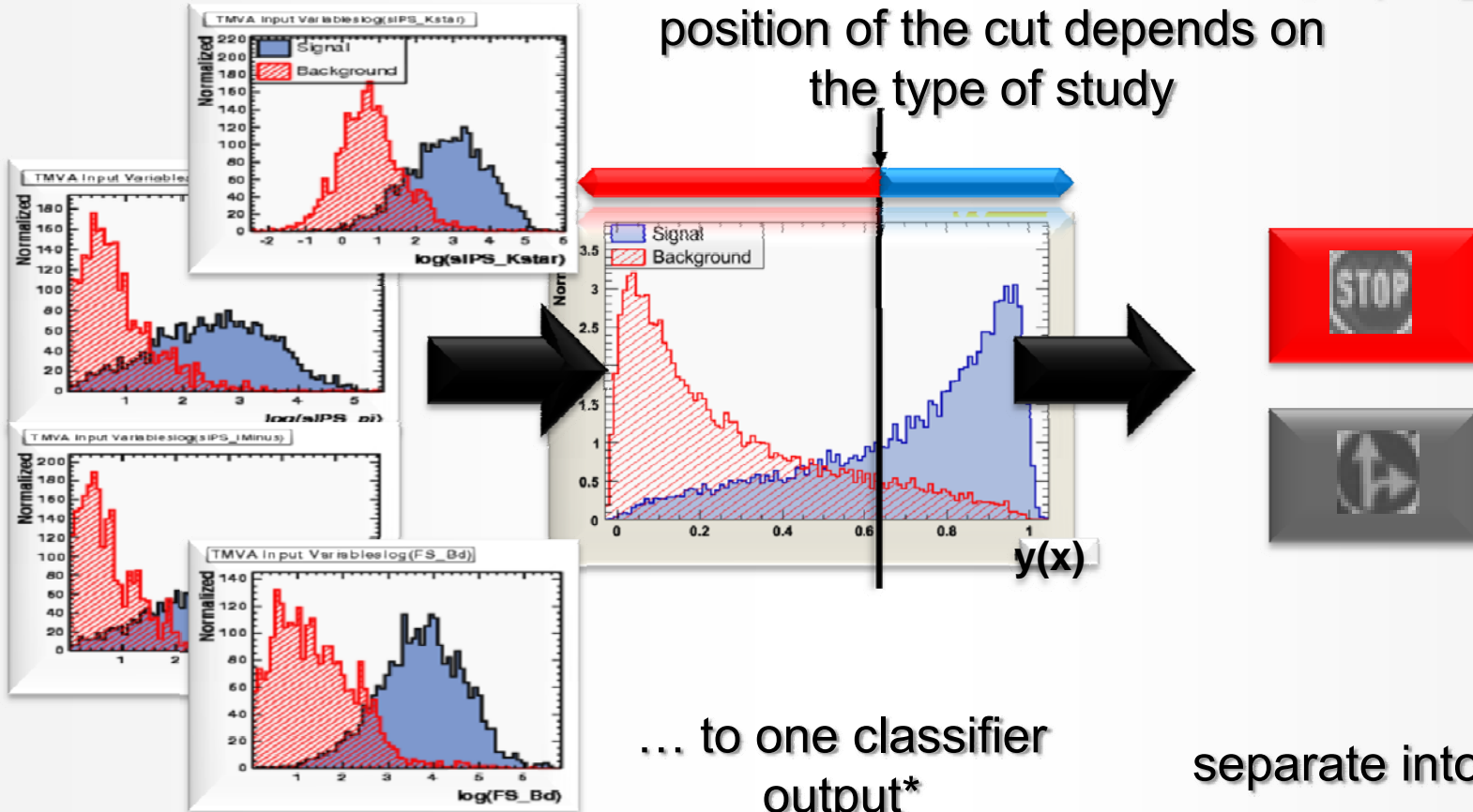


R



$\{C_1, C_2\}$

position of the cut depends on the type of study



... to one classifier output*

separate into classes

multiple input variables

choose a cut value on the classifier y

*Cut classifier is an exception: Direct mapping from $R^N \rightarrow \{\text{Signal, Background}\}$

Multivariate Classification

R^N

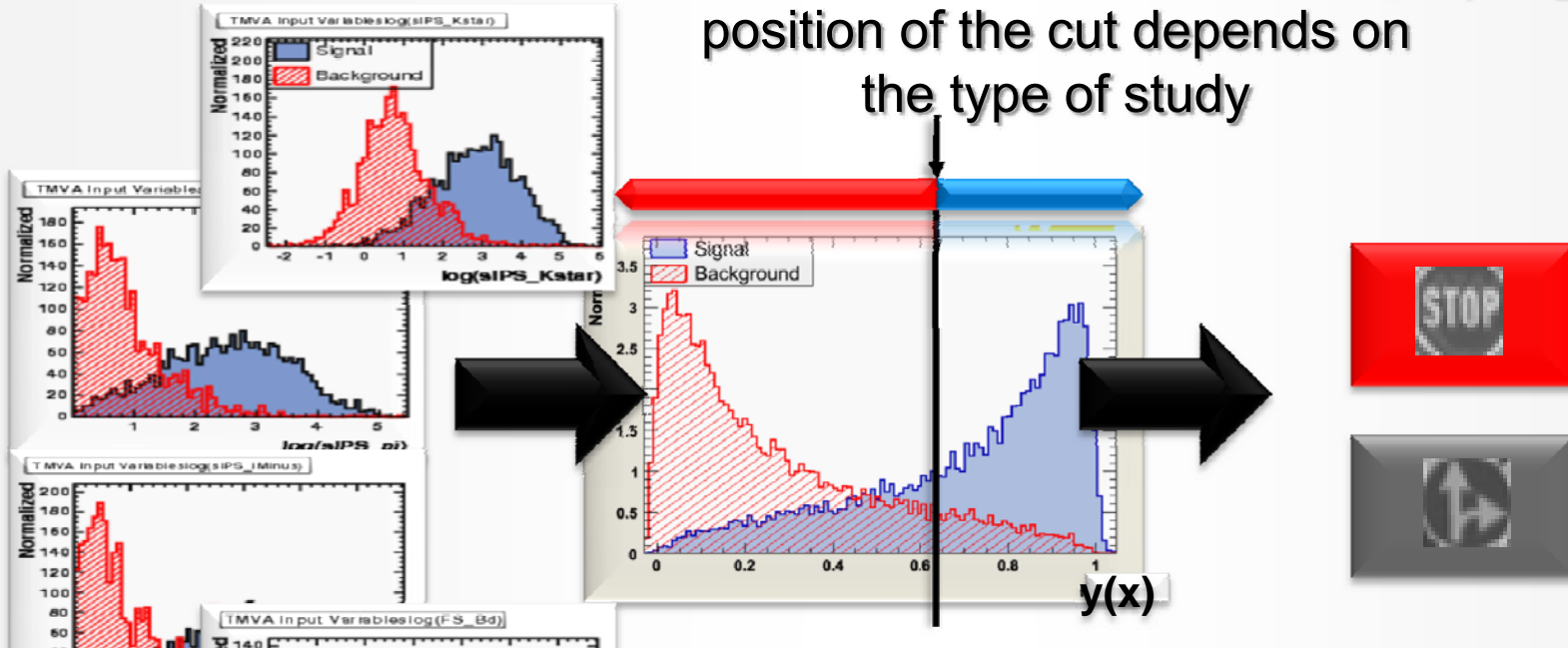


R



$\{C_1, C_2\}$

position of the cut depends on the type of study



- Distributions of $y(x)$: $PDF_S(y)$ and $PDF_B(y)$
- $y(x) = \text{const}$: surface defining the decision boundary.
- Overlap of $PDF_S(y)$ and $PDF_B(y)$ affects separation power, purity

classes

Cut classifier is an exception: Direct mapping from $R^N \rightarrow \{\text{Signal, Background}\}$

Event Classification

$P(\text{Class}=\text{C}|\mathbf{x})$ (or simply $P(\text{C}|\mathbf{x})$) : probability that the event class is of type C, given the measured observables $\mathbf{x} = \{x_1, \dots, x_D\} \rightarrow y(\mathbf{x})$

Probability density distribution according to the measurements \mathbf{x} and the given mapping function

Prior probability to observe an event of “class C”, *i.e.*, the relative abundance of “signal” versus “background”

$$P(\text{Class} = \text{C} | y) = \frac{P(y|\text{C}) \cdot P(\text{C})}{P(y)}$$

↑
Posterior probability

↑
Overall probability density to observe the actual measurement $y(\mathbf{x})$, *i.e.*,
$$P(y) = \sum_{\text{C}}^{\text{Classes}} P(y|\text{C}) \cdot P(\text{C})$$

Bayes Optimal Classification

$$P(\text{Class} = C | \mathbf{y}) = \frac{P(\mathbf{y} | C)P(C)}{P(\mathbf{y})} \quad \mathbf{x} = \{x_1, \dots, x_D\}: \text{measured observables}$$

$$y = y(\mathbf{x})$$

+

Minimum error in misclassification if C chosen such that it has maximum $P(C|\mathbf{y})$

→ to select S(signal) over B(background), place decision on:

[Or any
monotonic
function of
 $P(S|\mathbf{y}) / P(B|\mathbf{y})$]

$$\frac{P(S | \mathbf{y})}{P(B | \mathbf{y})} = \frac{P(\mathbf{y} | S)}{P(\mathbf{y} | B)} \cdot \frac{P(S)}{P(B)} > c$$

← “c” determines
efficiency and purity

Posterior
odds ratio

Likelihood ratio
as discriminating
function $y(\mathbf{x})$

Prior odds ratio of choosing a signal event
(relative probability of signal vs. bkg)

Any Decision Involves a Risk

Decide to treat an event as “Signal” or “Background”

Type-1 error: (false positive)

classify event as Class C even though it is not

(accept a hypothesis although it is not true)

(reject the null-hypothesis although it would have been the correct one)

→ **loss of purity** (in the selection of signal events)

Type-2 error: (false negative)

fail to identify an event from Class C as such

(reject a hypothesis although it would have been true)

(fail to reject the null-hypothesis/accept null hypothesis although it is false)

→ **loss of efficiency** (in selecting signal events)

Trying to select signal events:
(i.e. try to disprove the null-hypothesis stating it were “only” a background event)

accept as: truly is:	Signal	Back-ground
Signal	☺	Type-2 error
Back-ground	Type-1 error	☺

“A”: region of the outcome of the test where you accept the event as **signal**:

Significance α : Type-1 error rate:

(=p-value): α = **background selection “efficiency”**

$$\alpha = \int_A P(x | B) dx \quad \text{should be small}$$

miss rate β : Type-2 error rate:

Power: $1 - \beta$ = **signal selection efficiency**

$$\beta = \int_{!A} P(x | S) dx \quad \text{should be small}$$

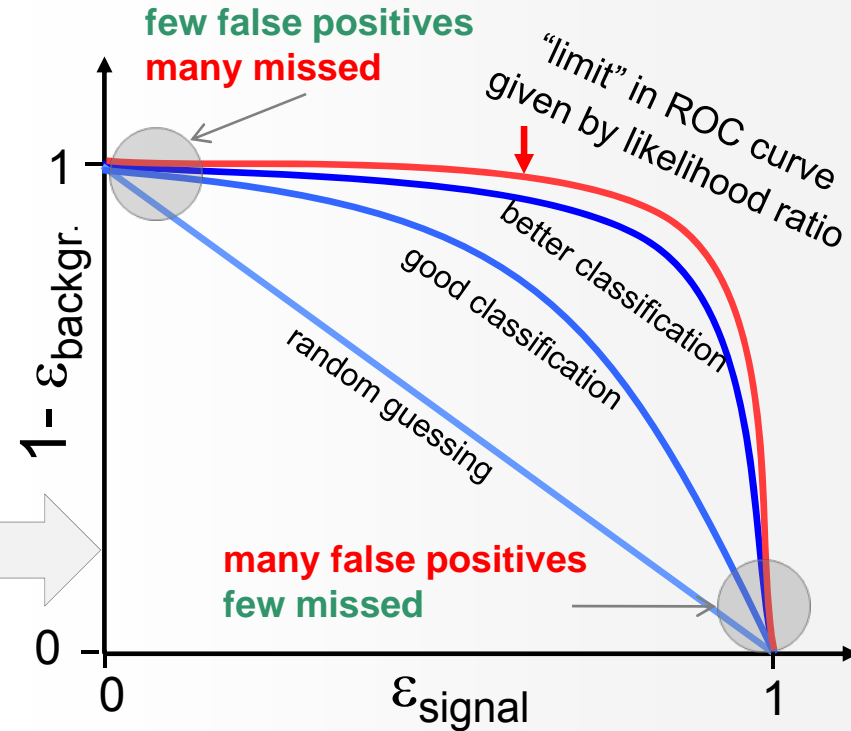
Neyman-Pearson Lemma

Likelihood Ratio :
$$y(x) = \frac{P(x | S)}{P(x | B)}$$

Neyman-Pearson:

The Likelihood ratio used as “selection criterion” $y(x)$ gives for each selection efficiency the best possible background rejection. (1933)

i.e. it maximises the area under the “Receiver Operation Characteristics” (ROC) curve



Varying $y(x) > \text{“cut”}$ moves the working point (efficiency and purity) along the ROC curve

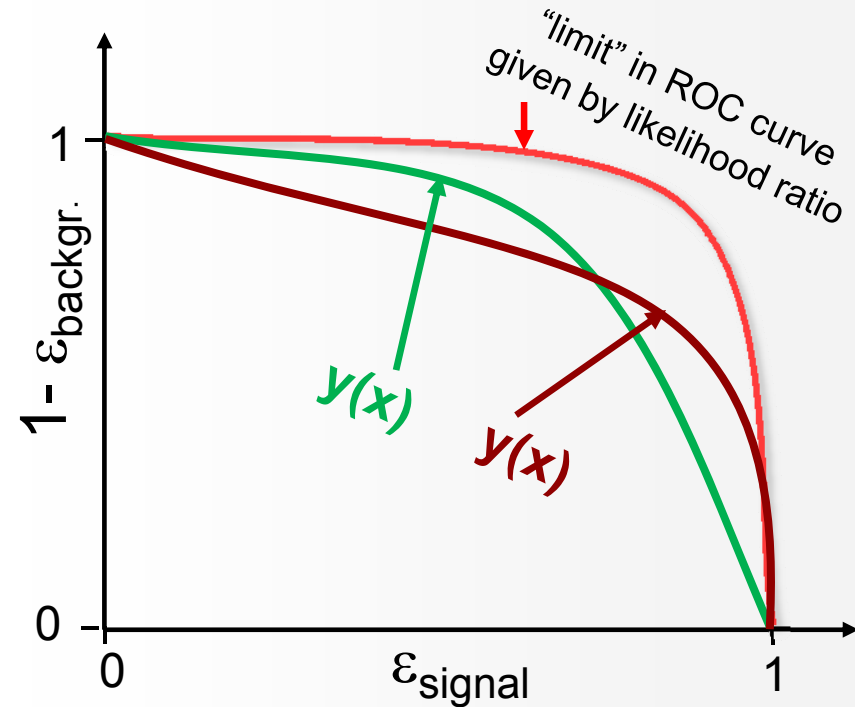
How to choose “cut”? \rightarrow need to know prior probabilities (S , B abundances)

- Measurement of signal cross section: maximum of $S/\sqrt{(S+B)}$ or equiv. $\sqrt{(\epsilon \cdot p)}$
- Discovery of a signal : maximum of $S/\sqrt{(B)}$
- Precision measurement: high purity (p)
- Trigger selection: high efficiency (ϵ)

Neyman-Pearson Lemma

if discriminating function $y(x)$ = “true likelihood ratio”
→ optimal working point for specific analysis lies somewhere on the ROC curve

$y(x)$ ≠ “true likelihood ratio” different, point on
→ $y(x)$ might be better for a specific working point than $y(x)$ and vice versa



Note:

for the determination of your working point (e.g. S/\sqrt{B})
you need the prior S and B probabilities!
→ number of events/luminosity

Realistic Event Classification

Unfortunately, the true probability densities functions are typically unknown:

→ Neyman-Pearson's lemma doesn't really help us...

Use MC simulation, or more generally: set of known (already classified) "events"

Use these "training" events to:

- Try to estimate the functional form of $P(x|C)$ from which the likelihood ratio can be obtained
→ e.g. D-dimensional histogram, Kernel density estimators, MC-based matrix-element methods, ...
- Find a "discrimination function" $y(x)$ and corresponding decision boundary (i.e. hyperplane* in the "feature space": $y(x) = \text{const}$) that optimally separates signal from background
→ e.g. Linear Discriminator, Neural Networks, ...

→ supervised (machine) learning

* hyperplane in the strict sense goes through the origin. Here is meant an "affine set" to be precise.

Realistic Event Classification

Unfortunately, the true probability densities functions are typically unknown:

→ Neyman-Pearson's lemma doesn't really help us...

Use MC simulation, or more generally: set of known (already classified) "events"

Use Of course, there is **no magic** in here. We still need to:

- Choose the discriminating variables
- Choose the class of models (linear, non-linear, flexible or less flexible)
- Tune the "learning parameters" → bias vs. variance trade off
- Check generalisation properties
- Consider trade off between statistical and systematic uncertainties

→ e.g. Linear Discriminator, Neural Networks, ...

→ supervised (machine) learning

* hyperplane in the strict sense goes through the origin. Here is meant an "affine set" to be precise.

What is **TMVA**

- 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 `TMultLayerPerceptron` does):
 - Have one common platform / interface for high-end multivariate 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

Multivariate Analysis Methods

➤ Examples for classifiers and regression methods

- Rectangular cut optimisation
- Projective and multidimensional likelihood estimator
- k-Nearest Neighbor algorithm
- Fisher, Linear and H-Matrix discriminants
- Function discriminants
- Artificial neural networks
- Boosted decision trees
- RuleFit
- Support Vector Machine

➤ Examples for preprocessing methods:

- Decorrelation, Principal Value Decomposition, Gaussianisation

➤ Examples for combination methods:

- Boosting, Categorisation

Data Preprocessing

Data Preprocessing: Decorrelation

- Commonly realised for all methods in **TMVA**
- Removal of linear correlations by rotating input variables
 - Cholesky decomposition: determine *square-root* C' of covariance matrix C , i.e., $C = C' C'$
 - Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$

■ Principal component analysis

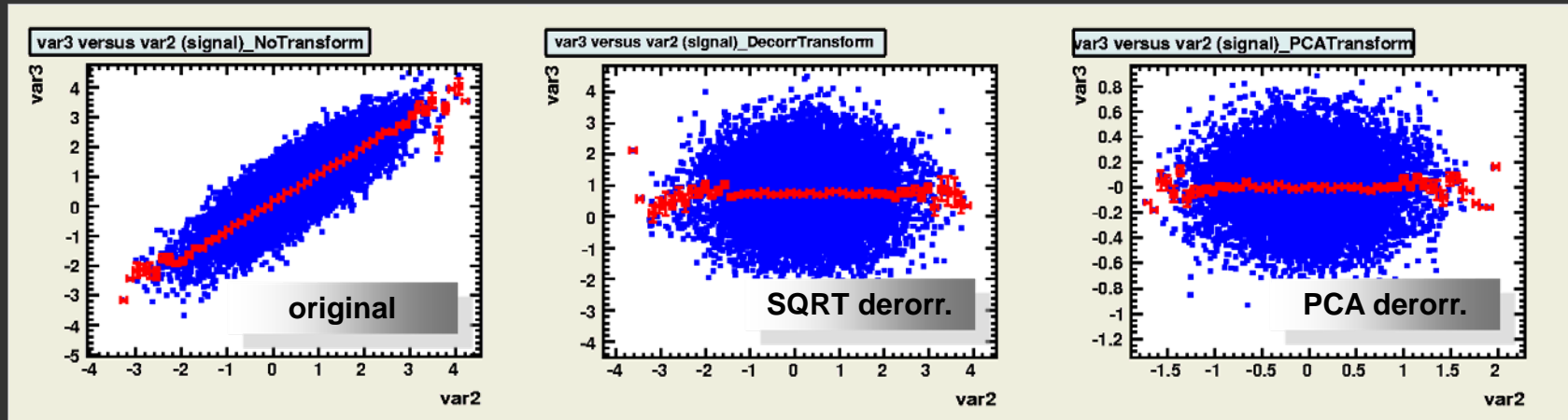
- Variable hierarchy: linear transformation projecting on axis to achieve largest variance

$$x_k^{\text{PC}}(i_{\text{event}}) = \sum_{v \in \{\text{variables}\}} [x_v(i_{\text{event}}) - \bar{x}_v] \cdot v_v^{(k)}, \quad \forall k \in \{\text{variables}\}$$

Diagram illustrating the Principal Component Analysis (PCA) equation. The equation shows the PC of variable k for an event i_{event} as a sum over all variables v of the deviation of the variable value from its sample mean, multiplied by the corresponding eigenvector component. Labels with arrows point to the terms: "PC of variable k " points to x_k^{PC} , "Sample means" points to \bar{x}_v , and "Eigenvector" points to $v_v^{(k)}$. To the right, a scatter plot shows data points in a 2D space with axes x_1 and x_2 . The data points are clustered along a line, and two principal components, PC1 and PC2, are shown as orthogonal axes. The angle between PC1 and the x_1 axis is labeled α_1 , and the angle between PC2 and the x_2 axis is labeled α_2 . A third angle α_3 is also indicated.

- Matrix of eigenvectors V obeys relation: $C \cdot V = D \cdot V$ thus PCA eliminates correlations
- Diagram illustrating the matrix relation $C \cdot V = D \cdot V$. The matrix C is labeled "correlation matrix" and the matrix D is labeled "diagonalised square root of C". Arrows point from these labels to the matrices in the equation.

Data Preprocessing: Decorrelation



Note that decorrelation is only complete, if

- Correlations are linear
- Input variables are Gaussian distributed
- ◆ Not very accurate conjecture in general

“Gaussian-isation”

■ Improve decorrelation by pre-“Gaussianisation” of variables

- ➔ First: “Rarity” transformation to achieve uniform distribution:

$$x_k^{\text{flat}}(i_{\text{event}}) = \int_{-\infty}^{x_k(i_{\text{event}})} p_k(x'_k) dx'_k, \quad \forall k \in \{\text{variables}\}$$

The diagram shows the equation $x_k^{\text{flat}}(i_{\text{event}}) = \int_{-\infty}^{x_k(i_{\text{event}})} p_k(x'_k) dx'_k, \quad \forall k \in \{\text{variables}\}$. Three arrows point from labels in boxes below to parts of the equation: one from 'Rarity transform of variable k ' to $x_k^{\text{flat}}(i_{\text{event}})$, one from 'Measured value' to $x_k(i_{\text{event}})$, and one from 'PDF of variable k ' to $p_k(x'_k)$.

