Multivariate Data Analysis with *T*MVA

Andreas Hoecker[(*)] (CERN)

Seminar at Edinburgh, Scotland, Dec 5, 2008

[(*)] On behalf of the present core developer team: A. Hoecker, P. Speckmayer, J. Stelzer, 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).

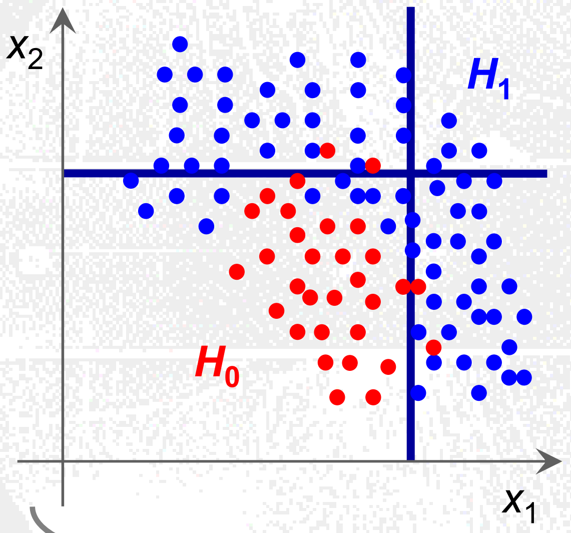
See acknowledgments on page 43

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

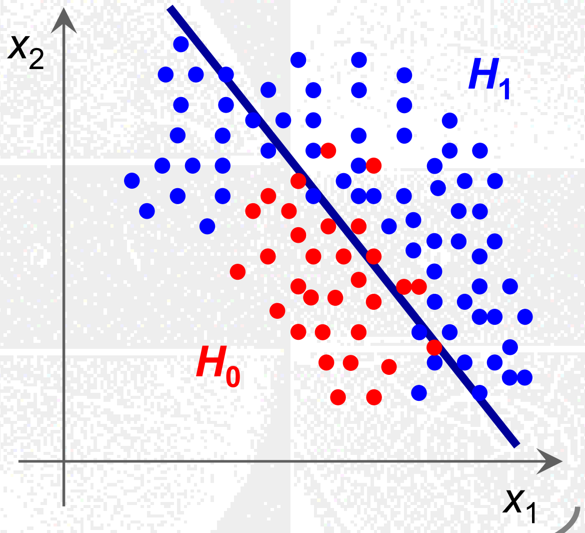
Event Classification

- 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 ?

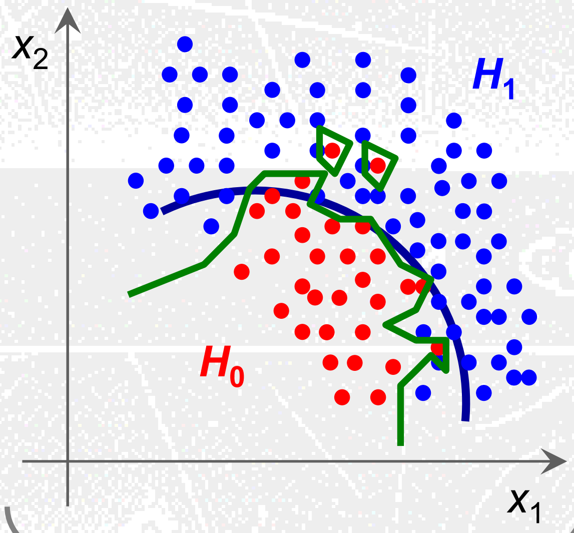
Rectangular cuts?



A linear boundary?



A nonlinear one?