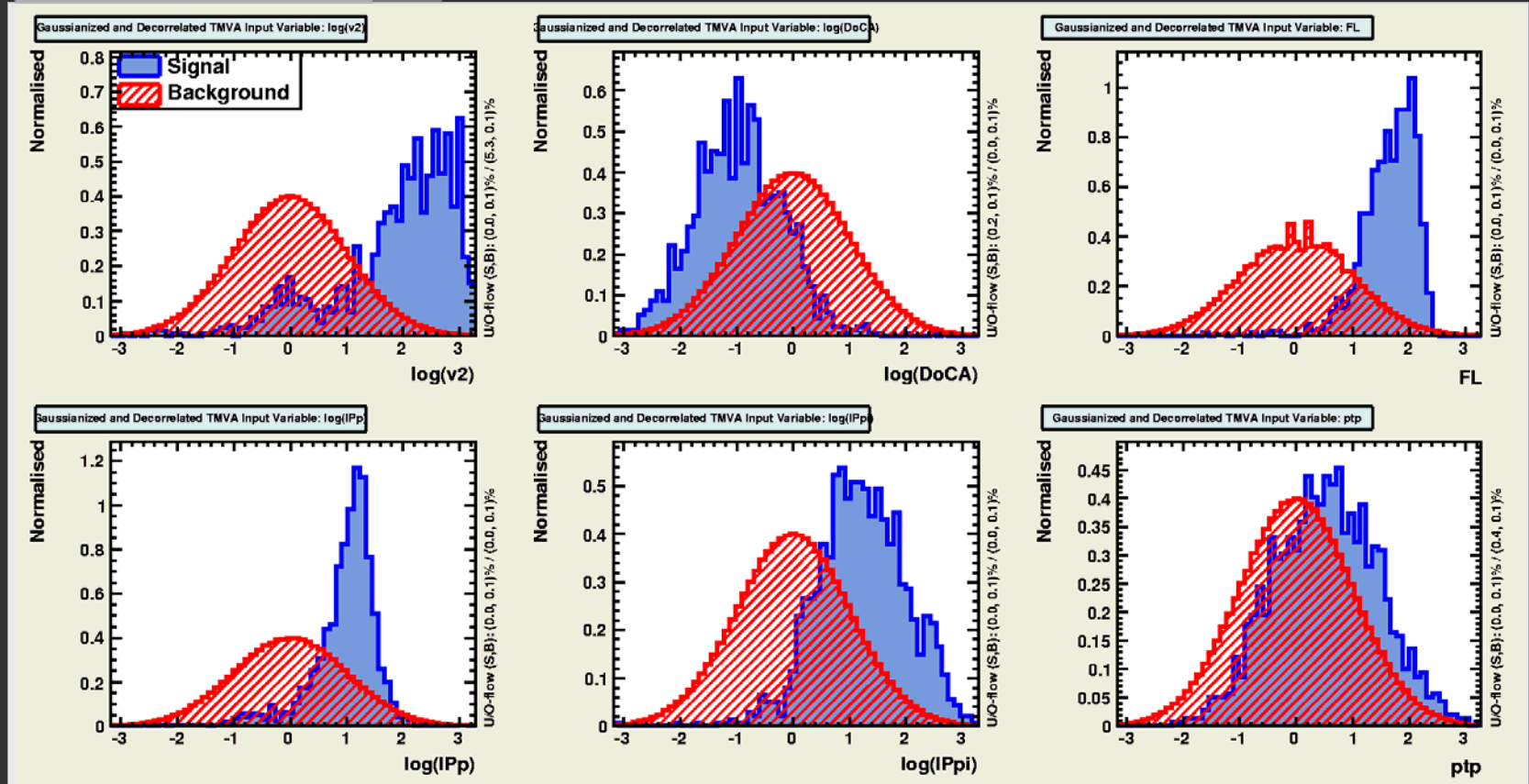
The integral can be solved in an unbinned way by event counting, or by creating non-parametric PDFs (see later for likelihood section)

- ➔ Second: make Gaussian via inverse error function: $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$

$$x_k^{\text{Gauss}}(i_{\text{event}}) = \sqrt{2} \cdot \text{erf}^{-1}(2x_k^{\text{flat}}(i_{\text{event}}) - 1), \quad \forall k \in \{\text{variables}\}$$

“Gaussian-isation”

Background - Gaussianised




We cannot simultaneously “gaussianise” both signal and background !

How to apply the Preprocessing Transformation ?

- Any type of preprocessing will be **different** for signal and background
- But: for a given test event, we do not know the species !
 - **Not so good solution:** choose one or the other, or a S/B mixture.
As a result, none of the transformations will be perfect. → for most of the methods
 - **Good solution:** for some methods it is possible to test both S and B hypotheses with *their* transformations, and to compare them. Example, projective likelihood ratio:

$$y_L(i_{\text{event}}) = \frac{\prod_{k \in \{\text{variables}\}} p_k^S(x_k(i_{\text{event}}))}{\prod_{k \in \{\text{variables}\}} p_k^S(x_k(i_{\text{event}})) + \prod_{k \in \{\text{variables}\}} p_k^B(x_k(i_{\text{event}}))}$$



$$y_L^{\text{trans}}(i_{\text{event}}) = \frac{\prod_{k \in \{\text{variables}\}} p_k^S(\hat{T}^S x_k(i_{\text{event}}))}{\prod_{k \in \{\text{variables}\}} p_k^S(\hat{T}^S x_k(i_{\text{event}})) + \prod_{k \in \{\text{variables}\}} p_k^B(\hat{T}^B x_k(i_{\text{event}}))}$$

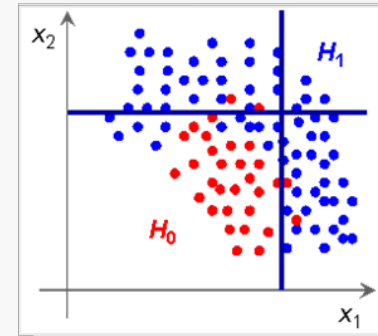
signal transformation
 background transformation

The Classifiers

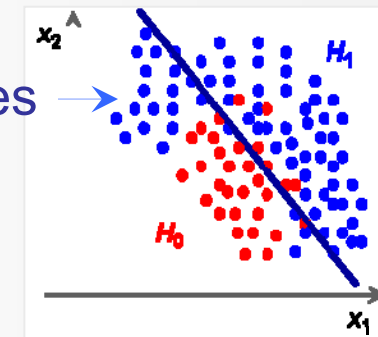
Rectangular Cut Optimisation

- Simplest method: cut in rectangular variable volume

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



- Cuts usually benefit from prior decorrelation of cut variables



- 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

Projective Likelihood Estimator (PDE Approach)

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

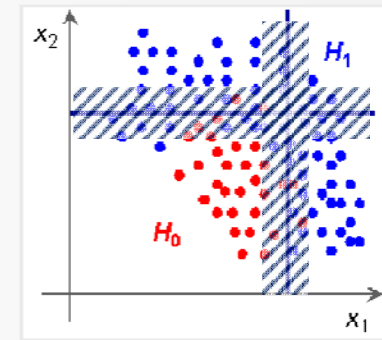
Likelihood ratio for event i_{event}

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

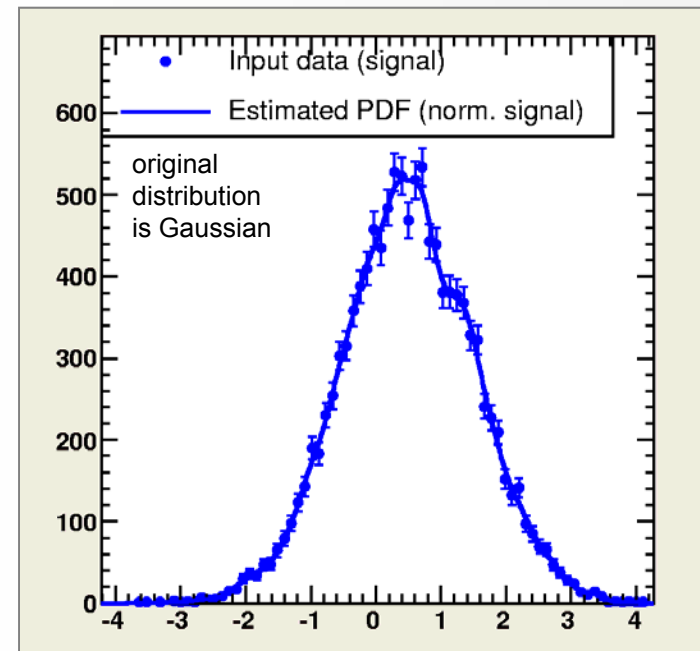
Easy to automate, can create
artefacts/suppress information

event counting

Automatic, unbiased,
but suboptimal

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

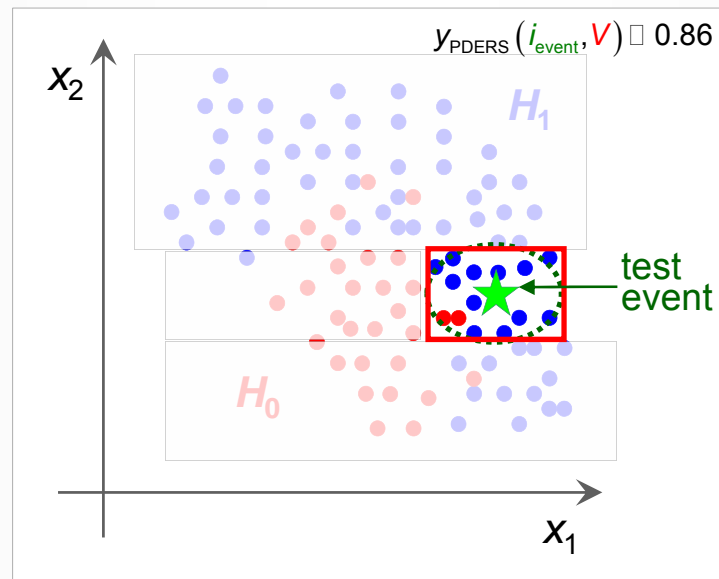
- Binned shape interpolation using spline functions and adaptive smoothing
- Unbinned adaptive kernel density estimation (KDE) with Gaussian smearing
- ➔ *TMVA* performs automatic validation of goodness-of-fit



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 \rightarrow 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

- 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)

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

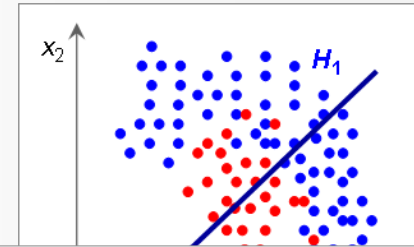


- Improve $y_{\text{PDE}}^{\text{RS}}$ estimate within V by using various N_{var} -D kernel estimators
- Enhance speed of event counting in volume by binary tree search

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, while confining events of



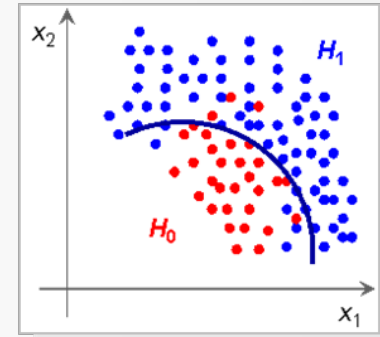
Function discriminant analysis (FDA)

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

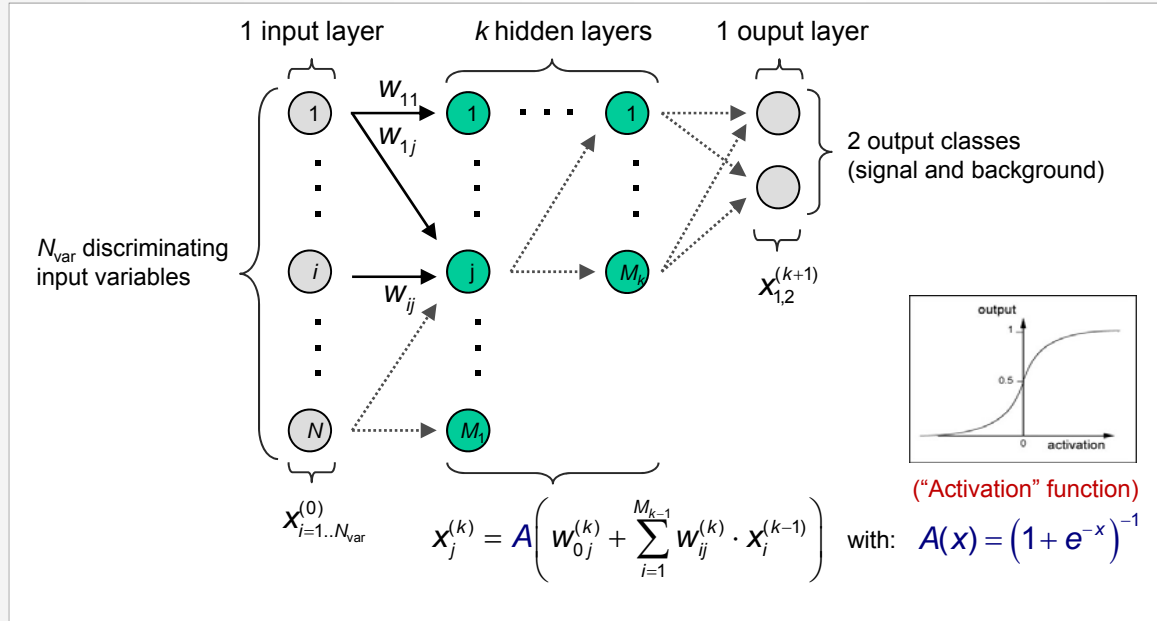
- Parameter fitting: Genetics Alg., MINUIT, MC and combinations
 - Easy reproduction of Fisher result, but can add nonlinearities
 - Very transparent discriminator
-
- Compute Fisher coefficients from signal and background covariance matrices
 - Fisher requires distinct sample means between signal and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

Nonlinear Analysis: Artificial Neural Networks

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



Feed-forward Multilayer Perceptron



Weight adjustment using analytical back-propagation

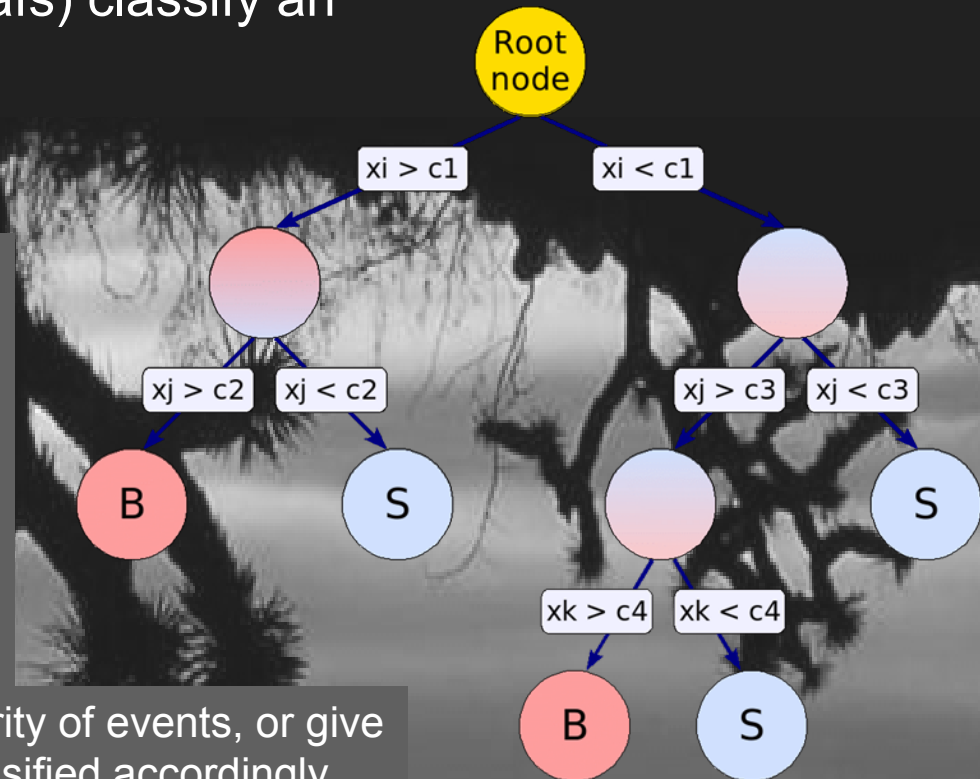
- Three different implementations in TMVA (all are Multilayer Perceptrons)
 - **TMlpANN:** Interface to ROOT’s MLP implementation
 - **MLP:** TMVA’s own MLP implementation for increased speed and flexibility
 - **CFMlpANN:** ALEPH’s Higgs search ANN, translated from FORTRAN

Decision Trees

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

- 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

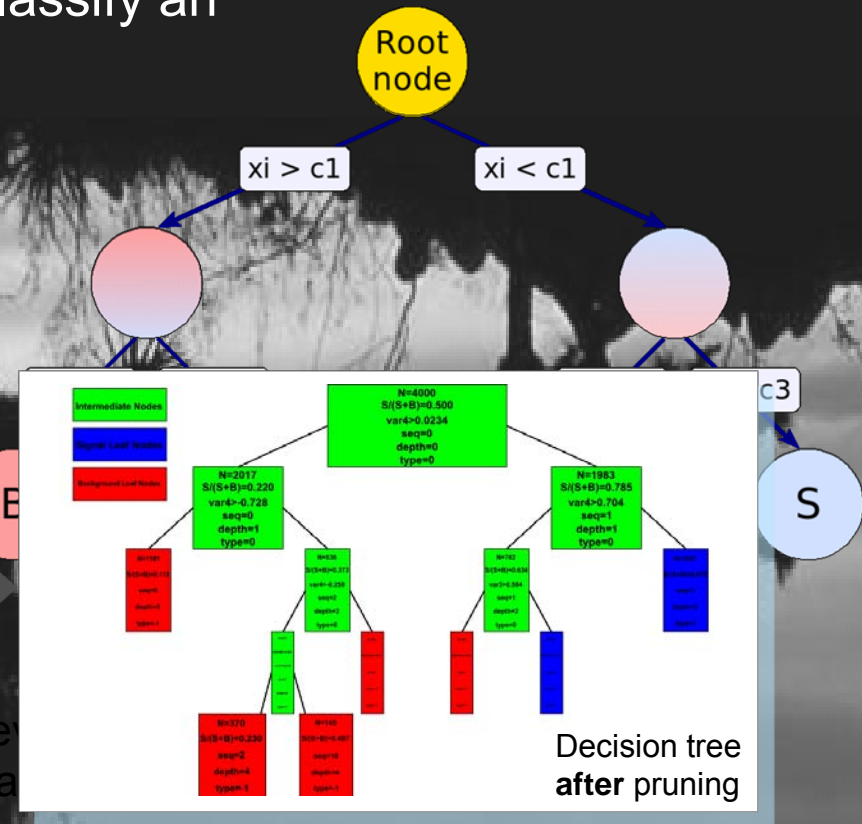


- Why not multiple branches (splits) per node ?

- Fragments data too quickly; also: multiple splits per node = series of binary node splits

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

Boosted Decision Trees (BDT)

- Data mining with decision trees is popular in science (so far mostly outside of HEP)

- ◆ Advantages:

- Easy to interpret
- 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, re-sampling 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

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

- The problem to solve is

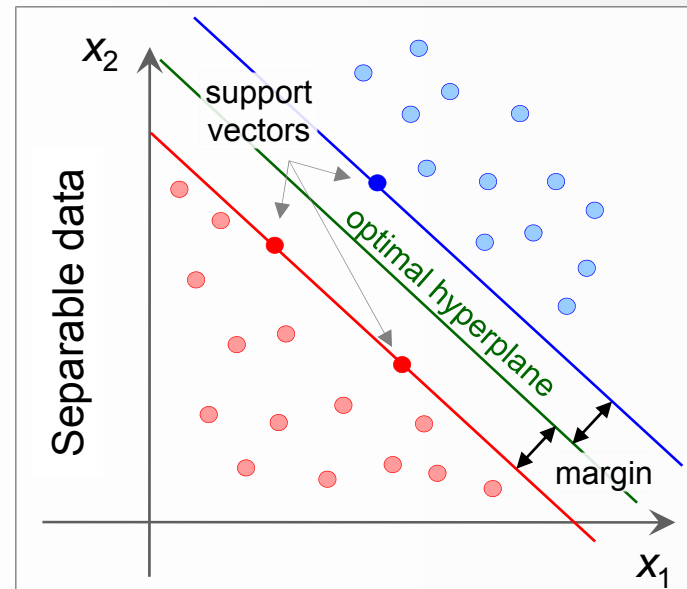
- Create rule ensemble: use forest of decision trees
- Fit coefficients a_m, b_k : gradient direct regularization minimising *Risk* (Friedman et al.)

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

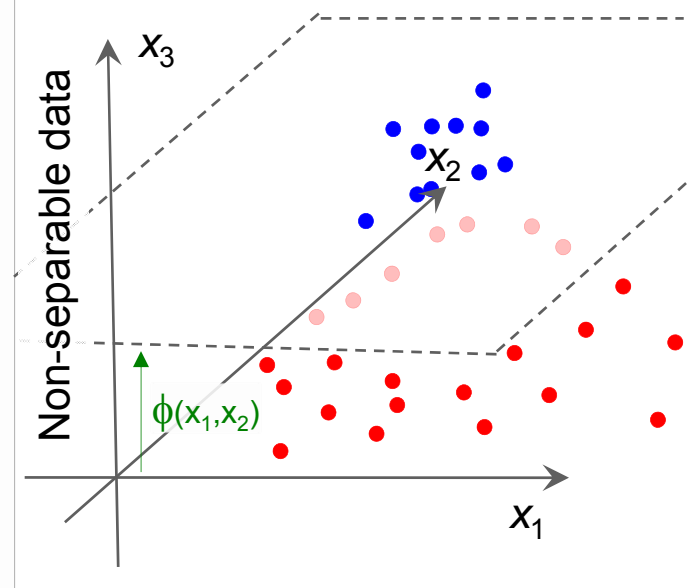
One of the elementary cellular automaton rules (Wolfram 1983, 2002). It specifies the next color in a cell, depending on its color and its immediate neighbors. Its rule outcomes are encoded in the binary representation 30=00011110₂.

Support Vector Machine (SVM)

- 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

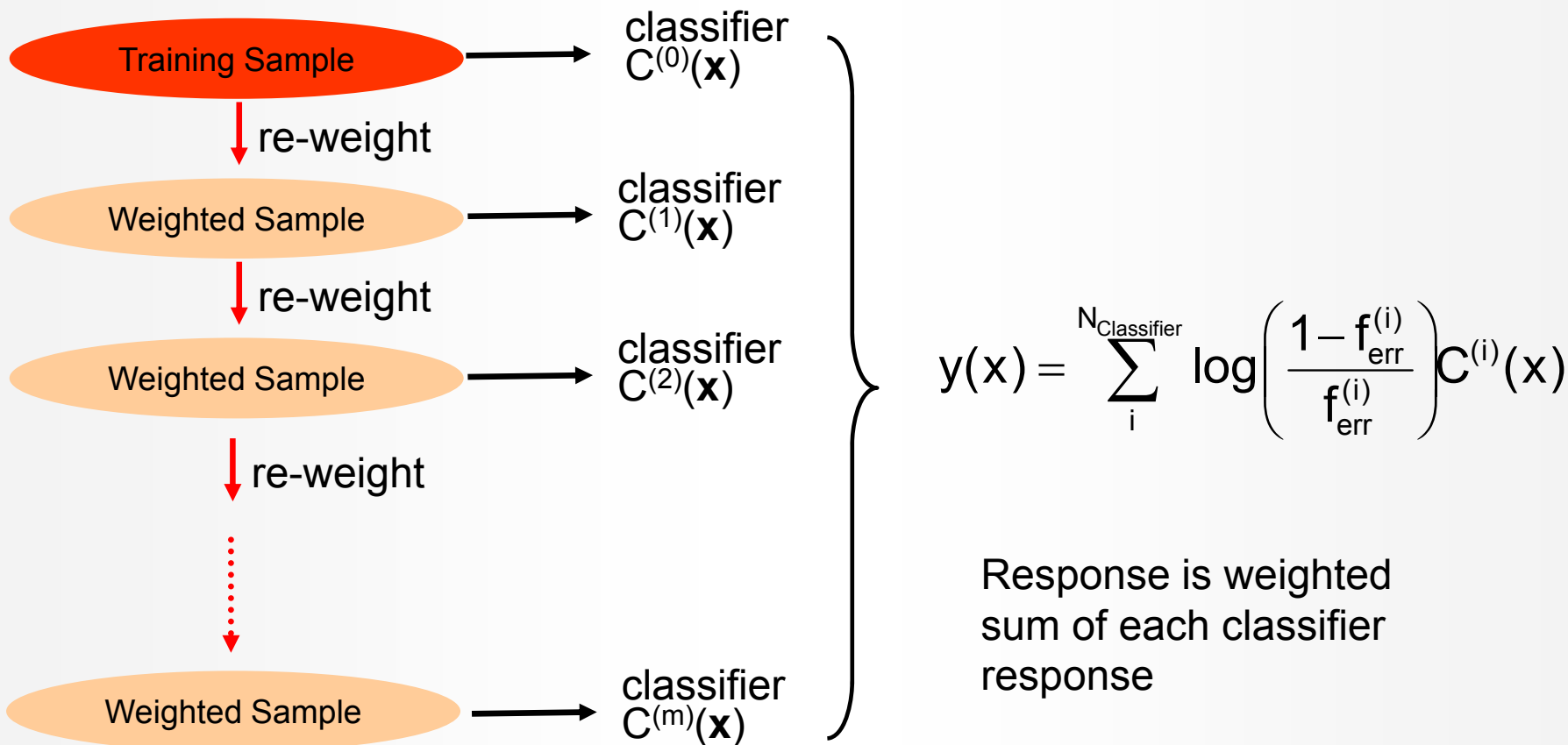


- Non-linear cases:
 - Transform variables into higher dim. space where a linear boundary can fully separate the data
 - Explicit transformation not required: use kernel functions to approximate scalar products between transformed vectors in the higher dim. space
 - Choose Kernel and fit the hyperplane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



Generalised Classifier Boosting

- Principle (just as in BDT): multiple training cycles, each time wrongly classified events get a higher event weight



Boosting will be interesting especially for Methods like Cuts, MLP, and SVM

Categorising Classifiers

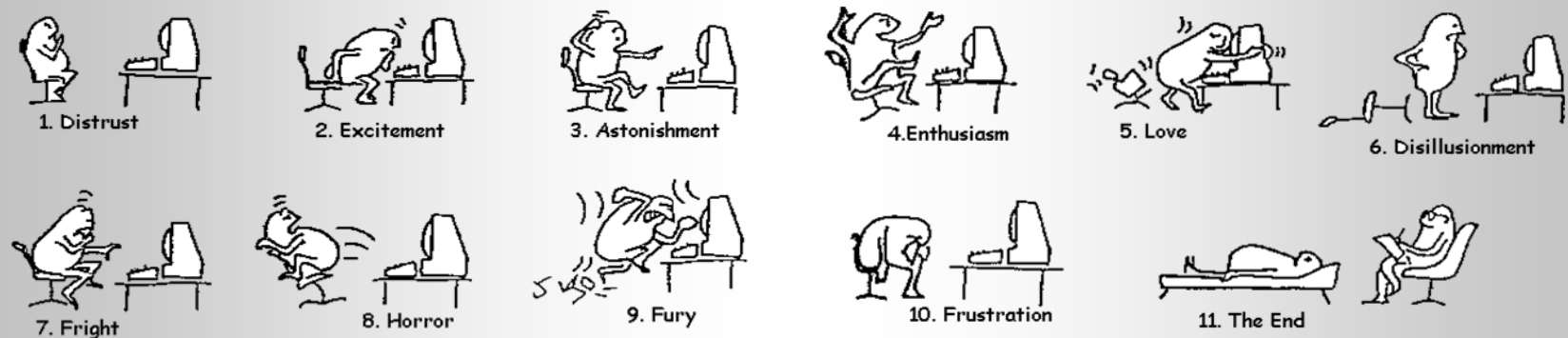
- Multivariate training samples often have distinct sub-populations of data
 - A detector element may only exist in the barrel, but not in the endcaps
 - A variable may have different distributions in barrel, overlap, endcap regions
- Ignoring this dependence creates correlations between variables, which must be learned by the classifier
 - Classifiers such as the projective likelihood, which do not account for correlations, significantly loose performance if the sub-populations are not separated
- Categorisation means splitting the data sample into categories defining disjoint data samples with the following (idealised) properties:
 - Events belonging to the same category are statistically indistinguishable
 - Events belonging to different categories have different properties
- In TMVA: All categories are treated independently for training and application (transparent for user), but evaluation is done for the whole data sample

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](#)

A Simple Example for *Training*

```
void TMVClassification( )
```

```
{
```

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

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

← 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:IV" );
```

```
  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 TMVClassificationApplication( )
```