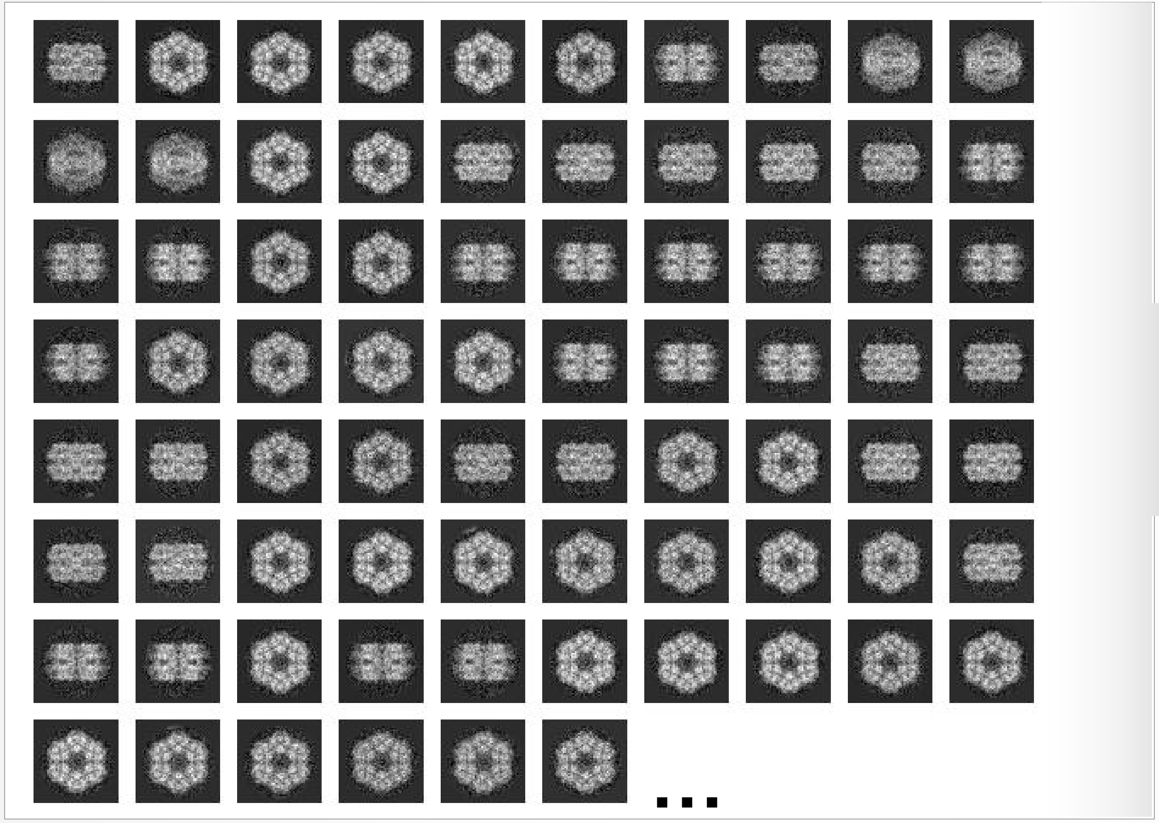


Low variance (stable), high bias methods

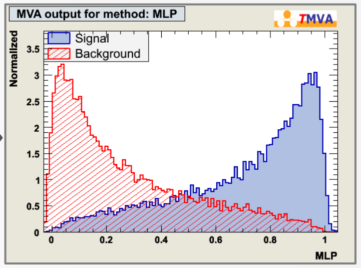
High variance, small bias methods

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*



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



This can be generalised to multiple classes and to regression

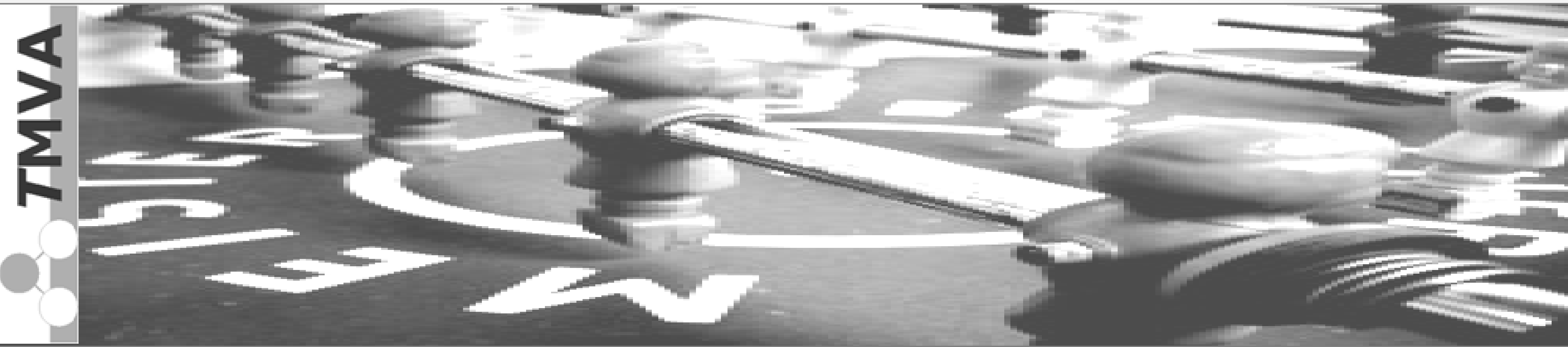
*T*MVA

Outline of this presentation:

- **The *T*MVA project**
- **Overview of available classifiers and processing steps**
- **Evaluation tools**
- **(Toy) examples**

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 TMulitLayerPercetron 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



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

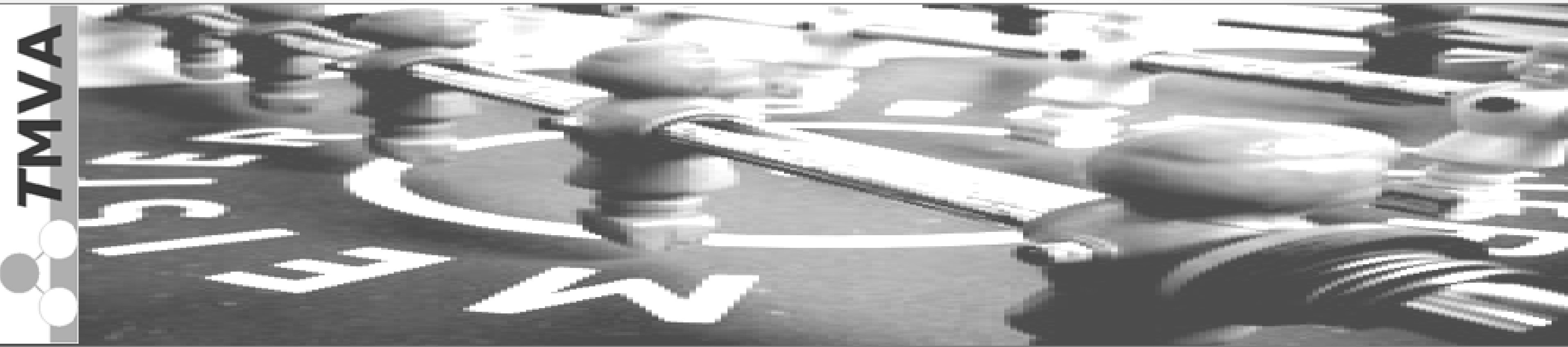
- Active project → fast response time on feature requests
  - Currently 4 core developers, and 25 contributors
  - >3500 downloads since March 2006 (not accounting CVS checkouts and ROOT users)

- Written in C++, relying on core ROOT functionality

- Integrated and distributed with ROOT since ROOT v5.11/03

Limitations of *T*MVA

- Development started beginning of 2006 – a mature but **not** a final package
- Known limitations / missing features
  - Performs classification only, and only in binary mode: *signal* versus *background*
  - Supervised learning only (no unsupervised “bump hunting”)
  - Relatively stiff design – not easy to mix methods, not easy to setup categories
  - Cross-validation not yet generalised for use by all classifiers
  - Optimisation of classifier architectures still requires tuning “by hand”
- Work ongoing in most of these areas → see outlook to *T*MVA 4



TMVA Content

- 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
  - RuleFit
  - Support Vector Machine

- Currently implemented data preprocessing stages:
  - Decorrelation
  - Principal Value Decomposition
  - Transformation to uniform and Gaussian distributions

Data Preprocessing

Data Preprocessing: Decorrelation

- Commonly realised for all methods in *T*MVA

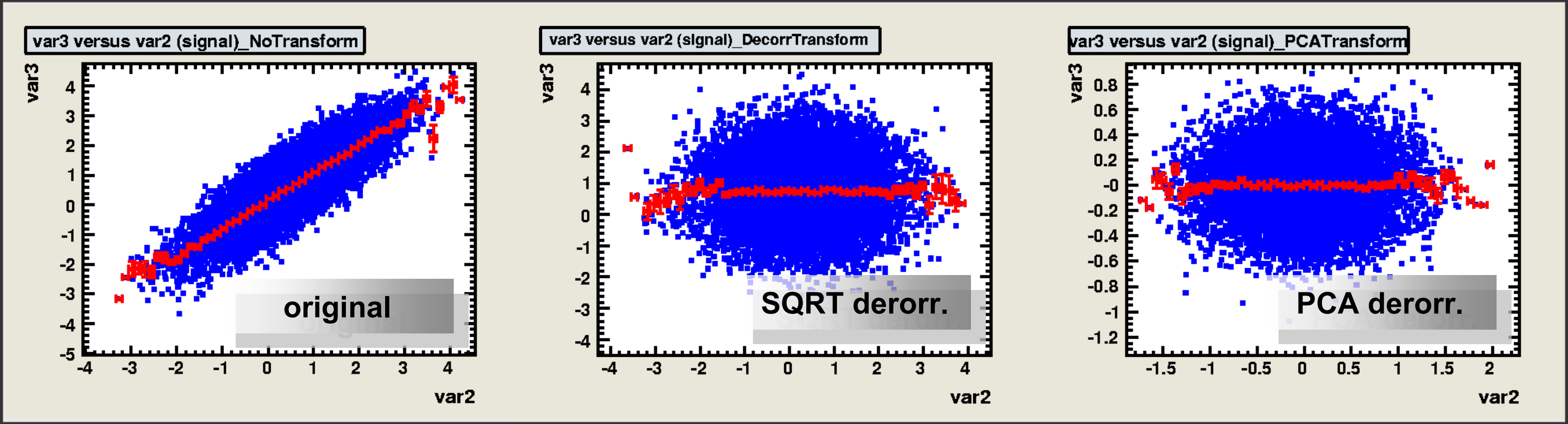
- Removal of linear correlations by rotating input variables
  - Cholesky decomposition: determine *square-root* C' of covariance matrix C, *i.e.*, $C = C'C'$
  - Transform original (x) into decorrelated variable space (x') by: $x' = C'^{-1}x$

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

$$\underbrace{x_k^{\mathrm{PC}}}_{\text{PC of variable } k}\left(i_{\mathrm{event}}\right) = \sum_{v\in\{\mathrm{variables}\}} \left[x_v\left(i_{\mathrm{event}}\right) - \underbrace{\bar{x}_v}_{\text{Sample means}}\right]\cdot \underbrace{v_v^{(k)}}_{\text{Eigenvector}} \quad ,\ \forall k \in \{\mathrm{variables}\}$$

  - Matrix of eigenvectors V obeys relation: $C \cdot V = D \cdot V$ thus PCA eliminates correlations

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:

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

Rarity transform of variable k | Measured value | PDF of variable k

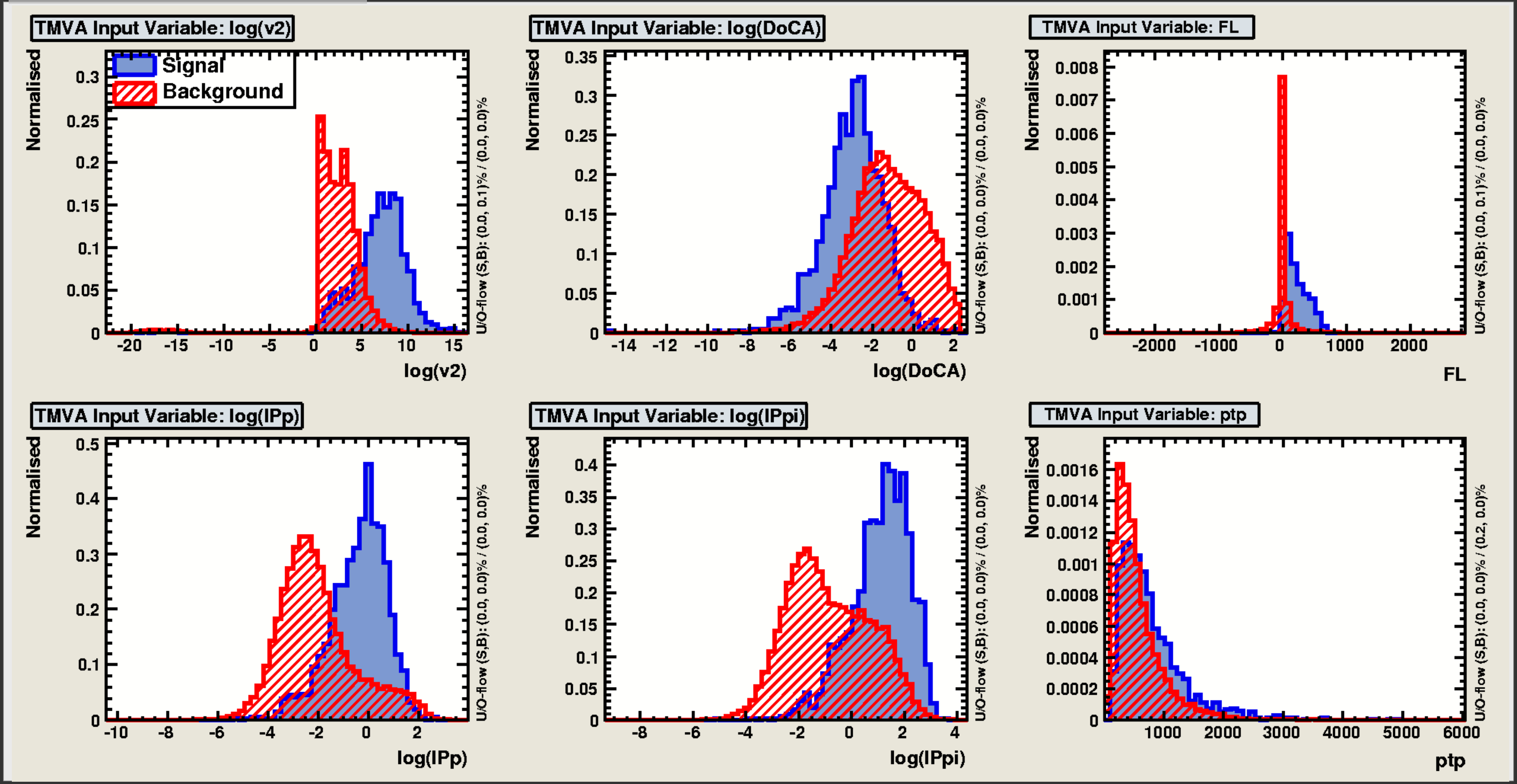
The integral can be solved in an unbinned way by event counting,
or by creating non-parametric PDFs (see later for likelihood section)