```
{
```

```
    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;  
}
```

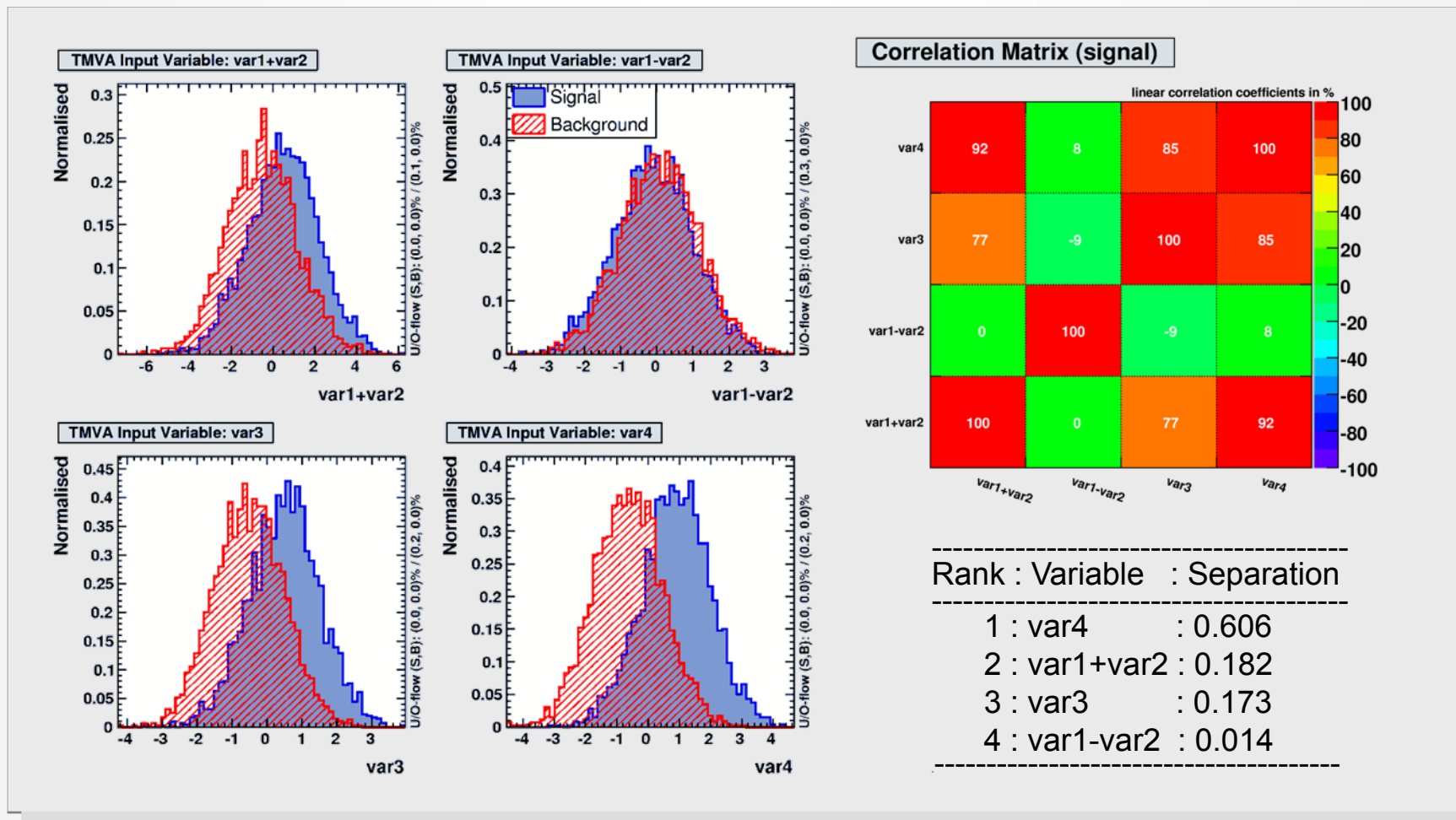
→ [TMVA tutorial](#)

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
 - Alternating
 - User defined training and test trees
- Preprocessing of input variables (e.g., decorrelation)

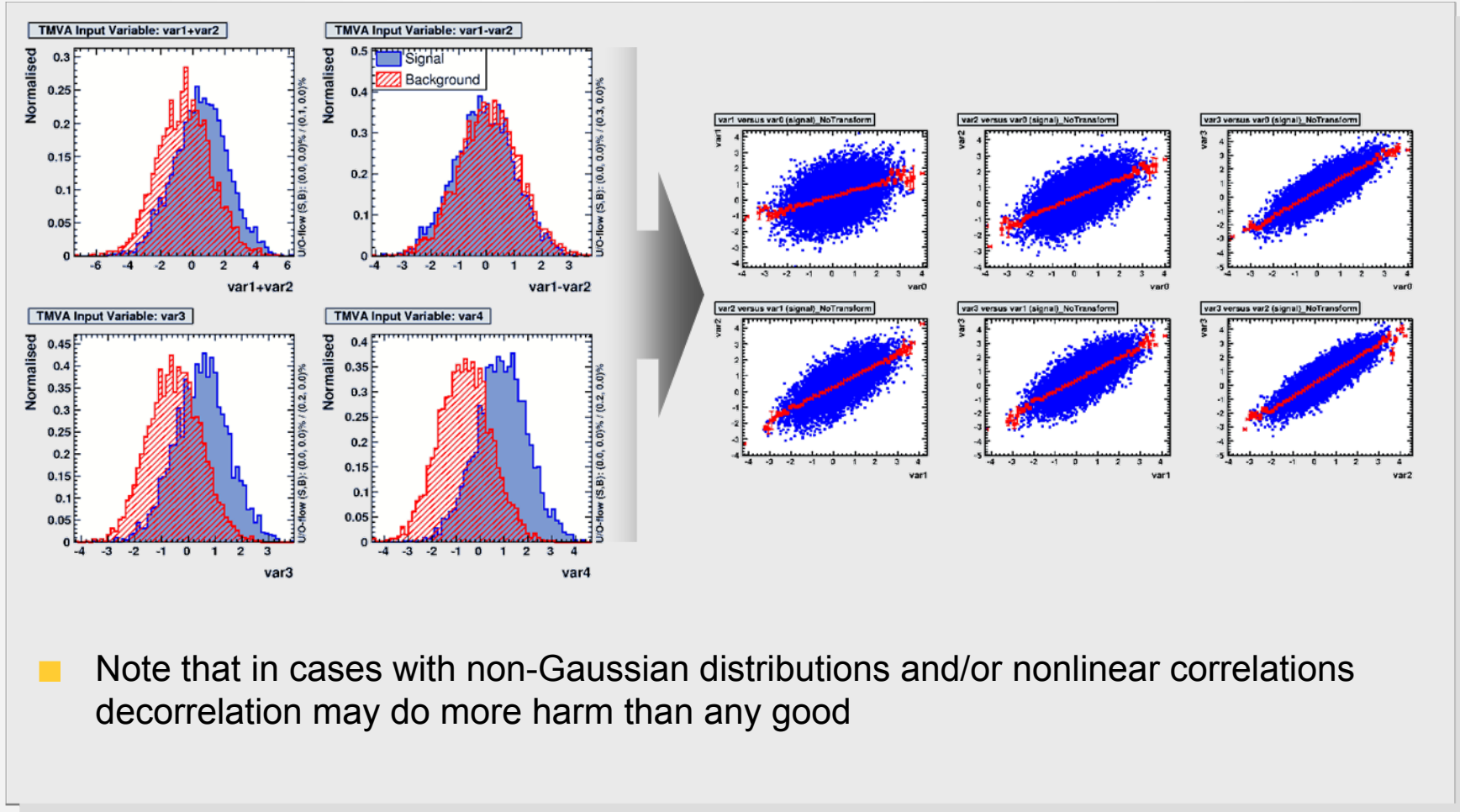
A Toy Example (idealized)

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



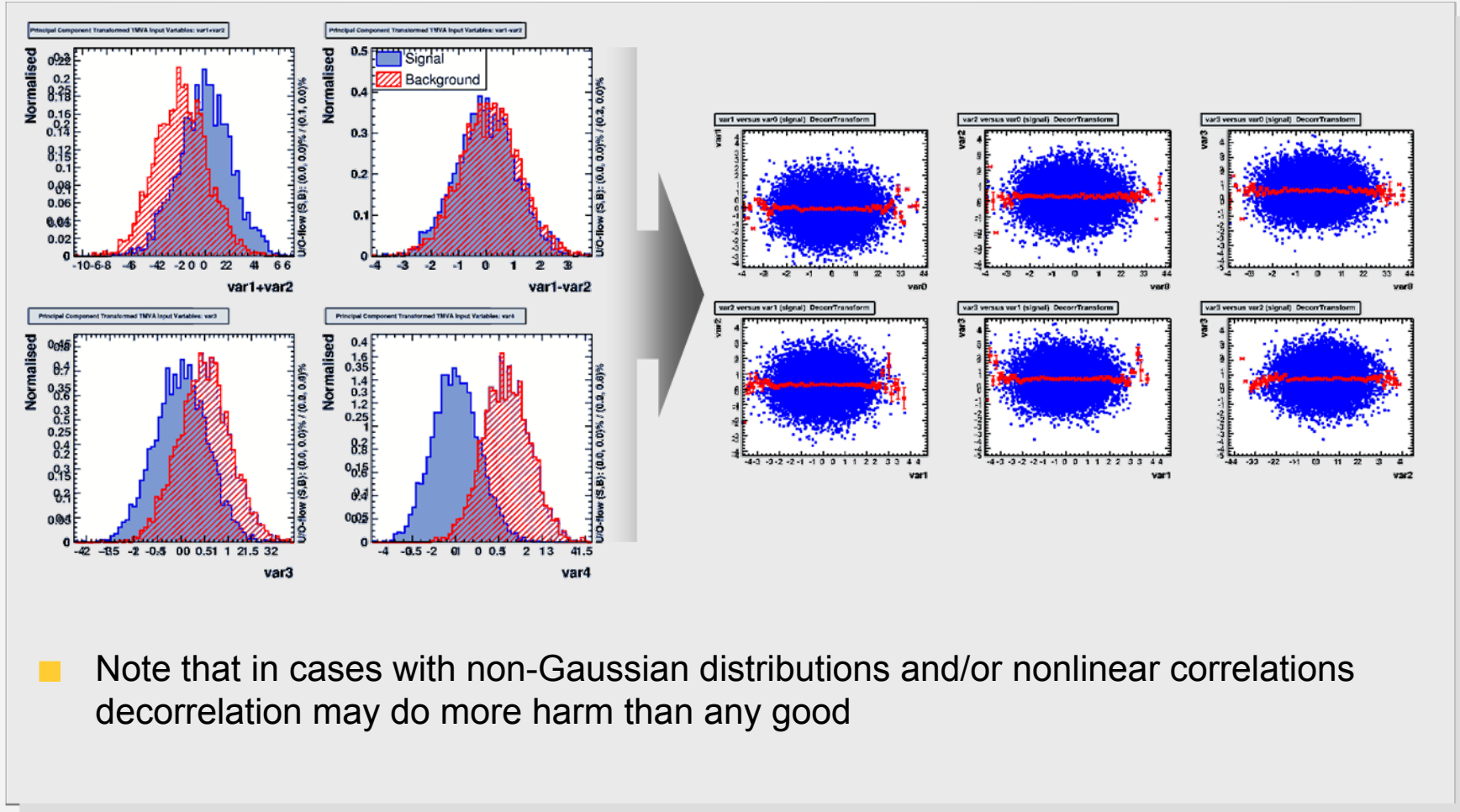
Preprocessing the Input Variables

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



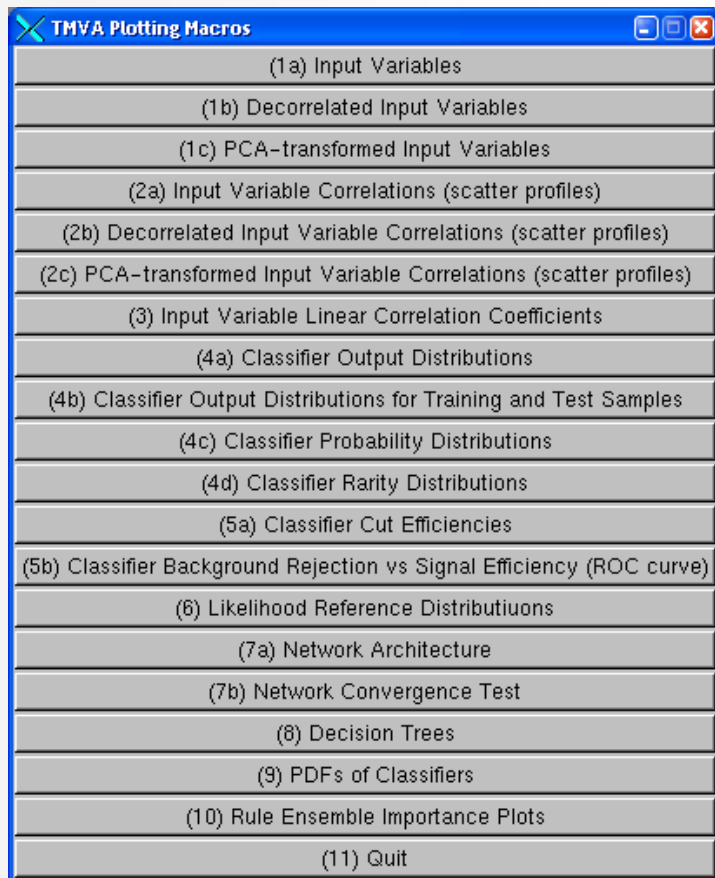
Preprocessing the Input Variables

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



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

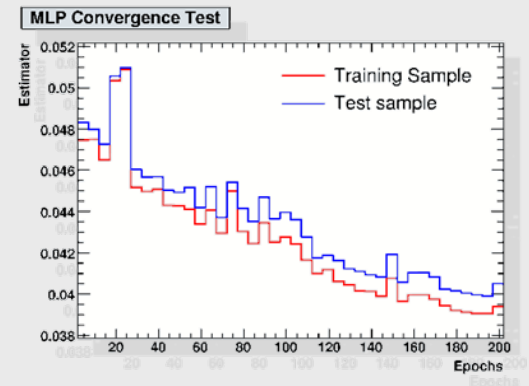
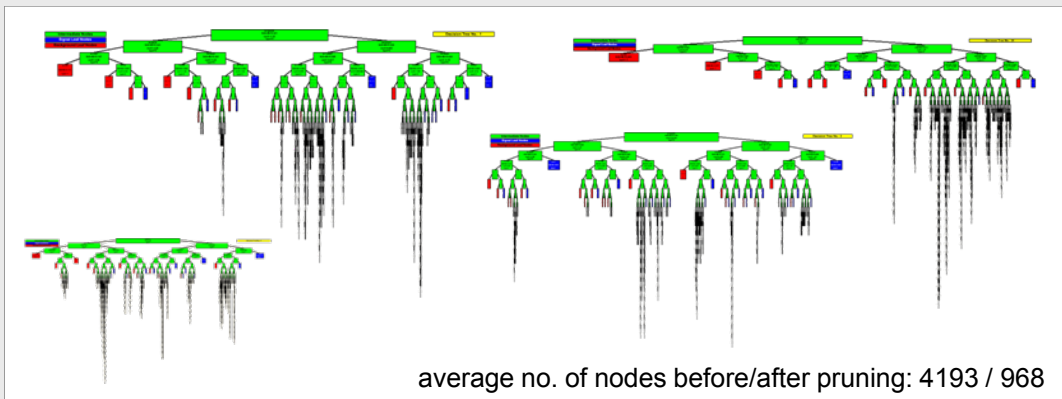
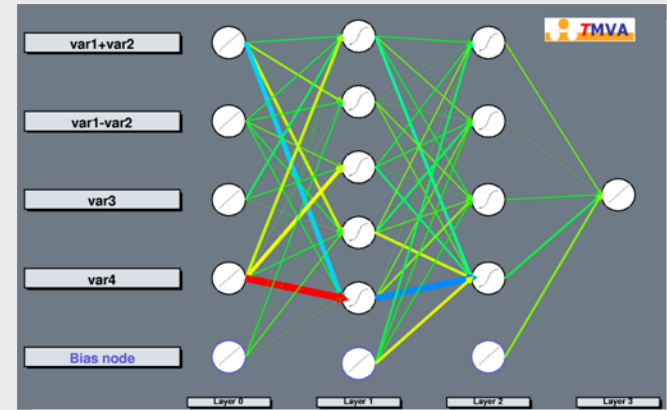
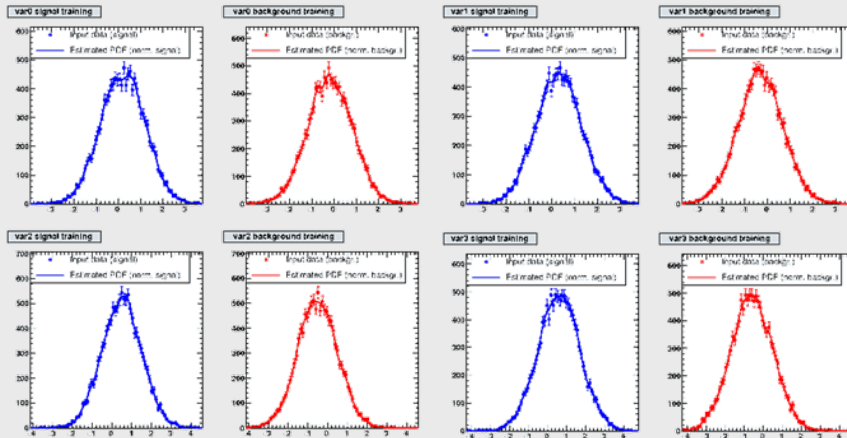
Classifier-specific plots:

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

• Visualise decision trees

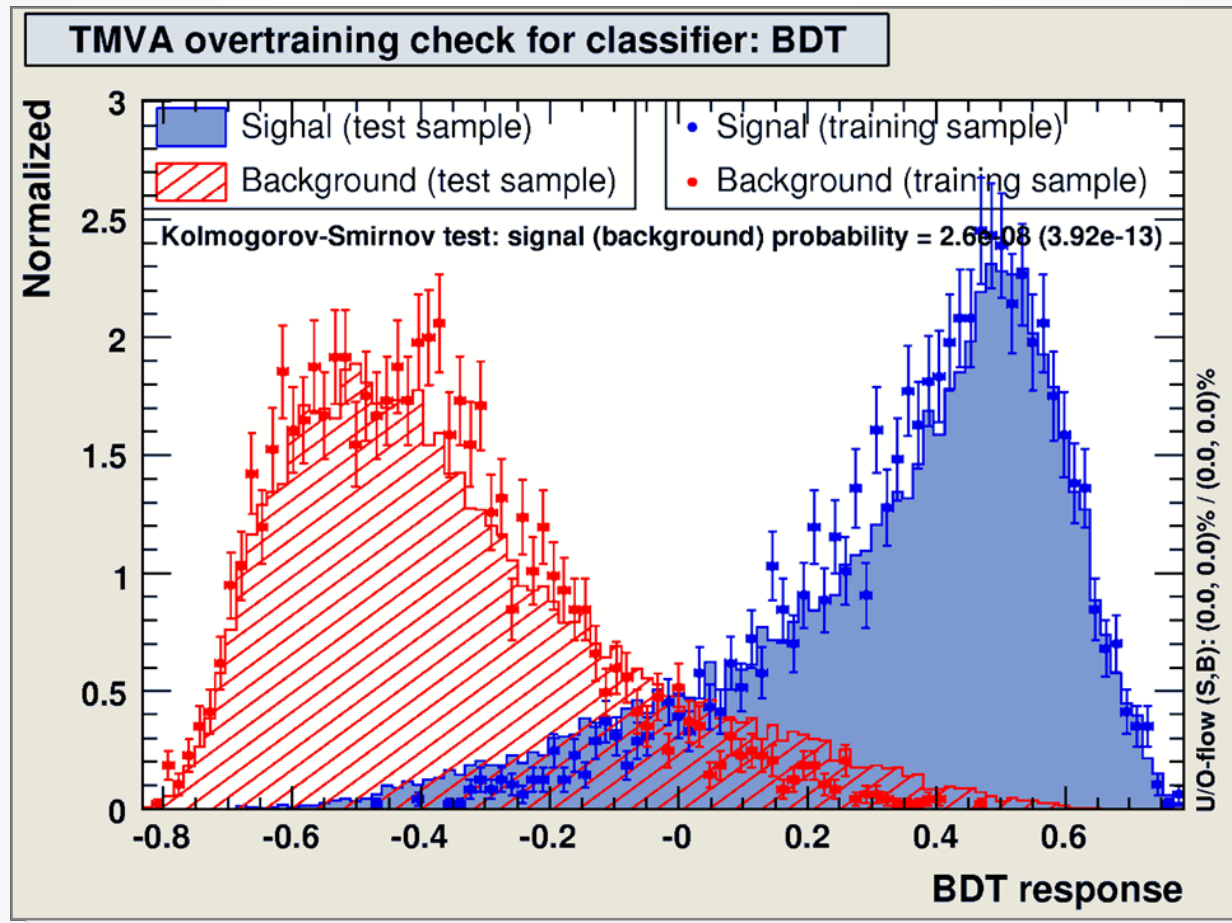
Evaluating the Classifier Training (I)

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



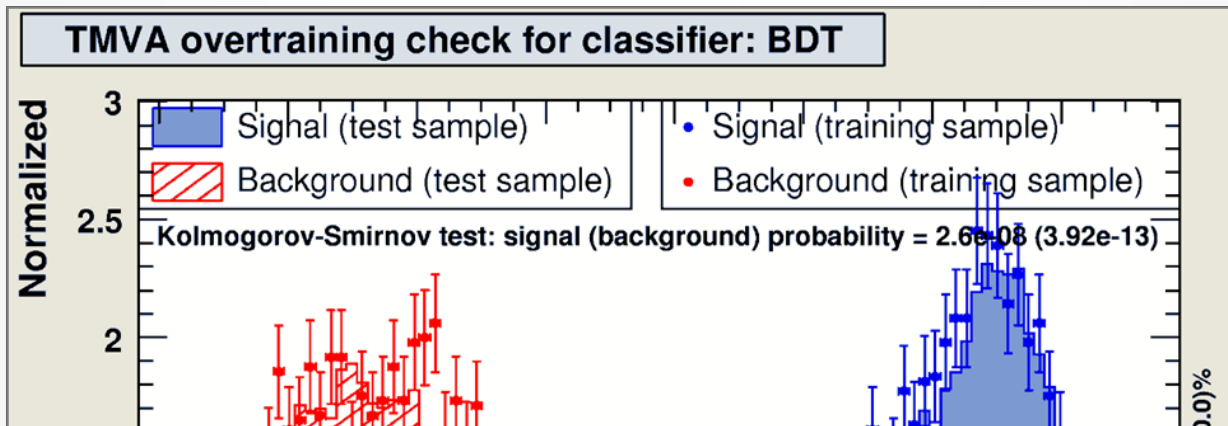
Evaluating the Classifier Training (II)

- Check for overtraining: classifier output for test *and* training samples ...



Evaluating the Classifier Training (II)

- Check for overtraining: classifier output for test *and* training samples ...

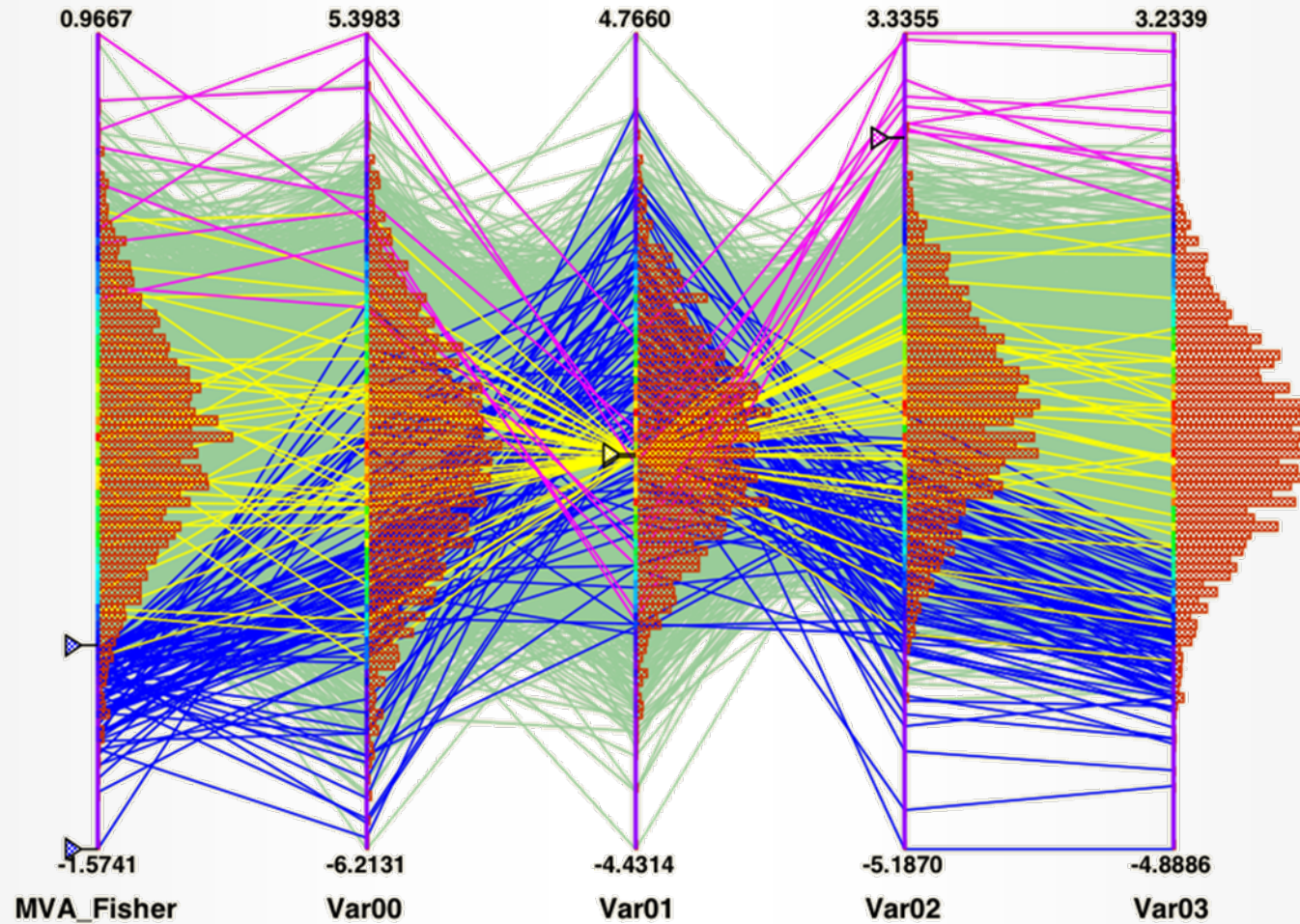


- Remark on **overtraining**

- 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, ...

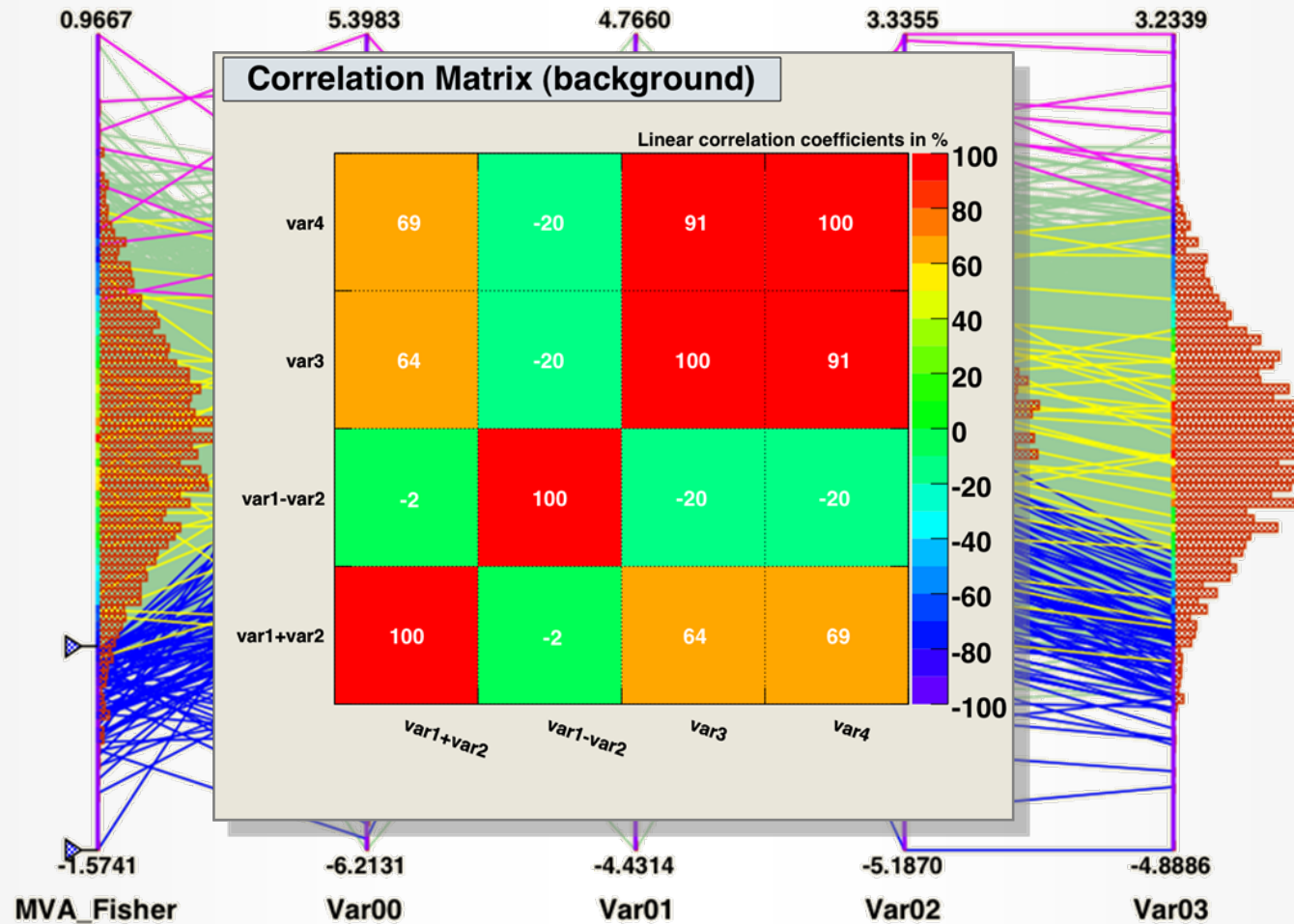
Evaluating the Classifier Training (III)

■ *Parallel Coordinates (ROOT class)*




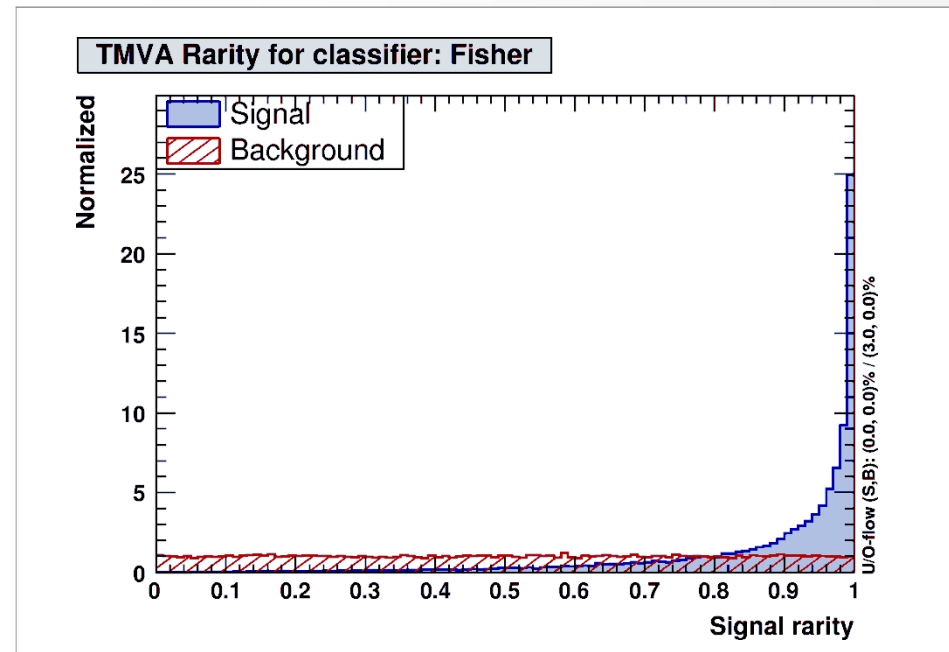
Evaluating the Classifier Training (III)

■ Parallel Coordinates (ROOT class)



Evaluating the Classifier Training (IV)

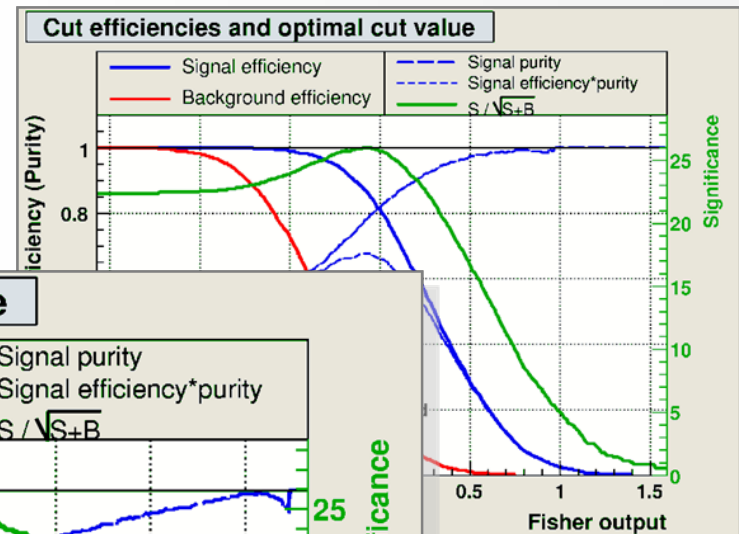
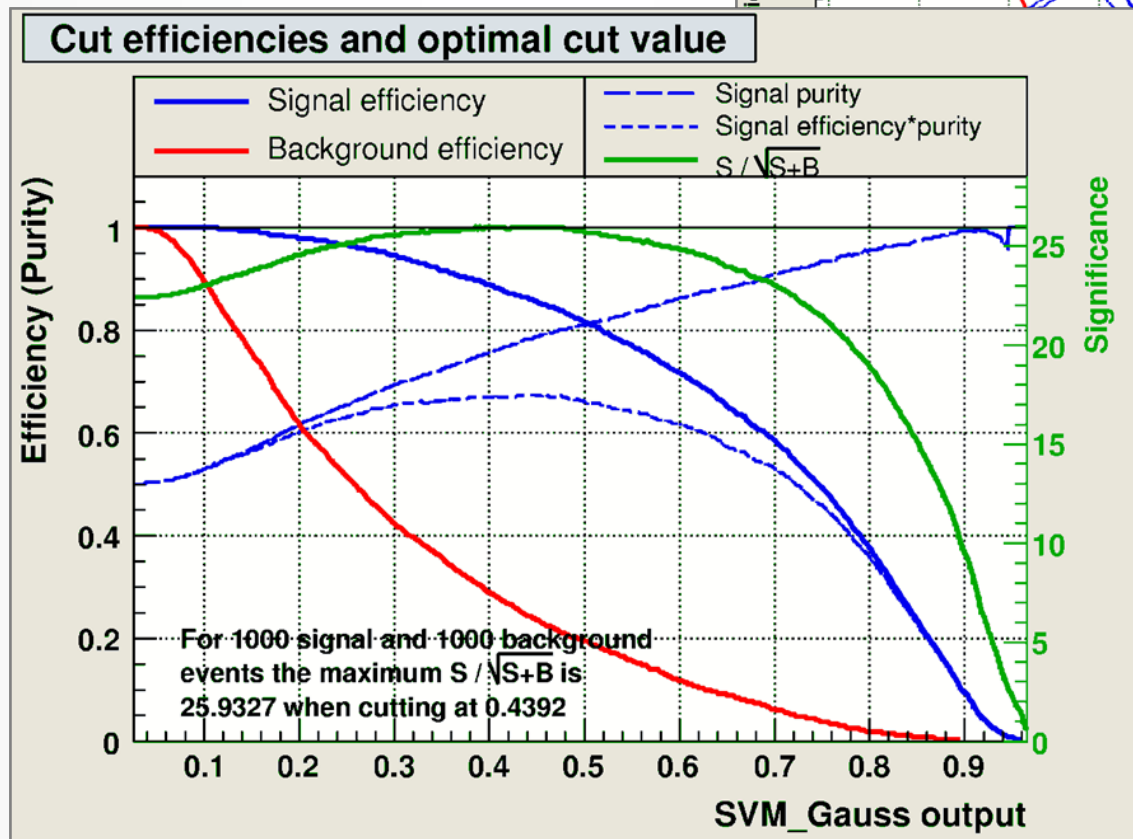
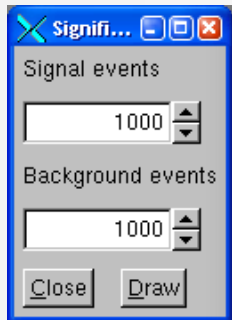
- There is no unique way to express the performance of a classifier
→ several benchmark quantities computed by **TMVA**
 - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
 - The *Separation*: $\frac{1}{2} \int \frac{(\hat{y}_S(y) - \hat{y}_B(y))^2}{\hat{y}_S(y)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



Evaluating the Classifier Training (V)

■ Optimal cut for each classifiers ...

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



Input Variable Ranking



```
--- 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 Training (VII) (taken from *TMVA* output...)

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

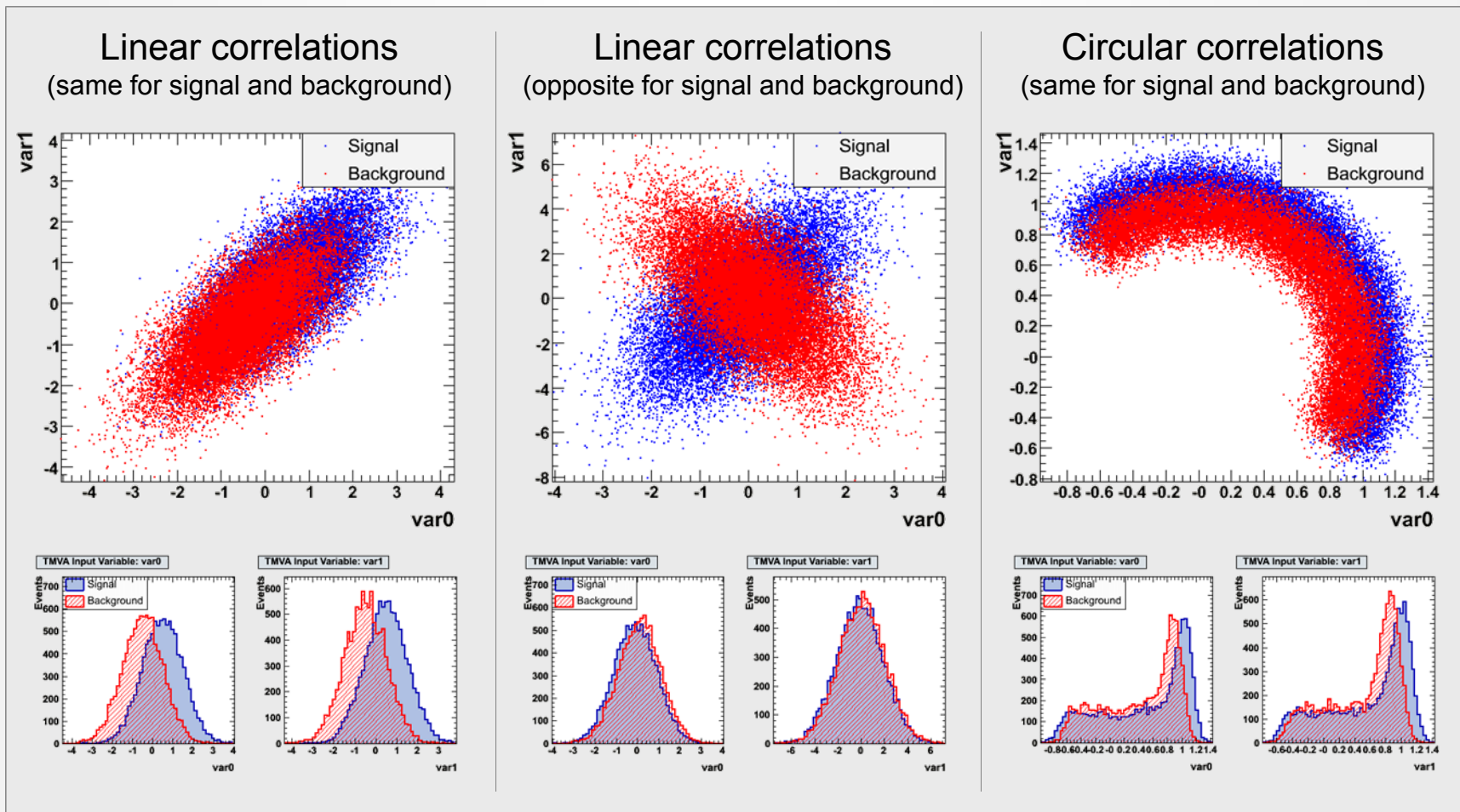
MVA Methods:	Signal efficiency at bkg eff. (error):				Sepa- ration:	Signifi- cance:
	@B=0.01	@B=0.10	@B=0.30	Area		
Fisher	: 0.268(03)	0.653(03)	0.873(02)	0.882	0.444	1.189
MLP	: 0.266(03)	0.656(03)	0.873(02)	0.882	0.444	1.260
LikelihoodD	: 0.259(03)	0.649(03)	0.871(02)	0.880	0.441	1.251
PDERS	: 0.223(03)	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