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

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

"Gaussian-isation"

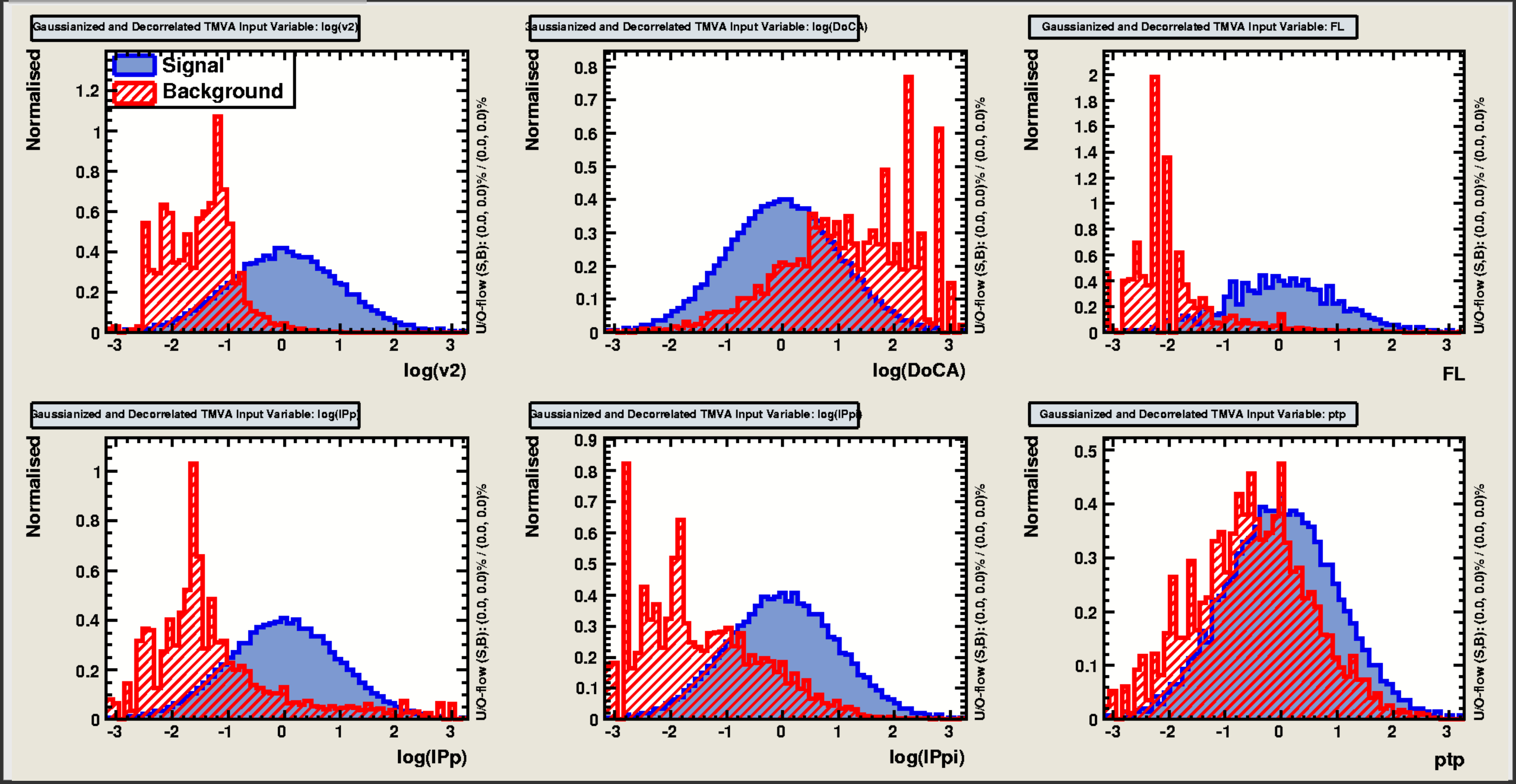
Original



We cannot simultaneously "gaussianise" both signal and background !

"Gaussian-isation"

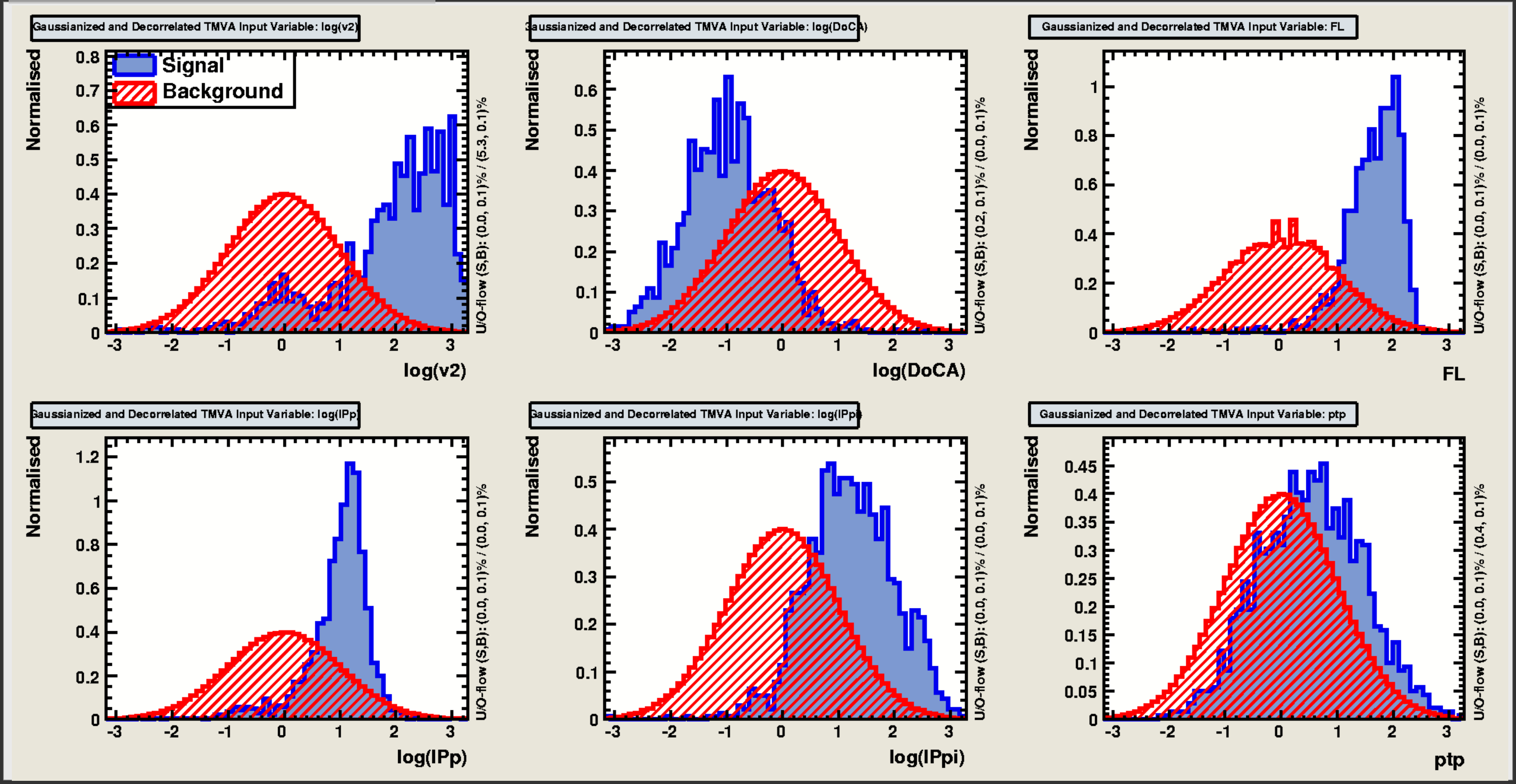
Signal - Gaussianised



We cannot simultaneously "gaussianise" both signal and background !

"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
  - **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:

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

$$\Downarrow$$

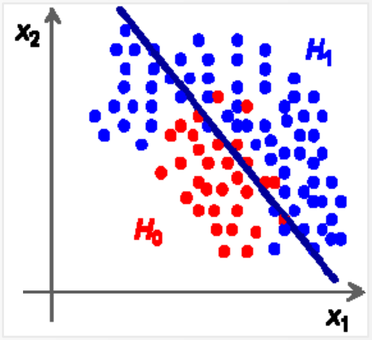
$$y_L^{\text{trans}}(i_{\text{event}}) = \frac{\prod_{k\in\{\text{variables}\}} p_k^S\left(\hat{T}^S x_k(i_{\text{event}})\right)}{\prod_{k\in\{\text{variables}\}} p_k^S\left(\hat{T}^S x_k(i_{\text{event}})\right) + \prod_{k\in\{\text{variables}\}} p_k^B\left(\hat{T}^B x_k(i_{\text{event}})\right)}$$

The Classifiers

Rectangular Cut Optimisation

- Simplest method: cut in rectangular variable volume

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



- 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

d i g r e s s i o n

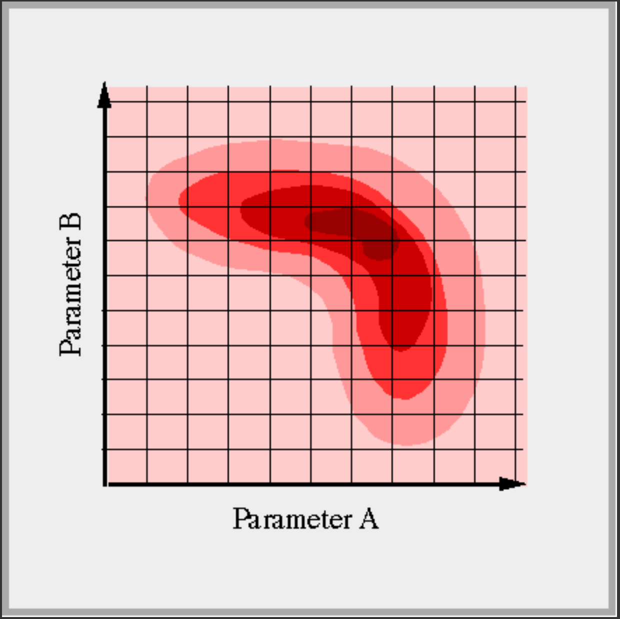
- Minimisation techniques in *T*MVA
- Binary tree sorting