More Toy Examples

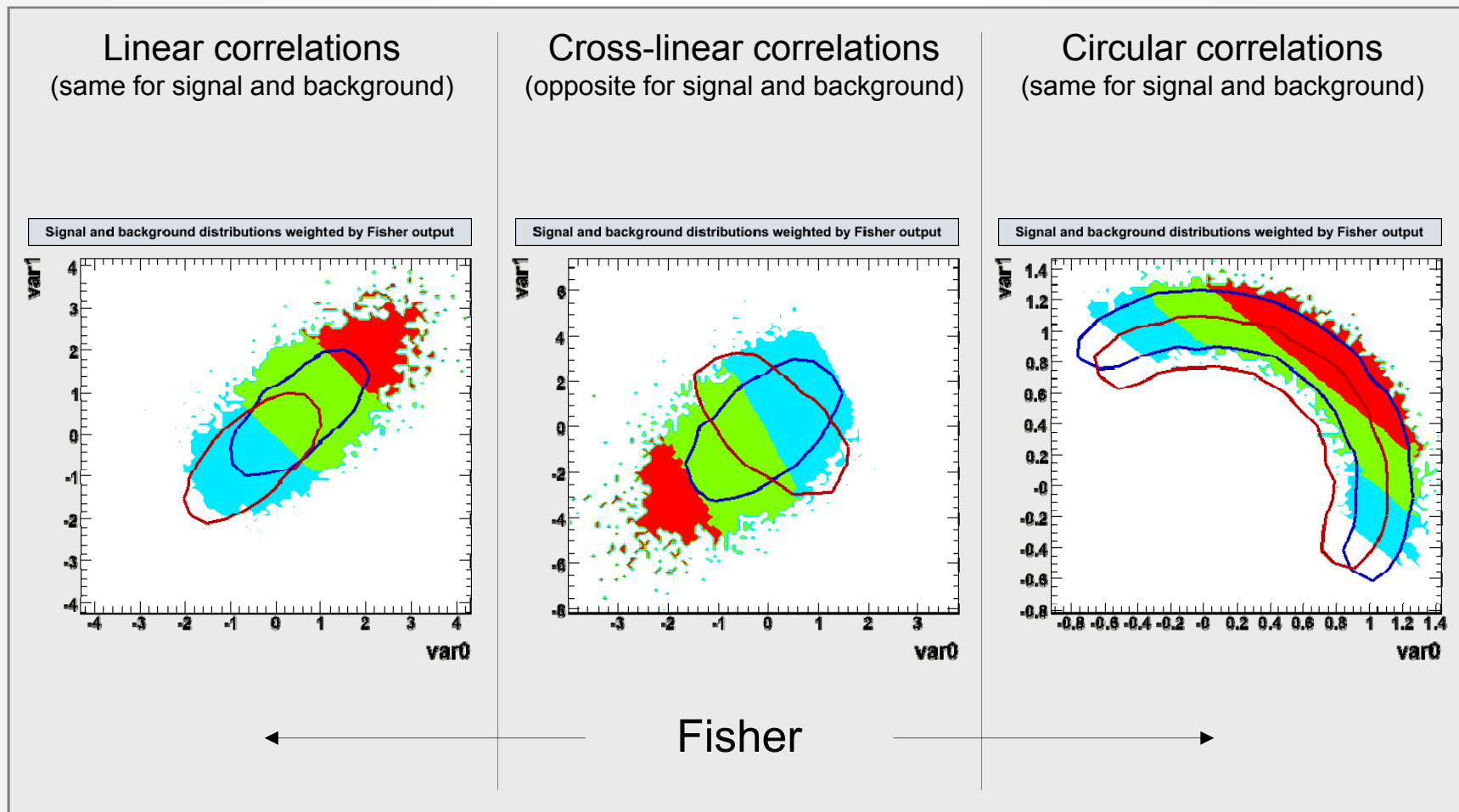
More Toys: Linear-, Cross-, Circular Correlations