Minimisation

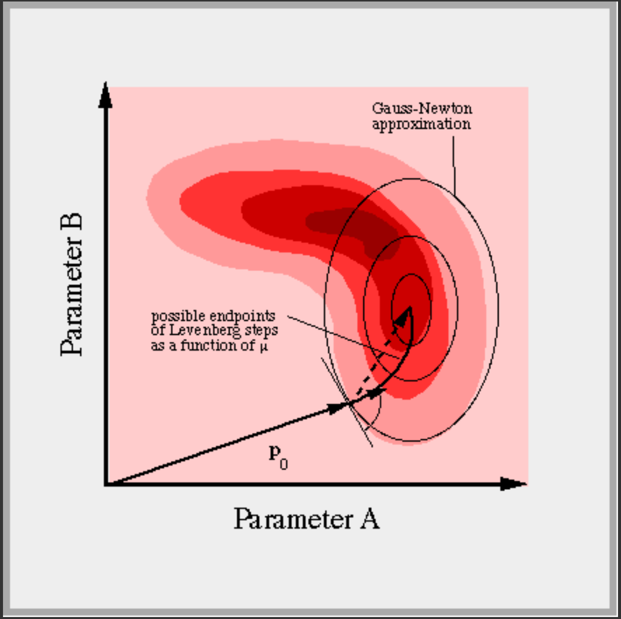
- Robust **global** minimum finder needed at various places in TMVA
- Brute force method: Monte Carlo Sampling
  - Sample entire solution space, and chose solution providing minimum estimator
  - Good global minimum finder, but poor accuracy
- Default solution in HEP: (T)Minuit/Migrad [How much longer 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.

Minimisation Techniques

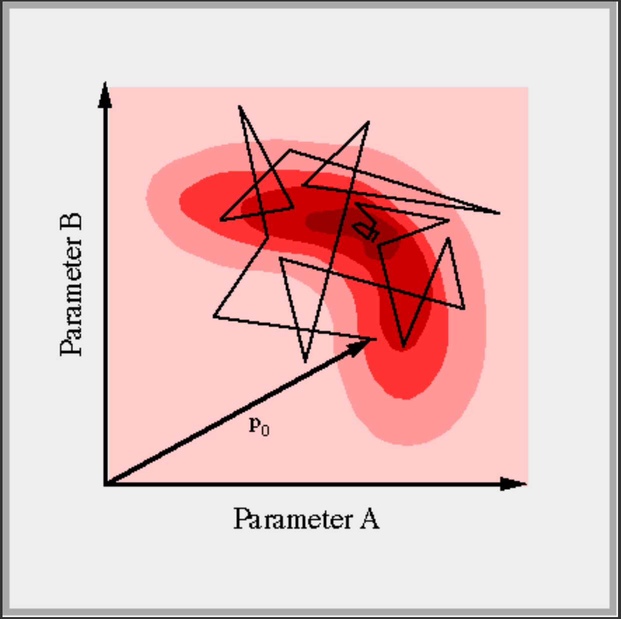
Grid search



Quadratic Newton


Gauss-Newton approximation
possible endpoints of Levenberg steps as a function of μ
p0
Parameter B
Parameter A

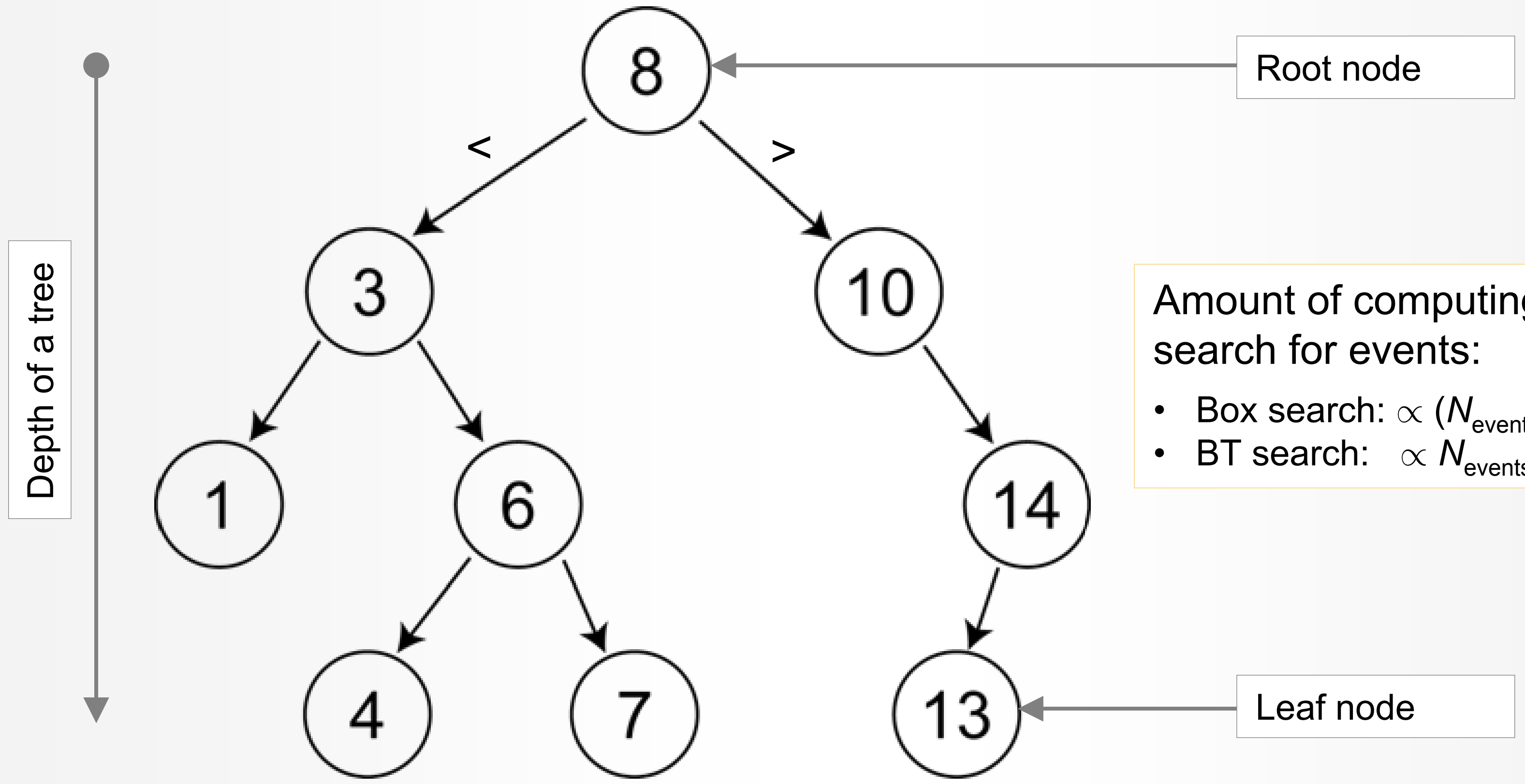
Simulated Annealing



Source: http://www-esd.lbl.gov/iTOUGH2/Minimization/minalg.html

Binary Trees

- Tree data structure in which each node has at most two children
  - Typically the child nodes are called left and right
  - Binary trees are used in TMVA to implement **binary search trees** and **decision trees**



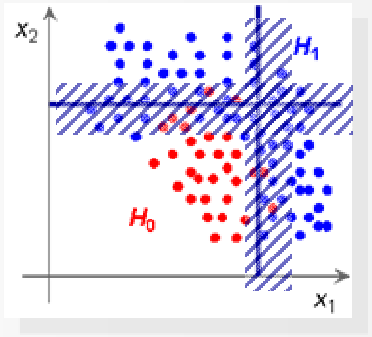
Amount of computing time to search for events:

- Box search: $\propto (N_{\text{events}})^{N\text{var}}$
- BT search: $\propto N_{\text{events}} \cdot N_{\text{var}} \ln_2(N_{\text{events}})$

)

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

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

Species: signal, background types

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

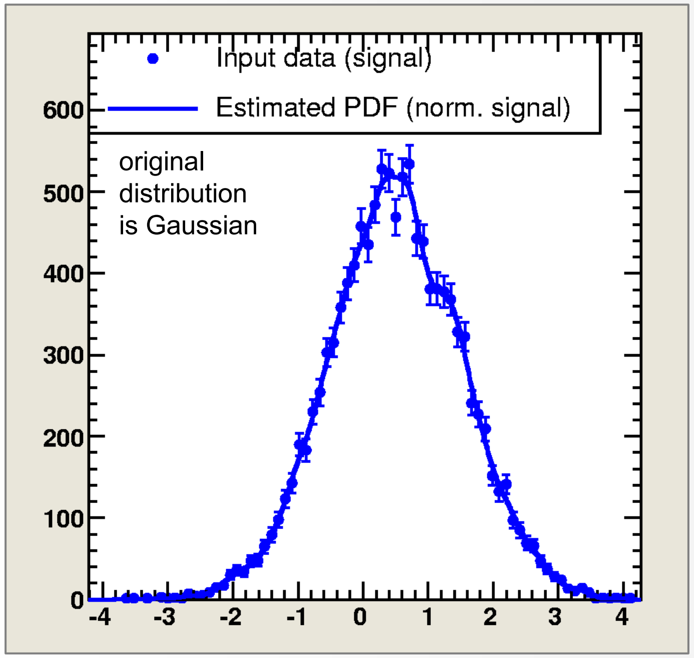
PDE Approach: Estimating PDF Kernels

- Technical challenge: how to estimate the PDF shapes

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

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

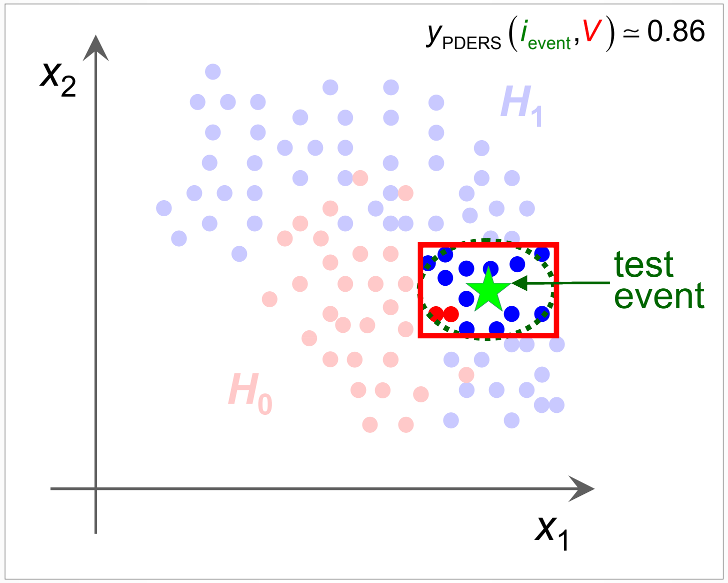
  - Binned shape interpolation using spline functions and adaptive smoothing
  - Unbinned adaptive kernel density estimation (KDE) with Gaussian smearing
  - *T*MVA performs automatic validation of goodness-of-fit



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)