- Illustrate the behaviour of linear and nonlinear classifiers



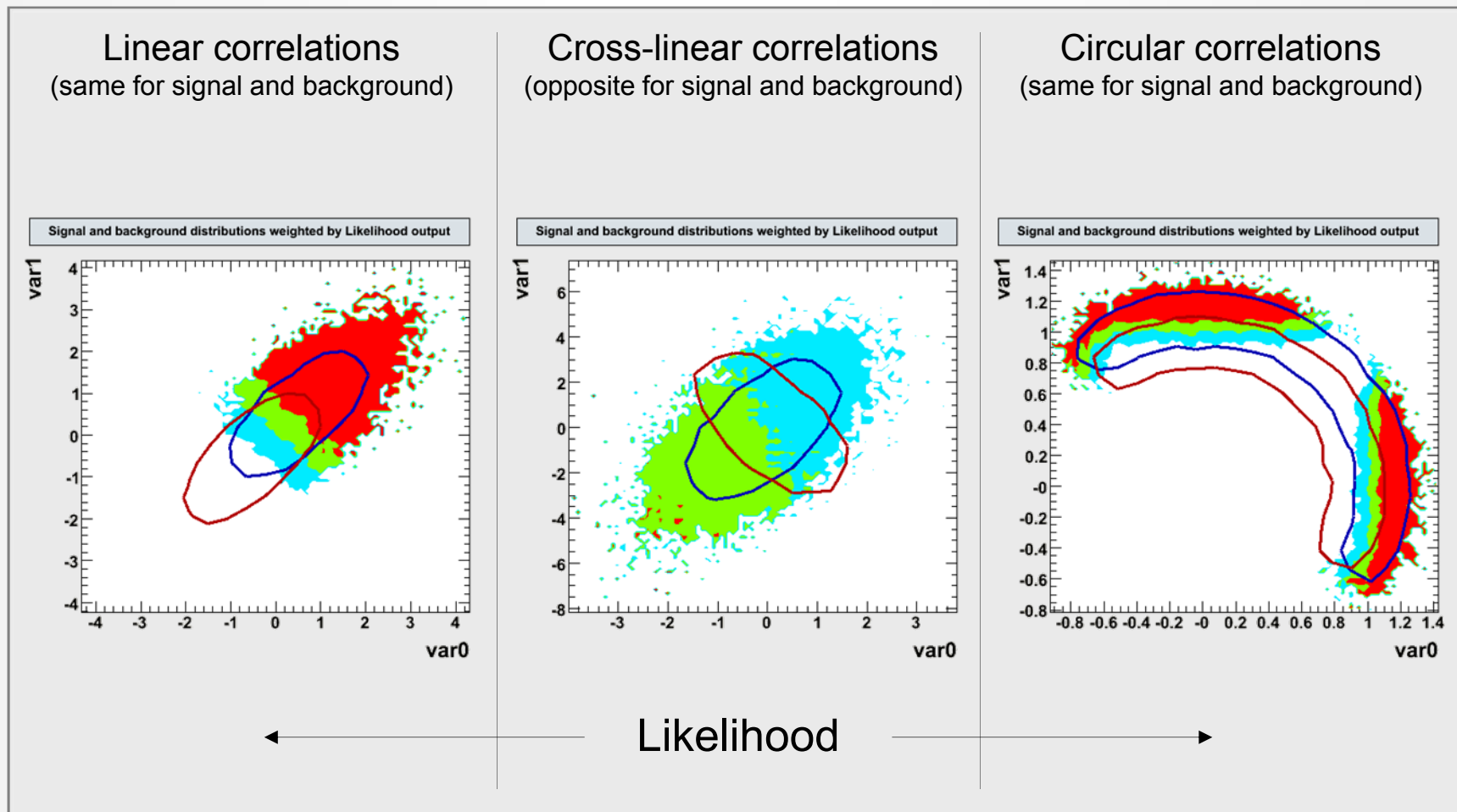
Weight Variables by Classifier Output

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



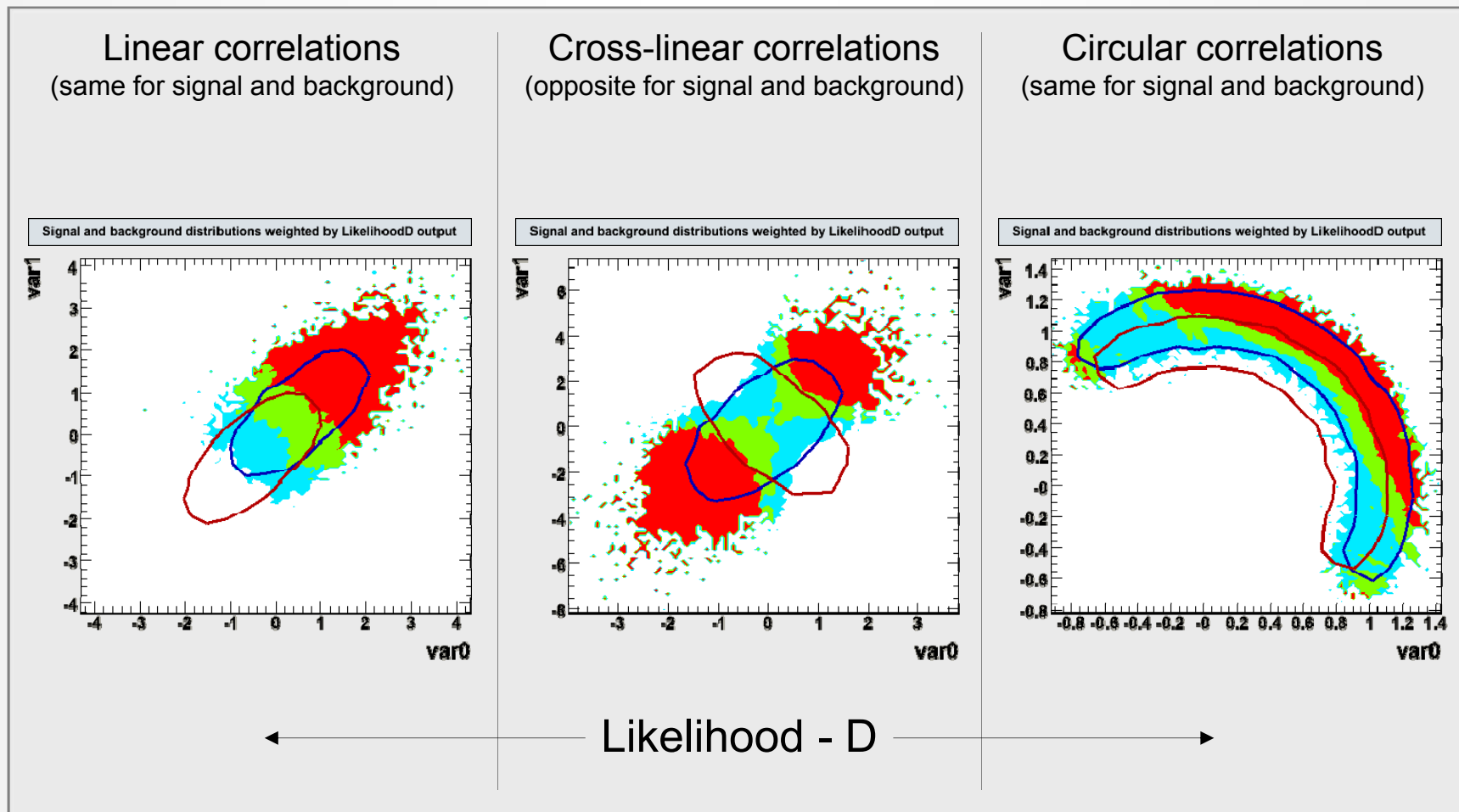
Weight Variables by Classifier Output

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



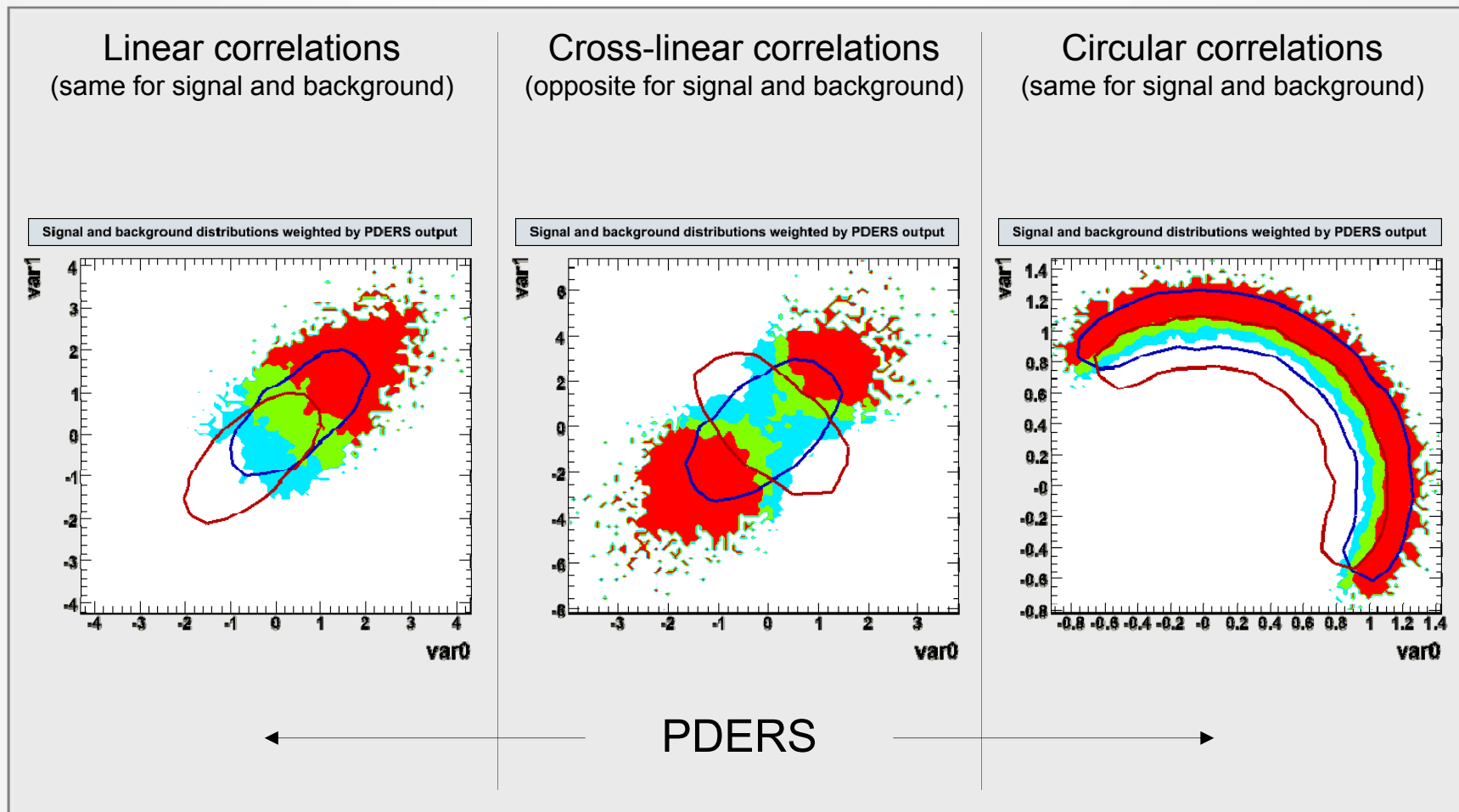
Weight Variables by Classifier Output

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



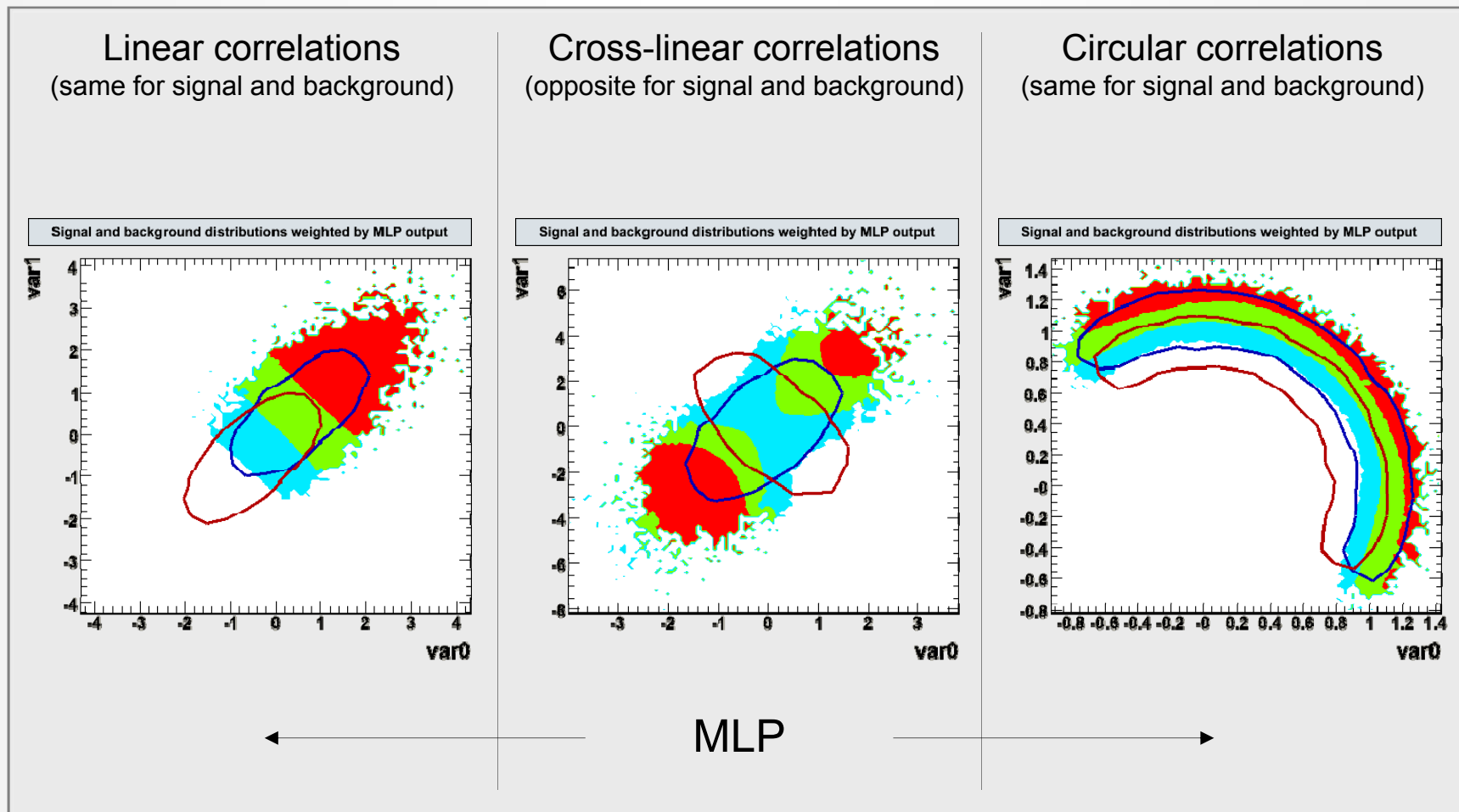
Weight Variables by Classifier Output

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



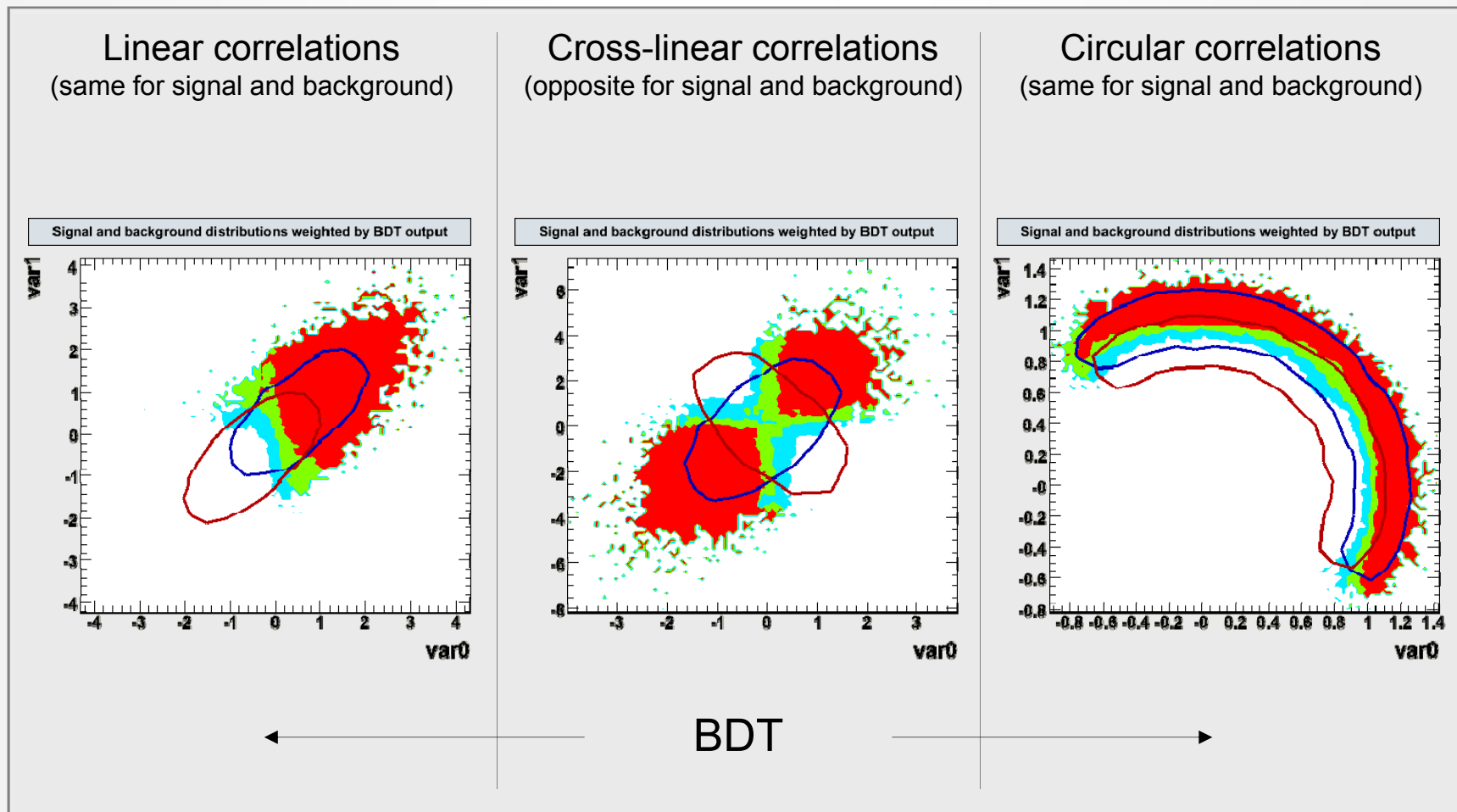
Weight Variables by Classifier Output

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



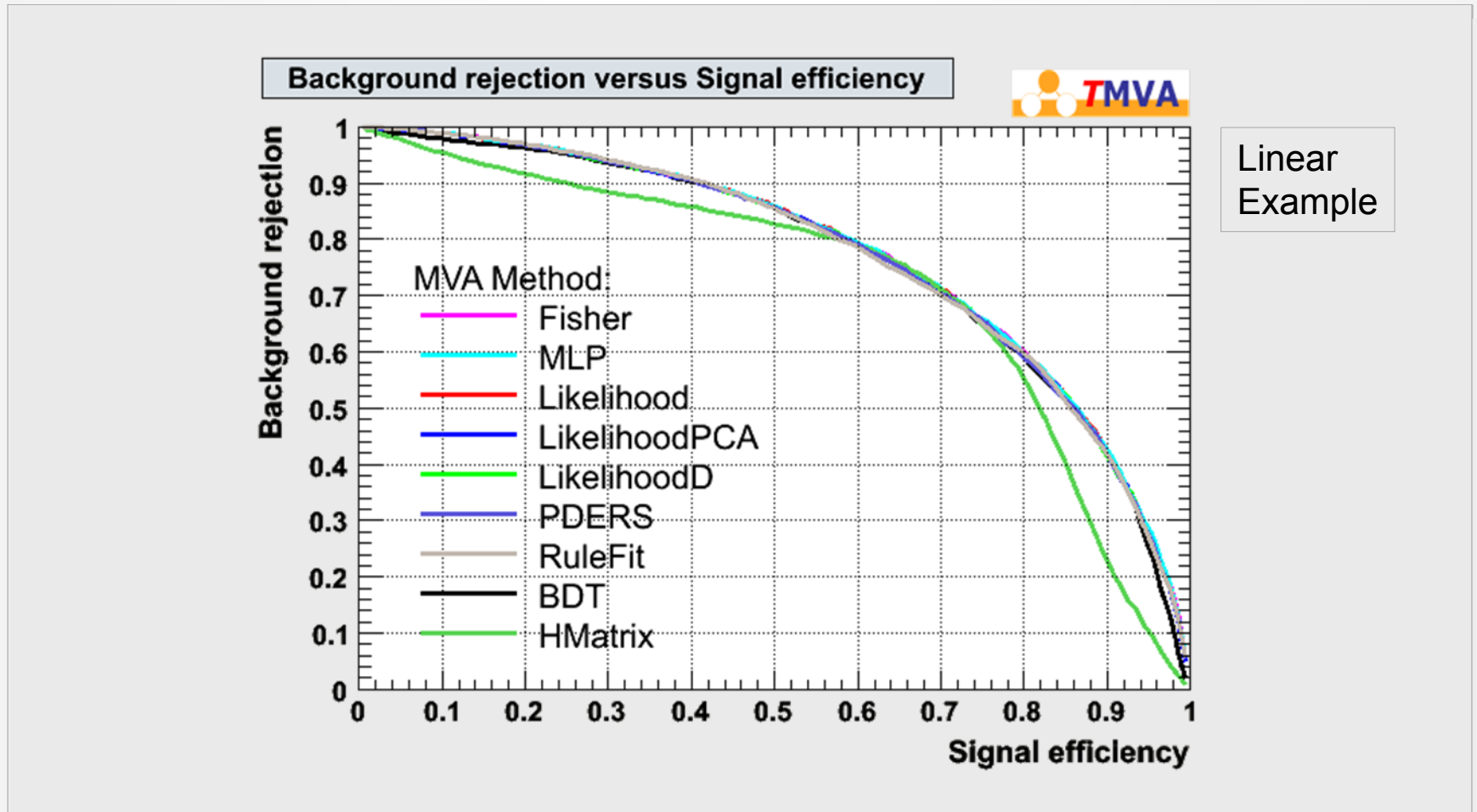
Weight Variables by Classifier Output

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



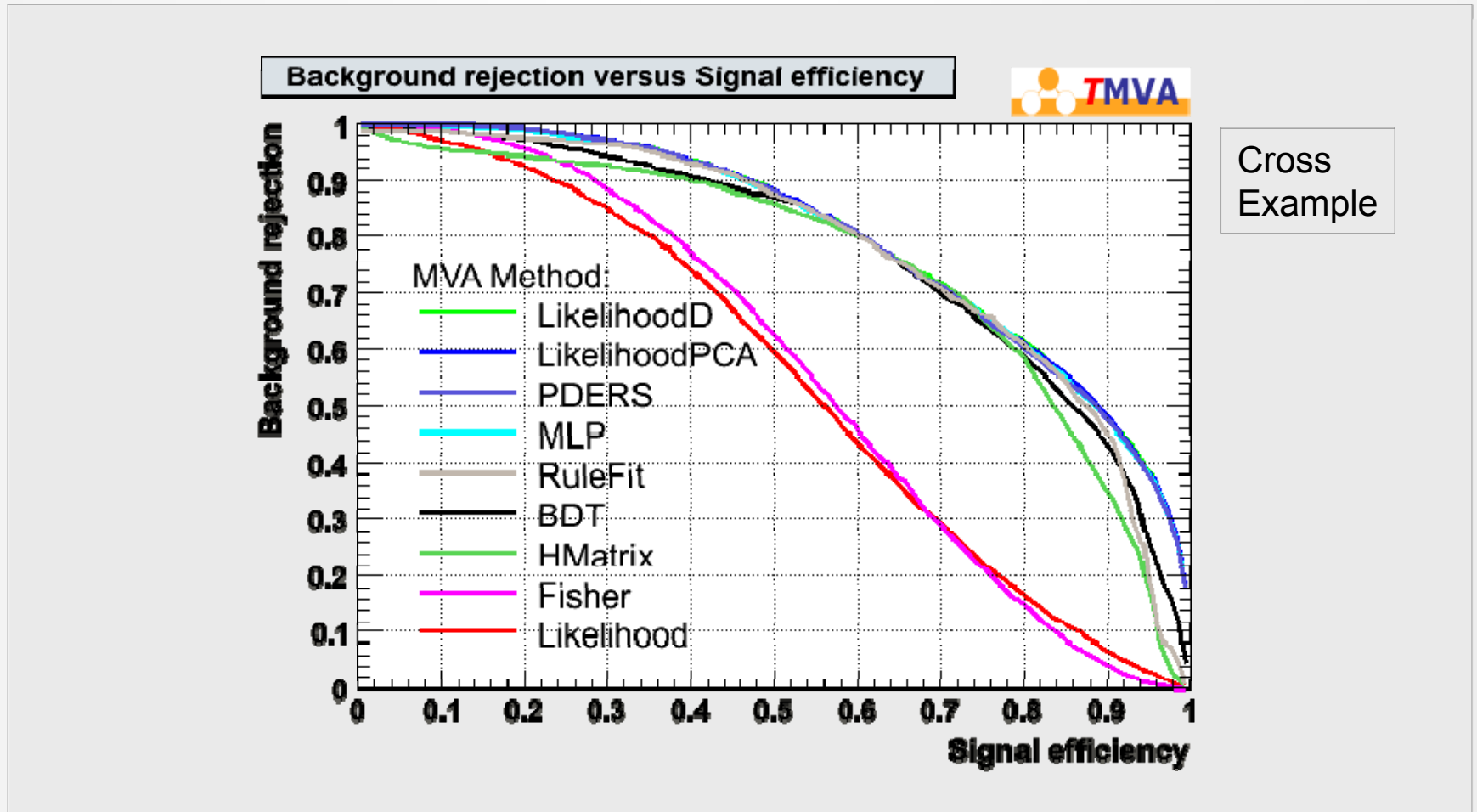
Final Classifier Performance

- Background rejection versus signal efficiency curve:



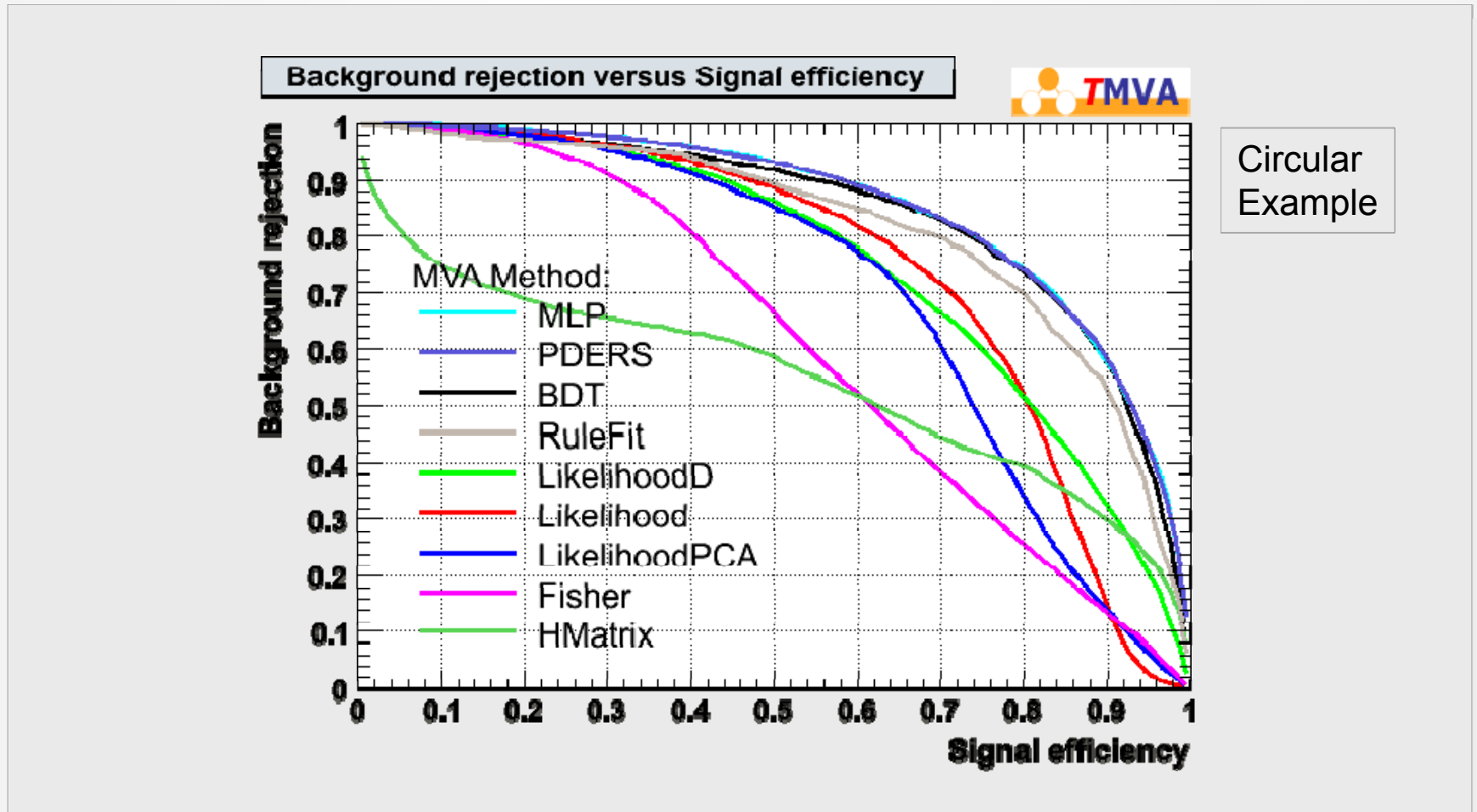
Final Classifier Performance

- Background rejection versus signal efficiency curve:



Final Classifier Performance

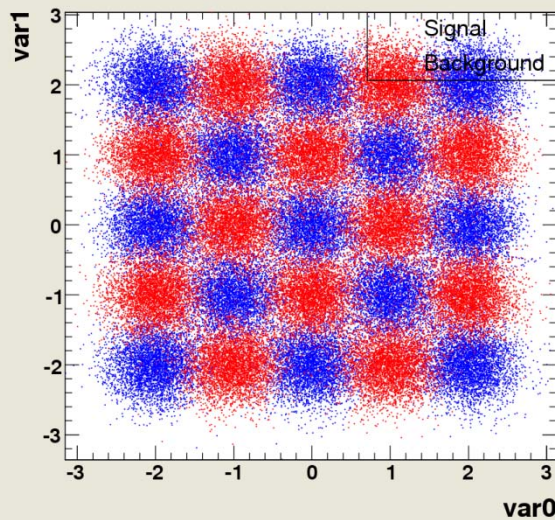
- Background rejection versus signal efficiency curve:



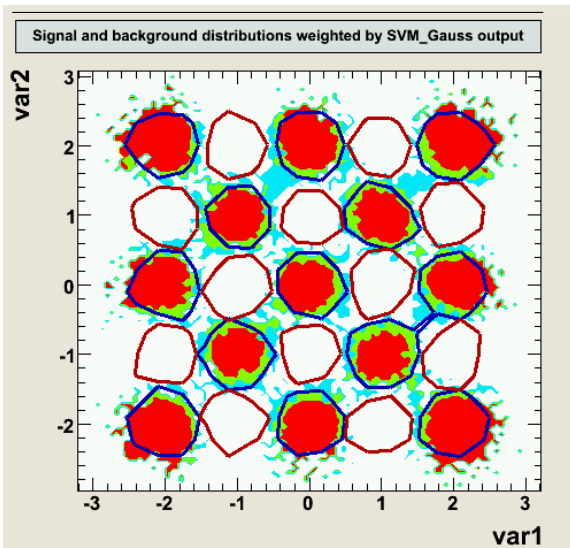
Circular Example

The *Schachbrett* Toy (chess board)

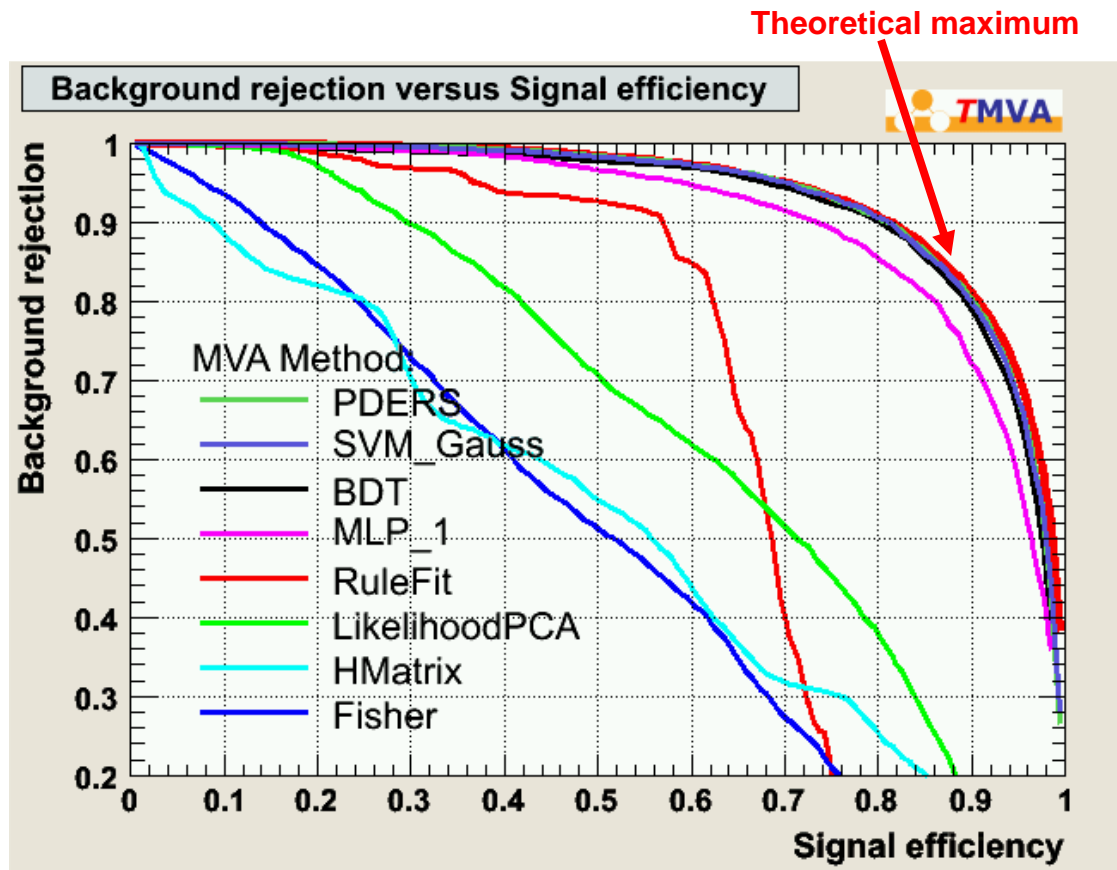
Event Distribution



Events weighted by SVM response

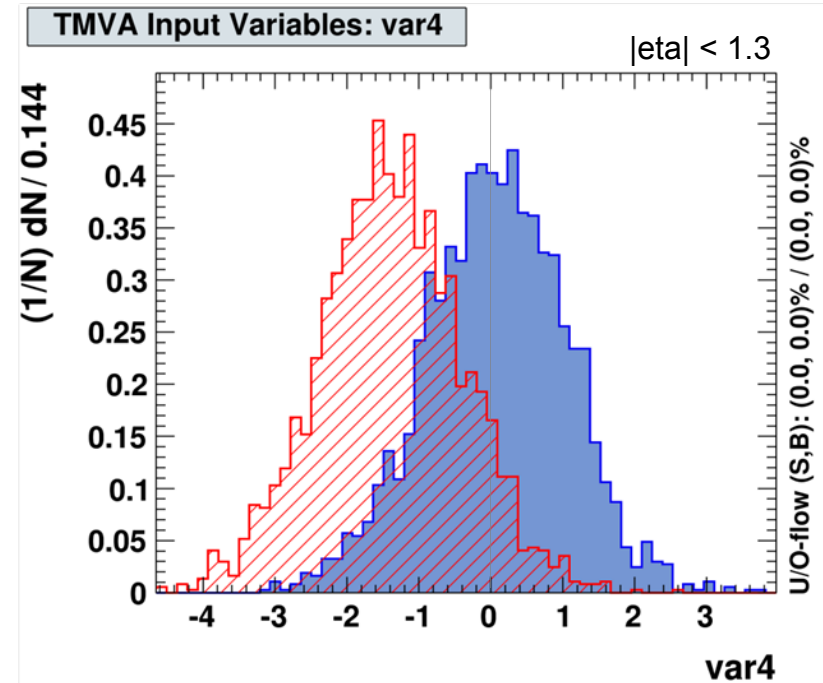
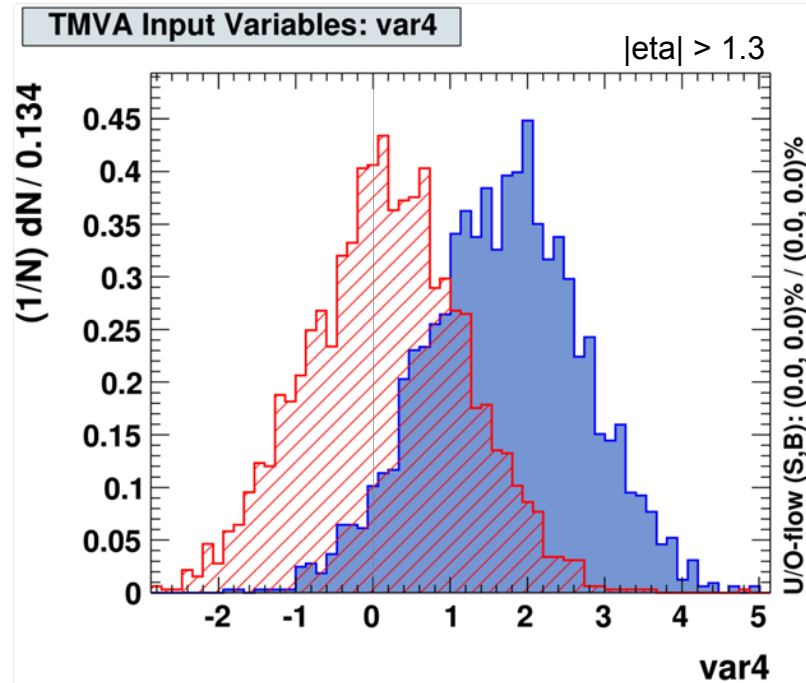


- Performance achieved without parameter adjustments: nearest Neighbour and BDTs are best “out of the box”
- After some parameter tuning, also SVM und ANN(MLP) perform



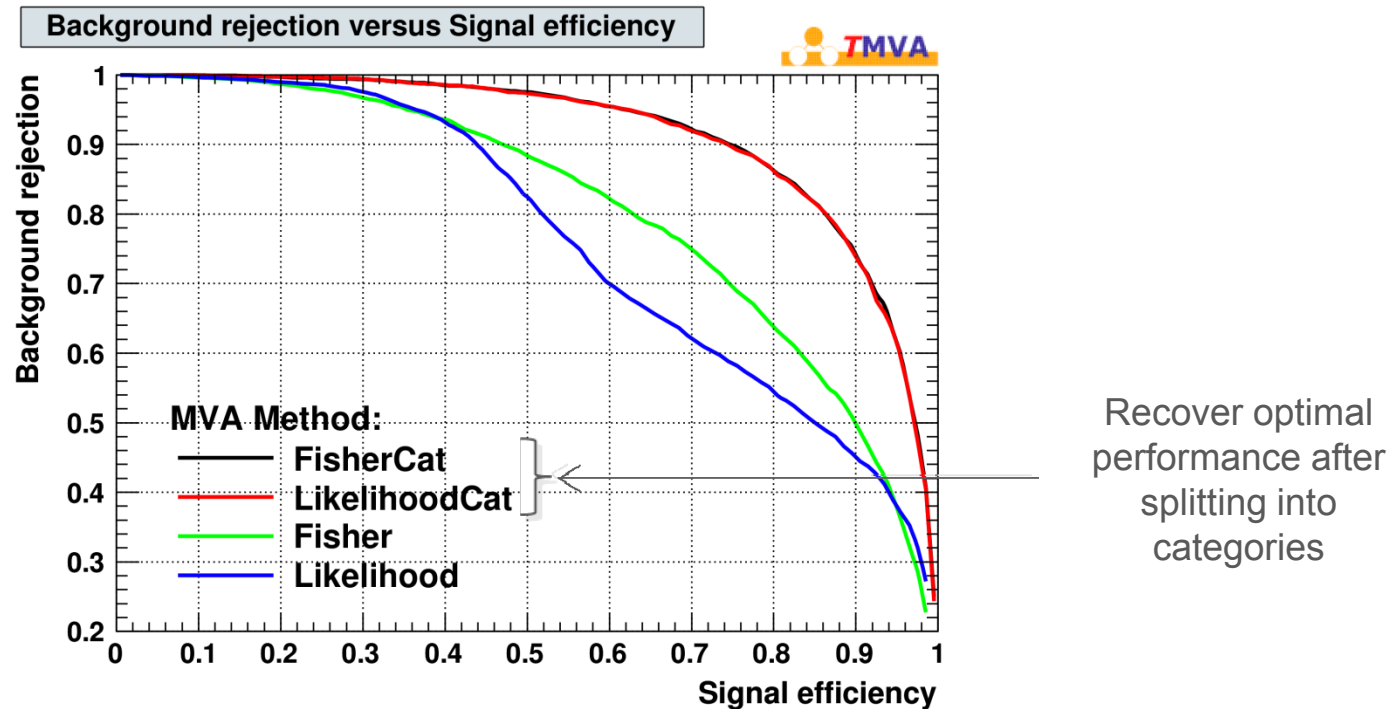
“Categorising” Classifiers

- Let's try our standard example of 4 Gaussian-distributed input variables:
 - Now, “var4” depends on a new variable “eta” (which may not be used for classification)
 - for $|\eta| > 1.3$ the Signal and Background Gaussian means are shifted w.r.t. $|\eta| < 1.3$



Categorising Classifiers

- Let's try our standard example of 4 Gaussian-distributed input variables:
 - Now, “var4” depends on a new variable “eta” (which may not be used for classification)
 - for $|\text{eta}| > 1.3$ the Signal and Background Gaussian means are shifted w.r.t. $|\text{eta}| < 1.3$



The category technique is heavily used in multivariate likelihood fits, eg, RooFit (RooSimultaneousPdf)

Summary

No Single Best Classifier ...

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

Summary

- MVA's for Classification and Regression
- The most important classifiers implemented in TMVA:
 - reconstructing the PDF and use Likelihood Ratio:
 - Nearest neighbour (Multidimensional Likelihood)
 - Naïve-Bayesian classifier (1dim (projective) Likelihood)
 - fitting directly the decision boundary:
 - Linear discriminant (Fisher)
 - Neuronal Network
 - Support Vector Machine
 - Boosted Decision Tress
- Introduction to TMVA
 - Training
 - Testing/Evaluation
- Toy examples

TMVA Development and Distribution

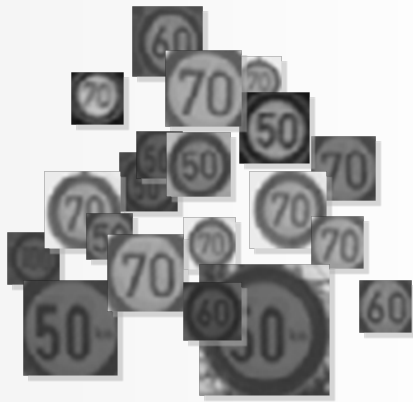
- **TMVA** is now shipped with ROOT, project page on sourceforge
 - Home page <http://tmva.sf.net/>
 - SF project page <http://sf.net/projects/tmva>
 - Mailing list http://sf.net/mail/?group_id=152074
 - Tutorial TWiki <https://twiki.cern.ch/twiki/bin/view/TMVA/WebHome>
- Active project → fast response time on feature requests
 - Currently 6 core developers, and ~25 contributors
 - >3500 downloads since March 2006 (not accounting SVN checkouts and ROOT users)
- Written in C++, relying on core ROOT functionality
- Integrated and distributed with ROOT since ROOT v5.11/03



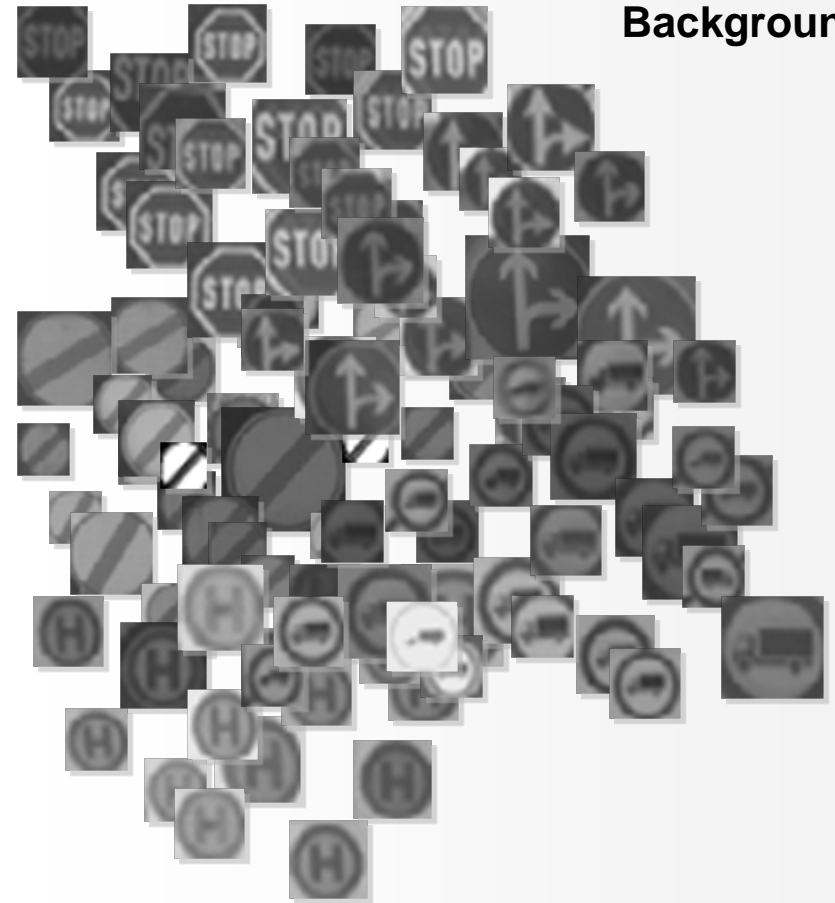
In development

Multi-Class Classification

Signal

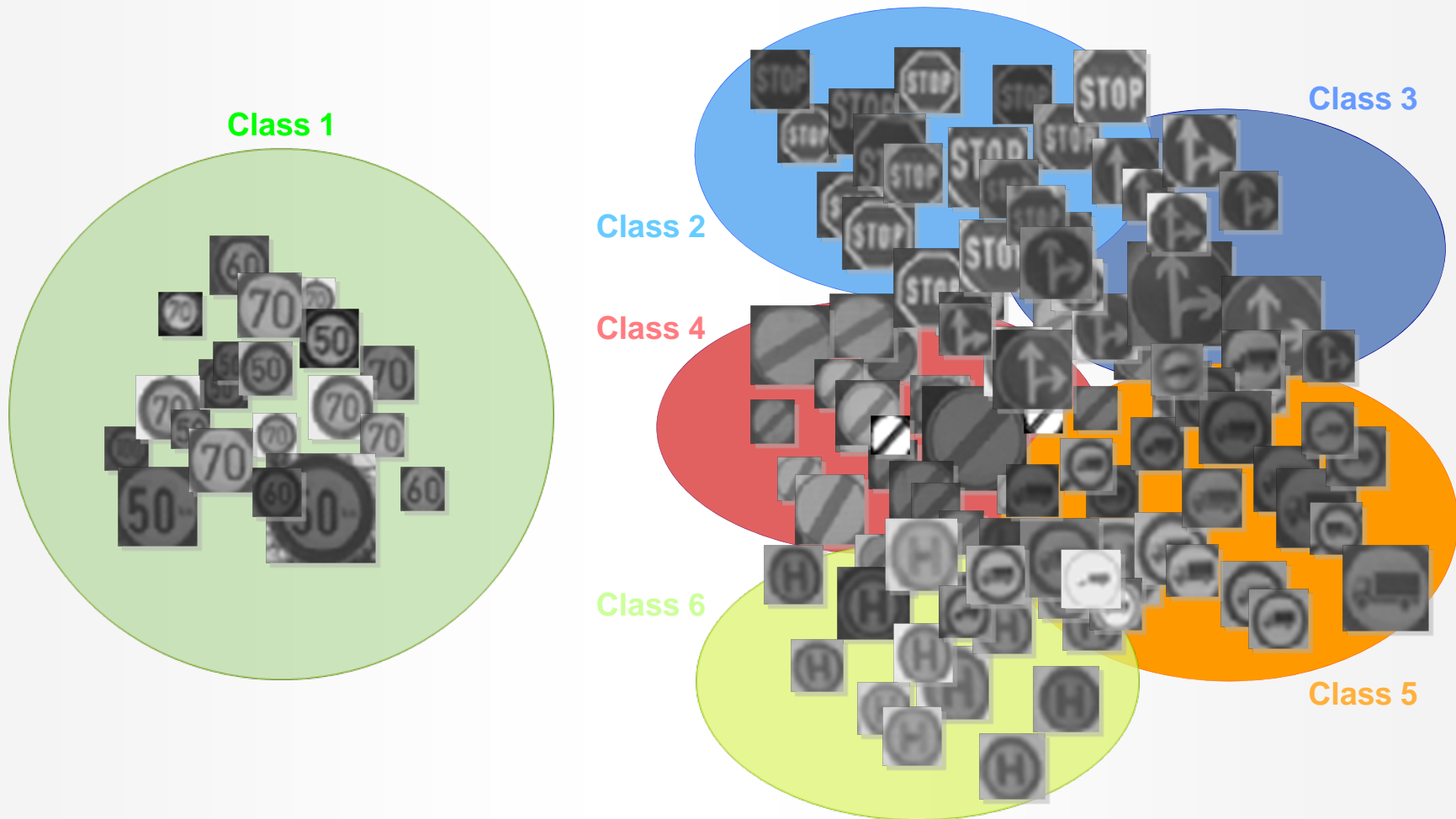


Background



Binary classification: two classes, “signal” and “background”

Multi-Class Classification

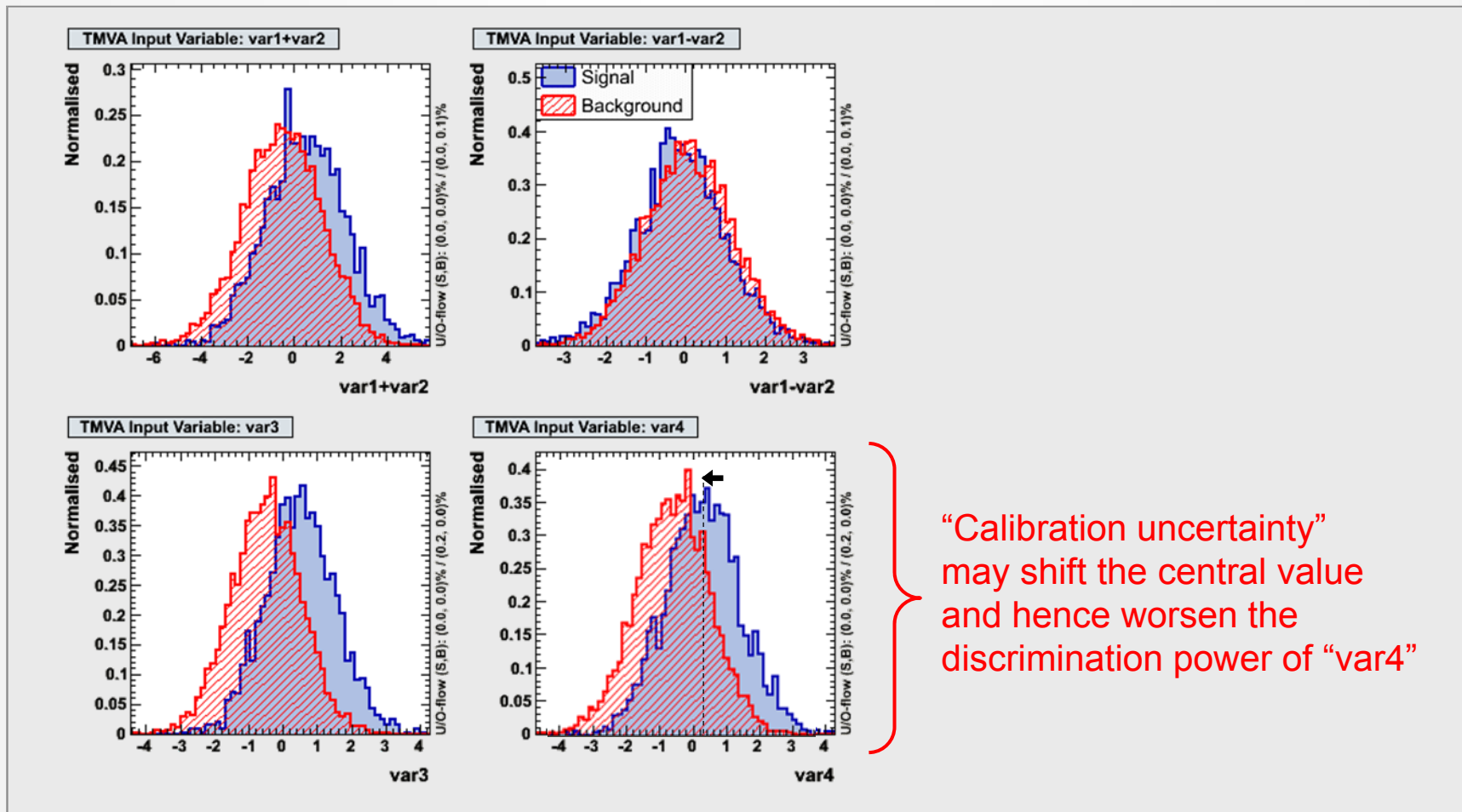


Multi-class classification – natural extension for many classifiers

A (brief) Word on
Systematics
&
Irrelevant Input Variables

Treatment of Systematic Uncertainties

- Assume strongest variable “var4” suffers from systematic uncertainty

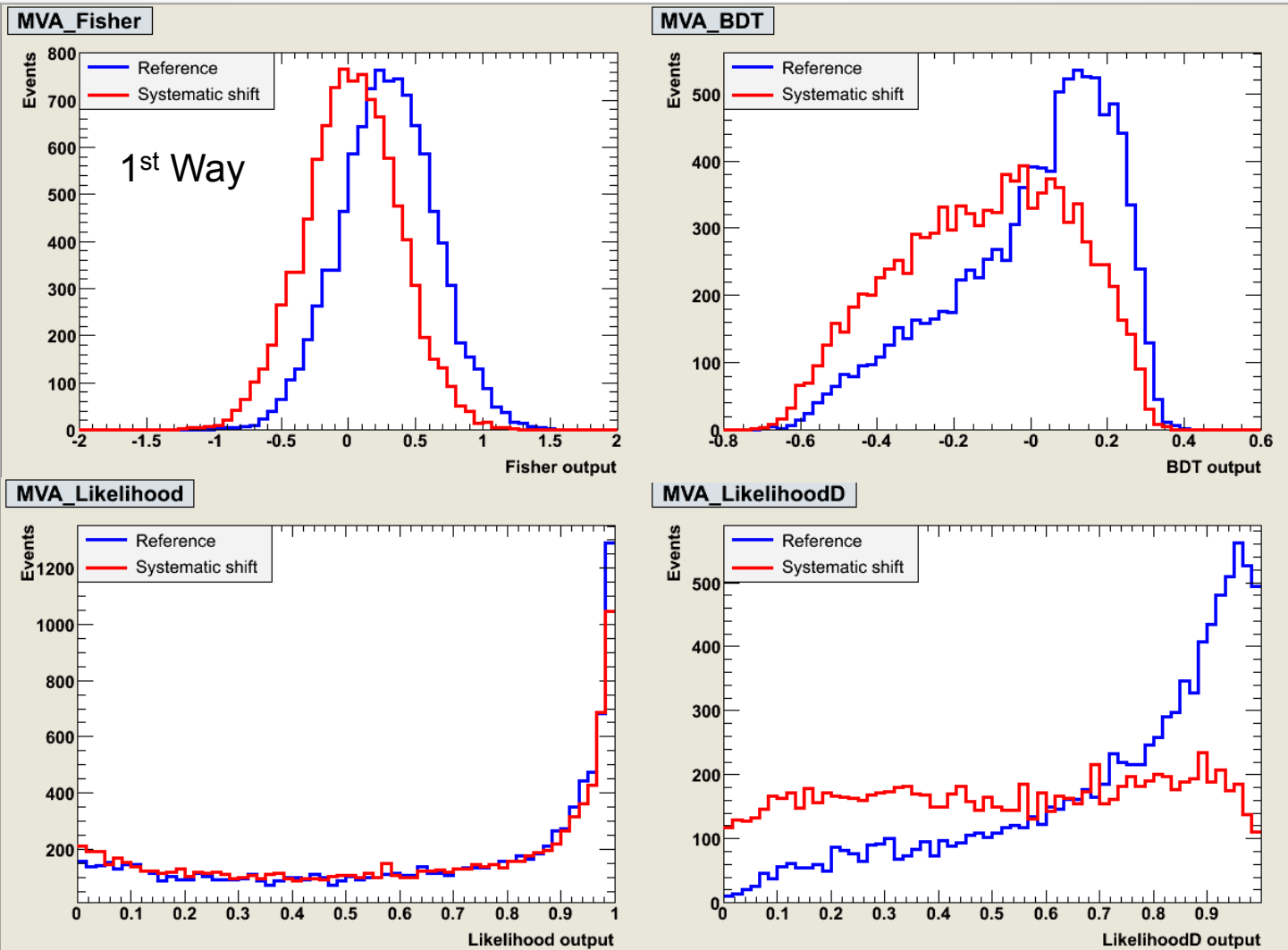


Treatment of Systematic Uncertainties

- 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) !

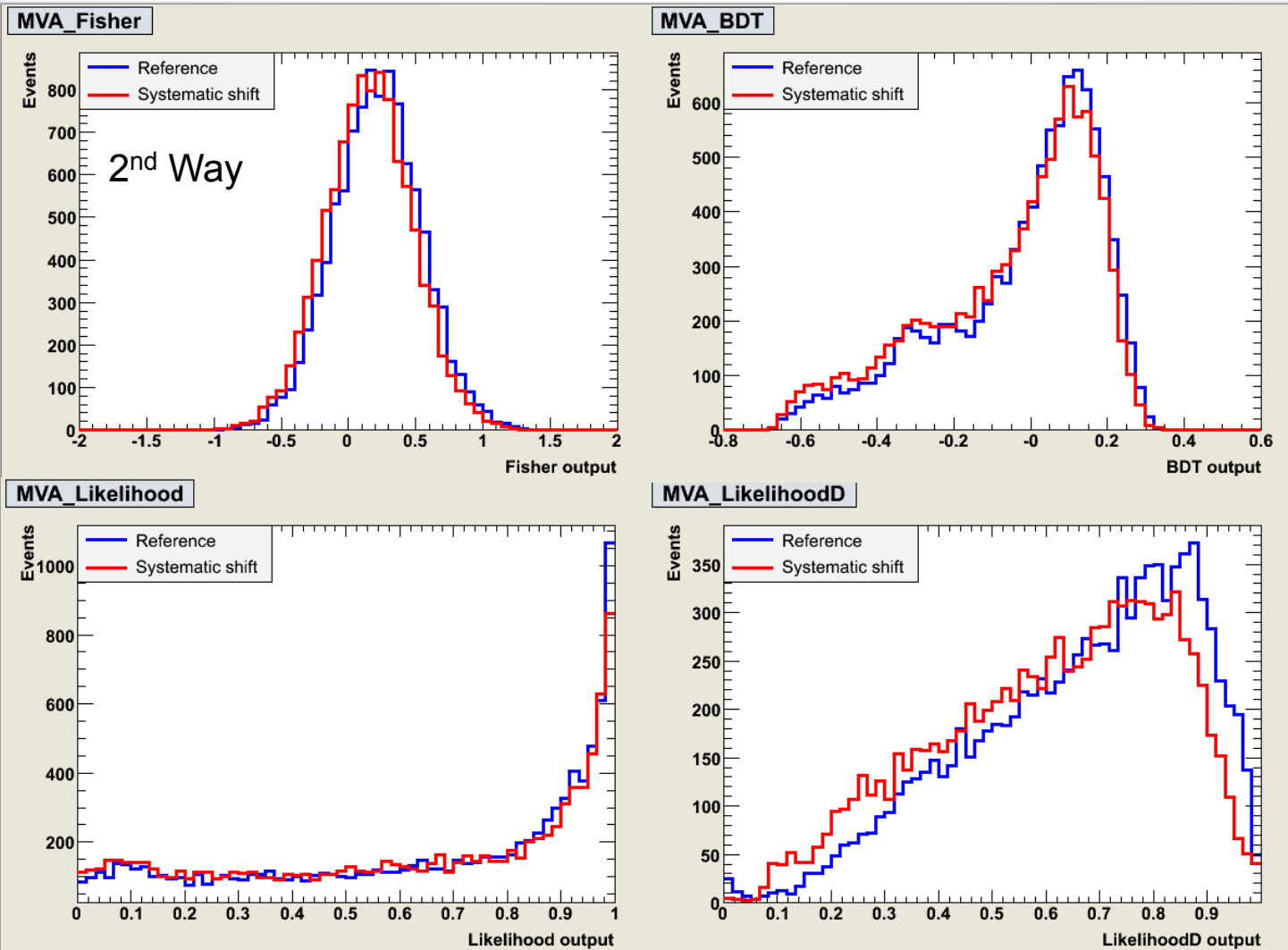
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



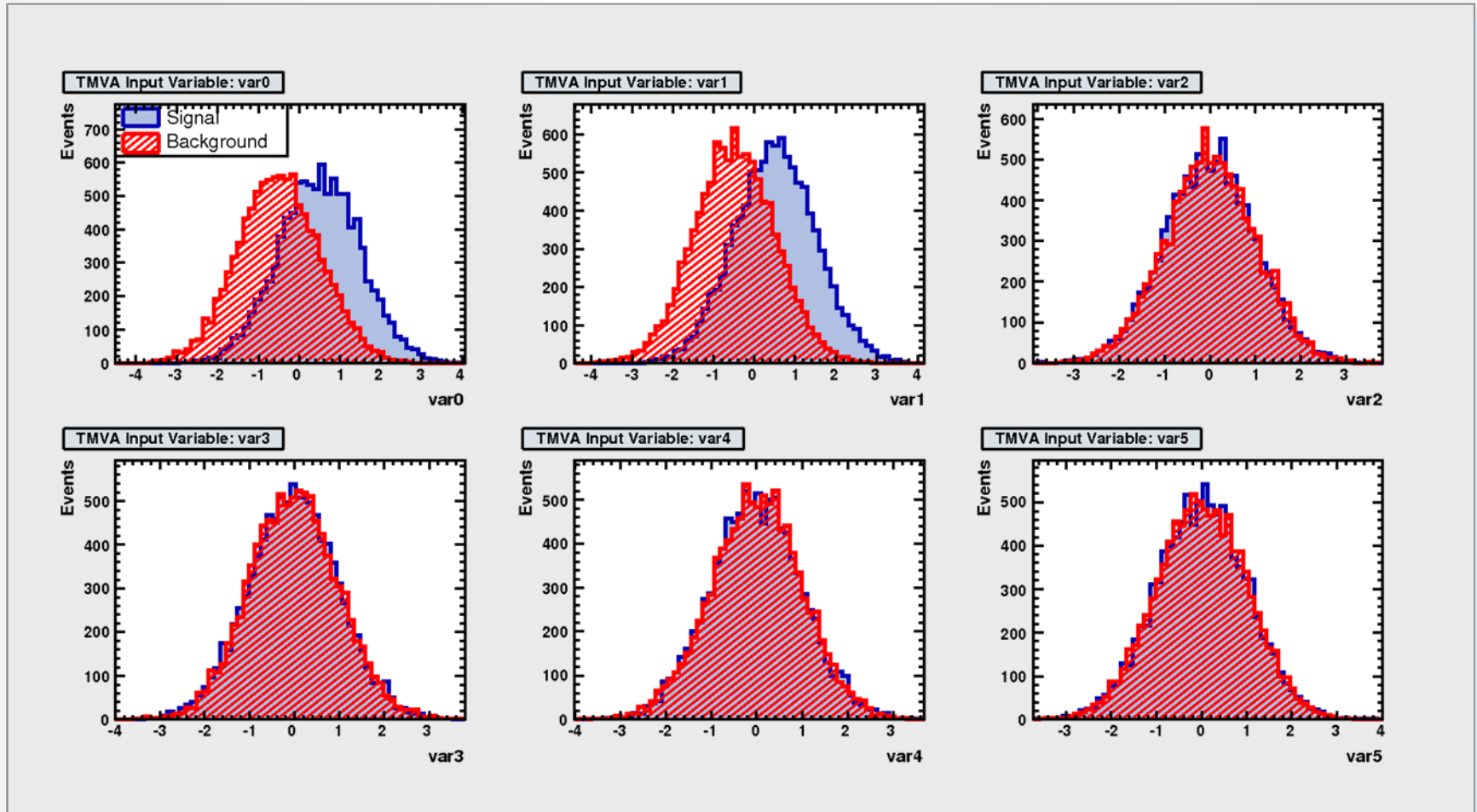
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



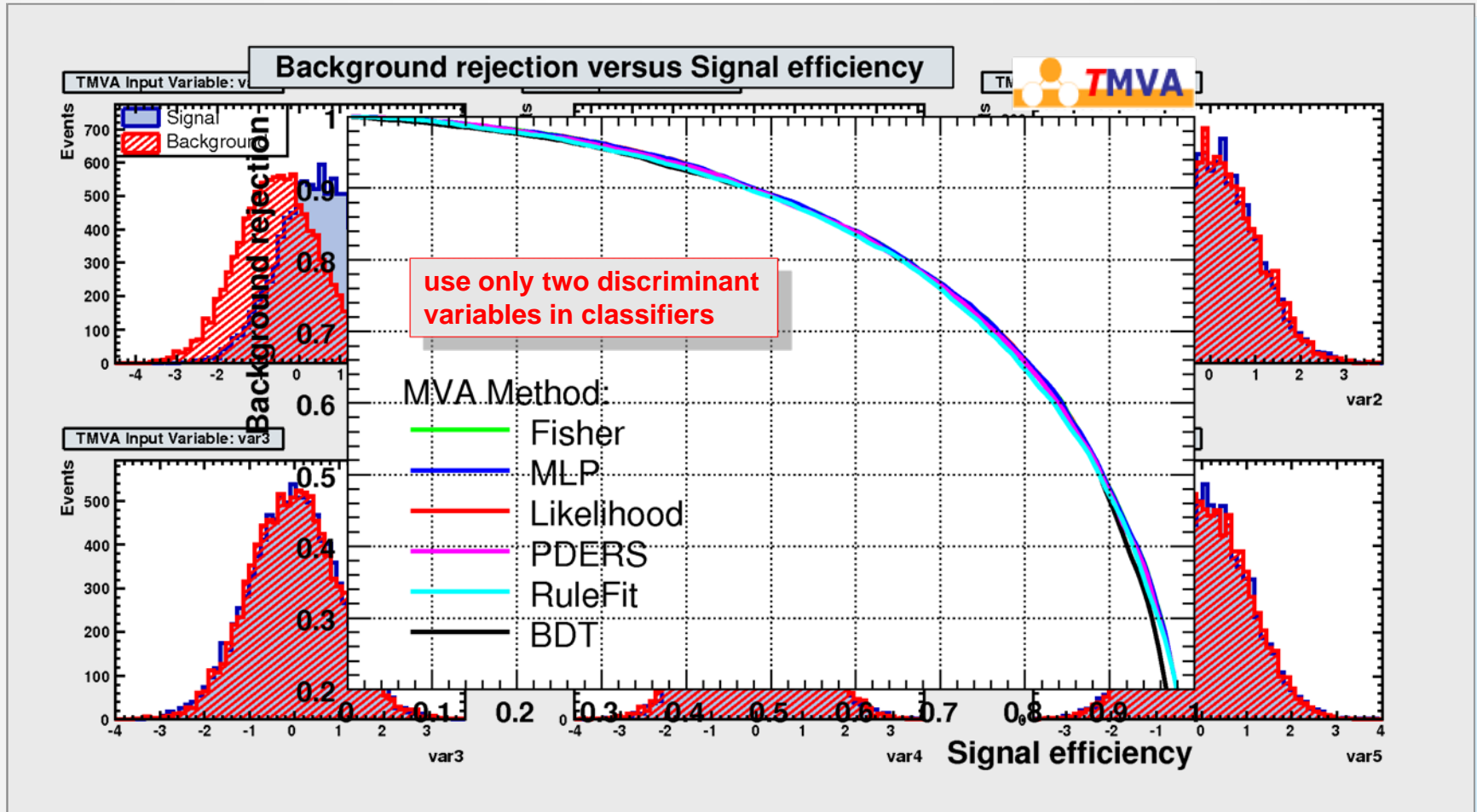
Stability with Respect to Irrelevant Variables

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



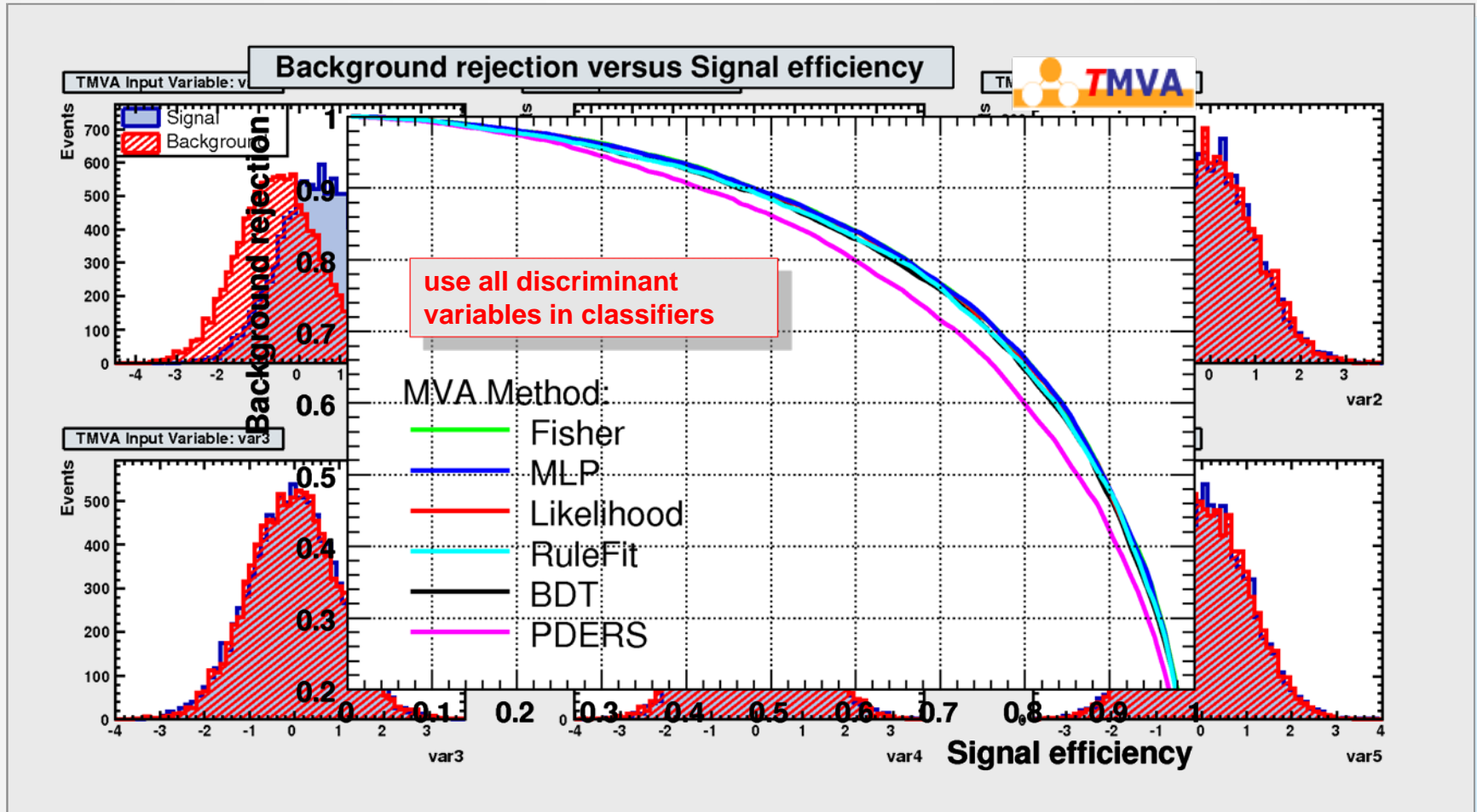
Stability with Respect to Irrelevant Variables

- 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 ?

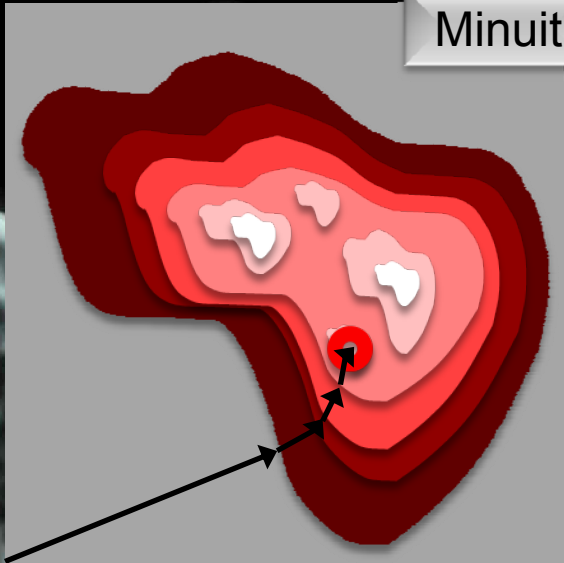


Minimisation

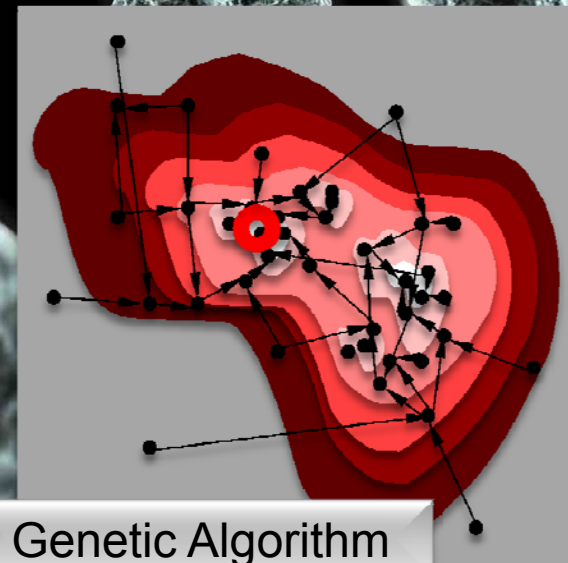
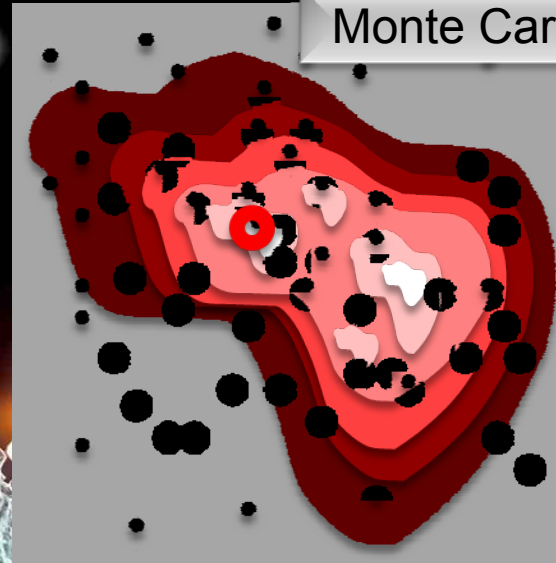
- Robust **global** minimum finder needed at various places in TMVA
- Brute force method: **Monte Carlo Sampling**
 - Sample entire solution space, and chose solution providing minimum estimator
 - Good global minimum finder, but poor accuracy
- Default solution in HEP: **(T)Minuit/Migrad** [How much longer do we need to suffer ?]
 - Gradient-driven search, using variable metric, can use quadratic Newton-type solution
 - Poor global minimum finder, gets quickly stuck in presence of local minima
- Specific **global** optimisers implemented in TMVA:
 - **Genetic Algorithm:** biology-inspired optimisation algorithm
 - **Simulated Annealing:** slow “cooling” of system to avoid “freezing” in local solution
- TMVA allows to chain minimisers
 - For example, one can use MC sampling to detect the vicinity of a global minimum, and then use Minuit to accurately converge to it.

Minimizers

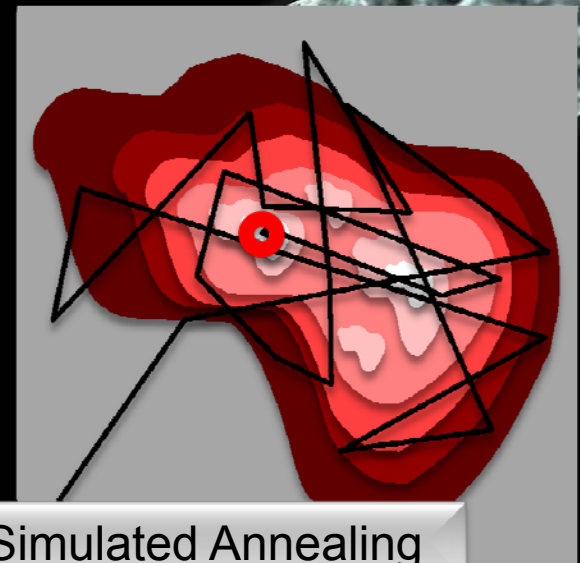
Minuit



Monte Carlo



Genetic Algorithm

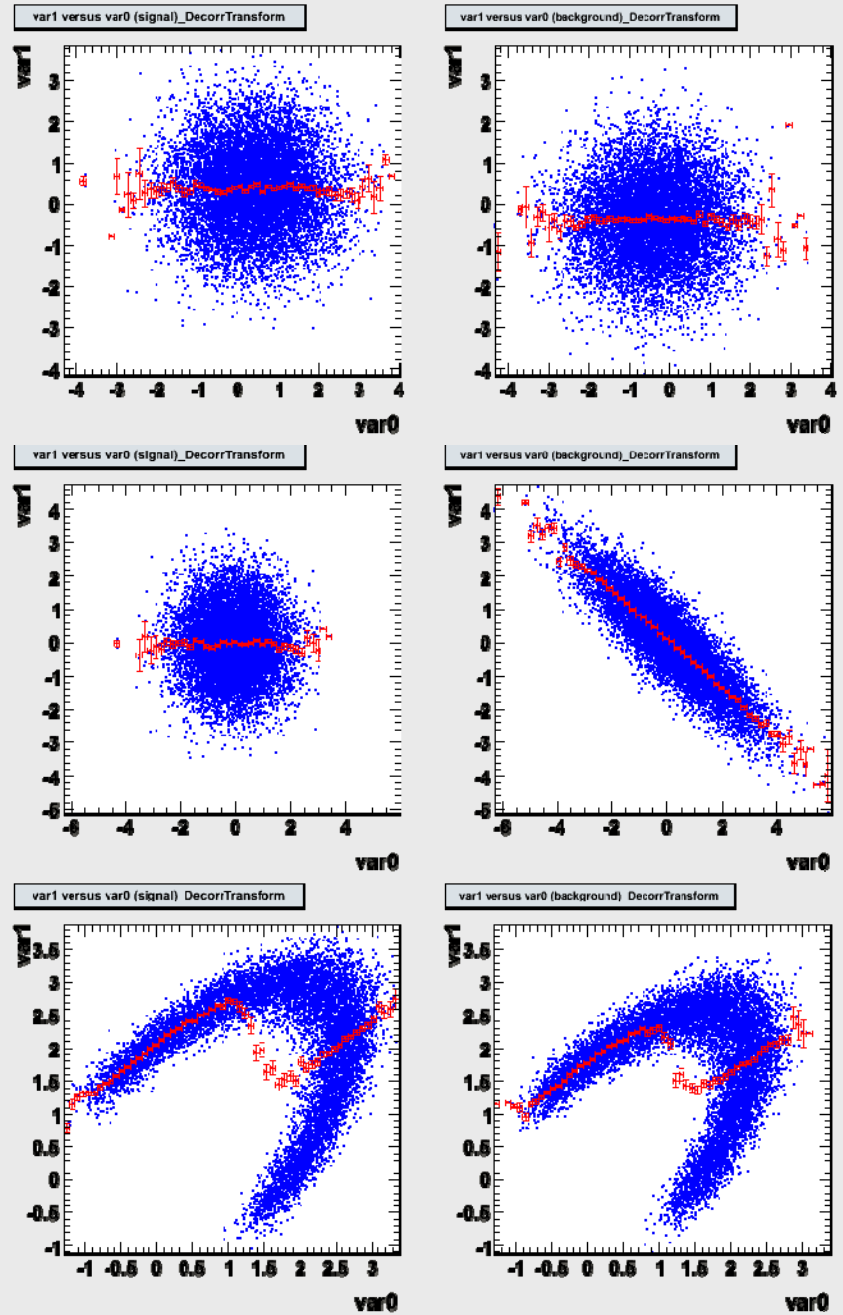


Simulated Annealing

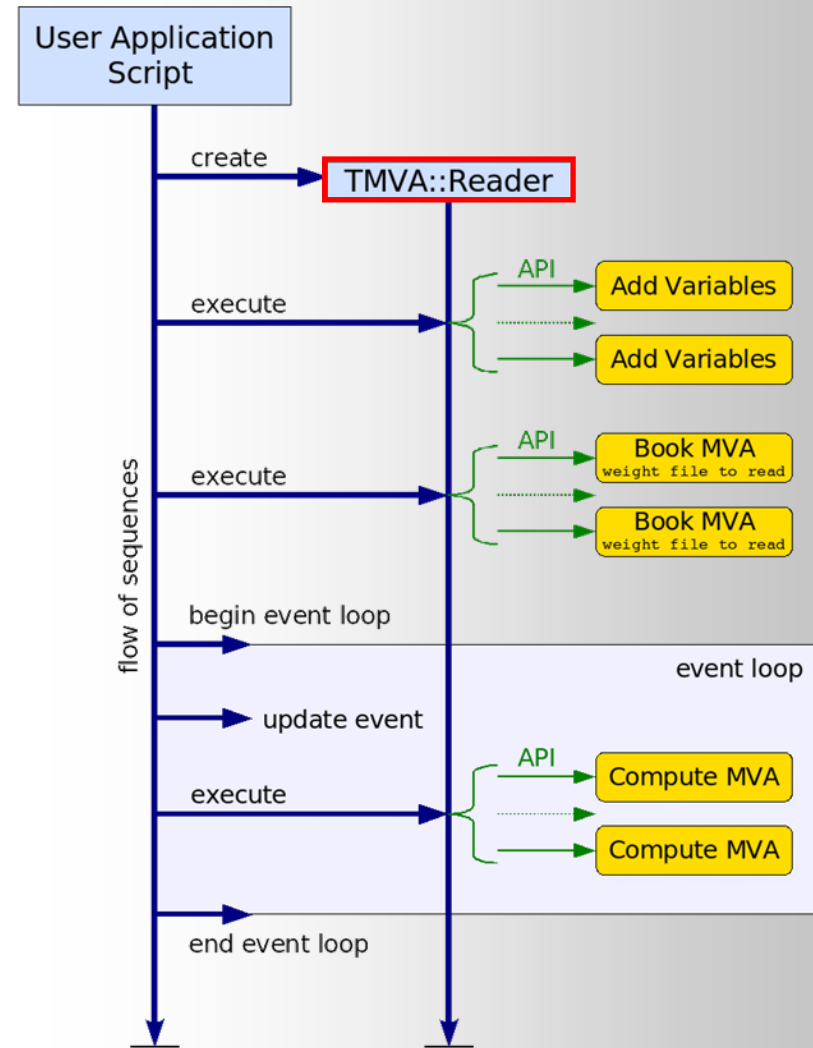
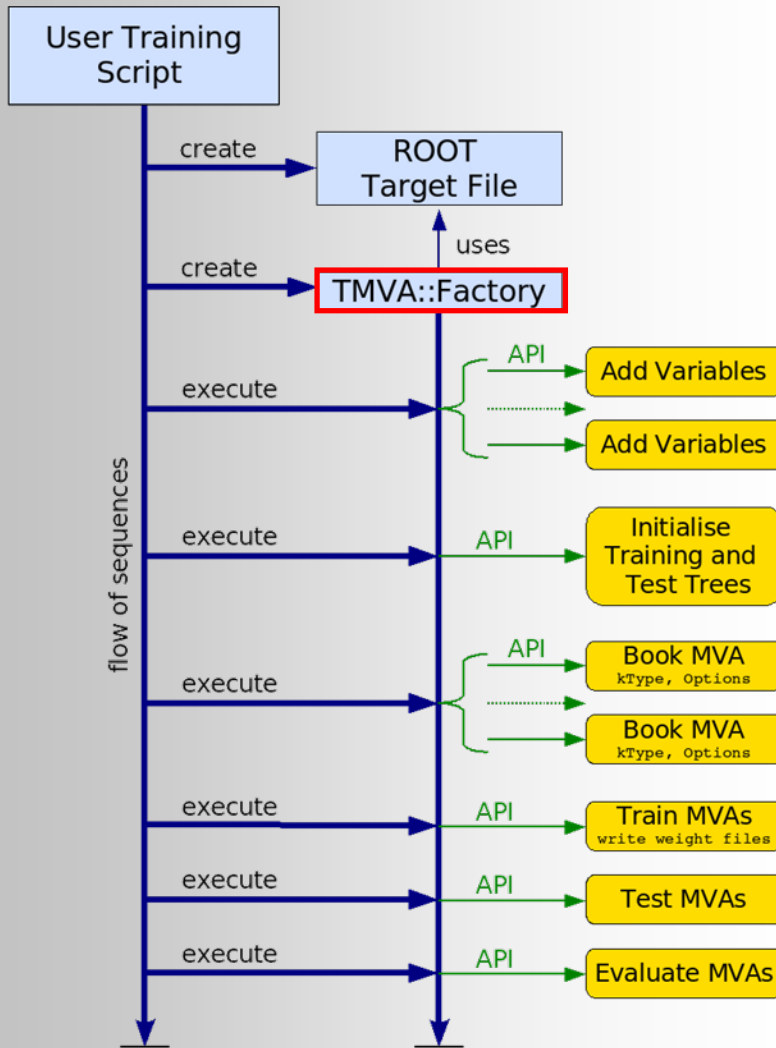
- How does linear decorrelation affect strongly nonlinear cases ?

Original correlations

SQRT decorrelation



Code Flow for *Training* and *Application* Phases



→ [TMVA tutorial](#)

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 (Cal Tech)
 - 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 thank Doug Schouten (S. Fraser U) for improving the BDT, Jan Therhaag (Bonn) for a reimplementaion of LD including regression, and Eckhard v. Toerne (Bonn) for improving the Cuts evaluation. Many thanks to Dominik Dannheim, Alexander Voigt and Tancredi Carli (CERN) for the implementation of the PDEFoam approach. We are grateful to Doug Applegate, Kregg Arms, René Brun and the ROOT team, Zhiyi Liu, Elzbieta Richter-Was, Vincent Tisserand and Alexei Volk for helpful conversations.