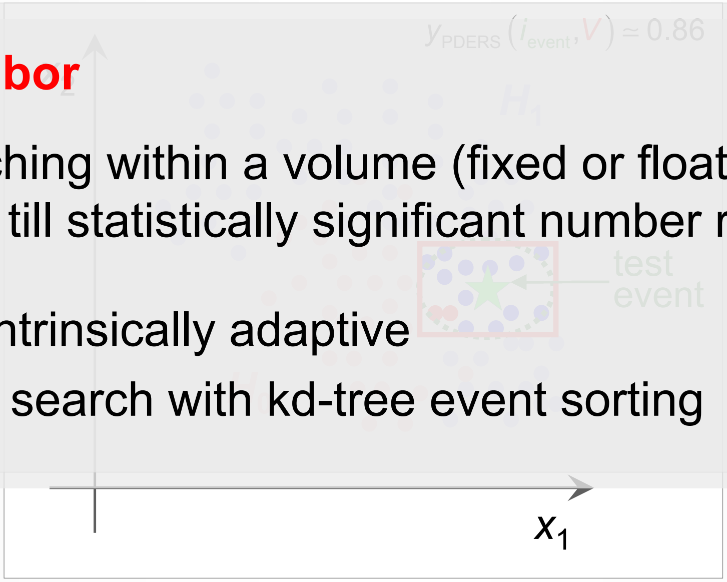


- Improve y_{PDERS} estimate within V by using various N_{var}-D kernel estimators
- Enhance speed of event counting in volume by binary tree search

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

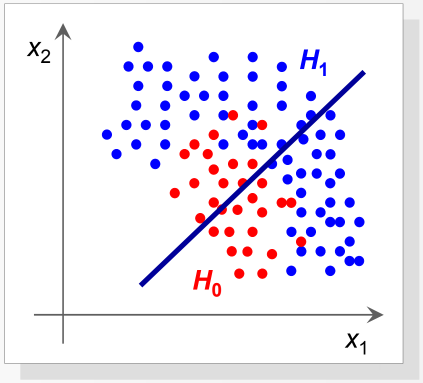
Better than searching within a volume (fixed or floating), count adjacent reference events till statistically significant number reached

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

- Improve y_{PDERS} 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 same class in close vicinity to each other



- Classifier response couldn't be simpler:

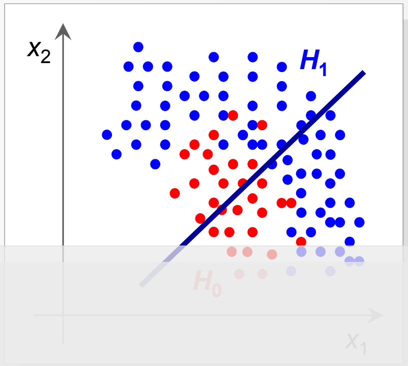
"Bias" "Fisher coefficients"

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

- 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

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 same class in close vicinity to each other



- Classifier response couldn’t be simpler:

“Bias” “Fisher coefficients”

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

Function discriminant analysis (FDA)

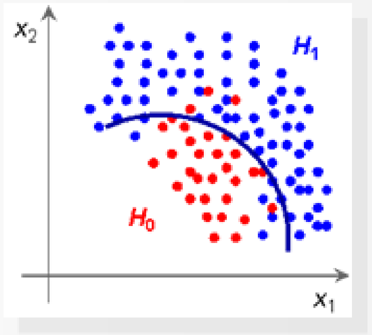
Fit any user-defined function of input variables requiring that signal events return →1 and background →0

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

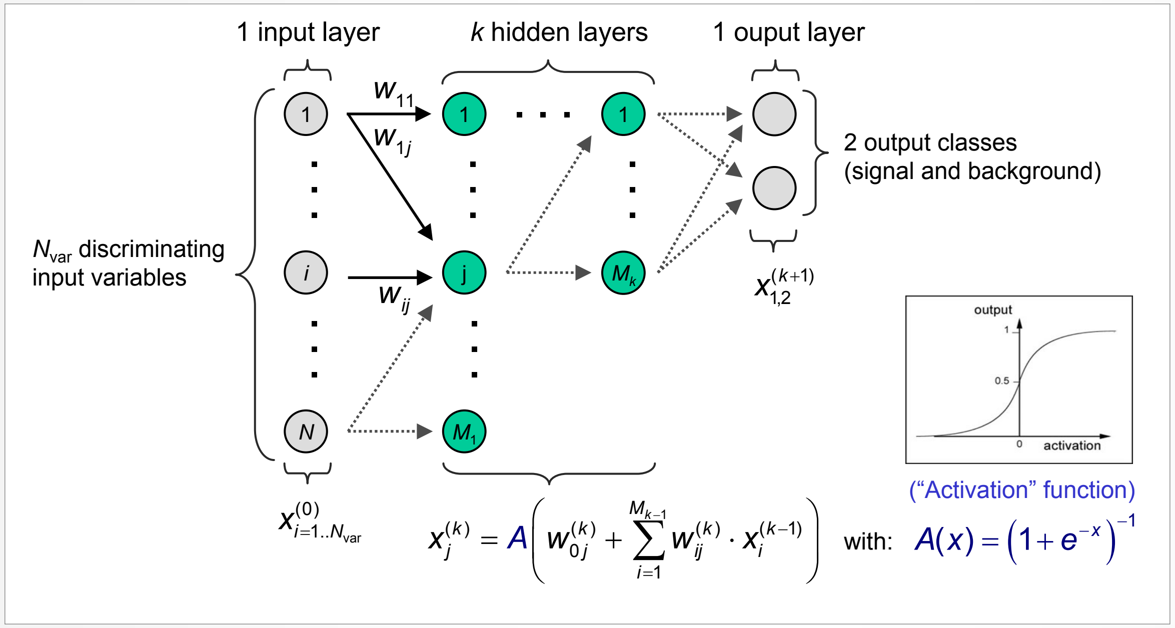
  - Compute Fisher coefficients from signal and background covariance matrices
- Fisher requires distinct sample means between signal and background
- Optimal classifier (Bayes limit) for linearly correlated Gaussian-distributed variables

Nonlinear Analysis: Artificial Neural Networks

- Achieve nonlinear classifier response by "activating" output nodes using nonlinear weights



Feed-forward Multilayer Perceptron



("Activation" function)

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

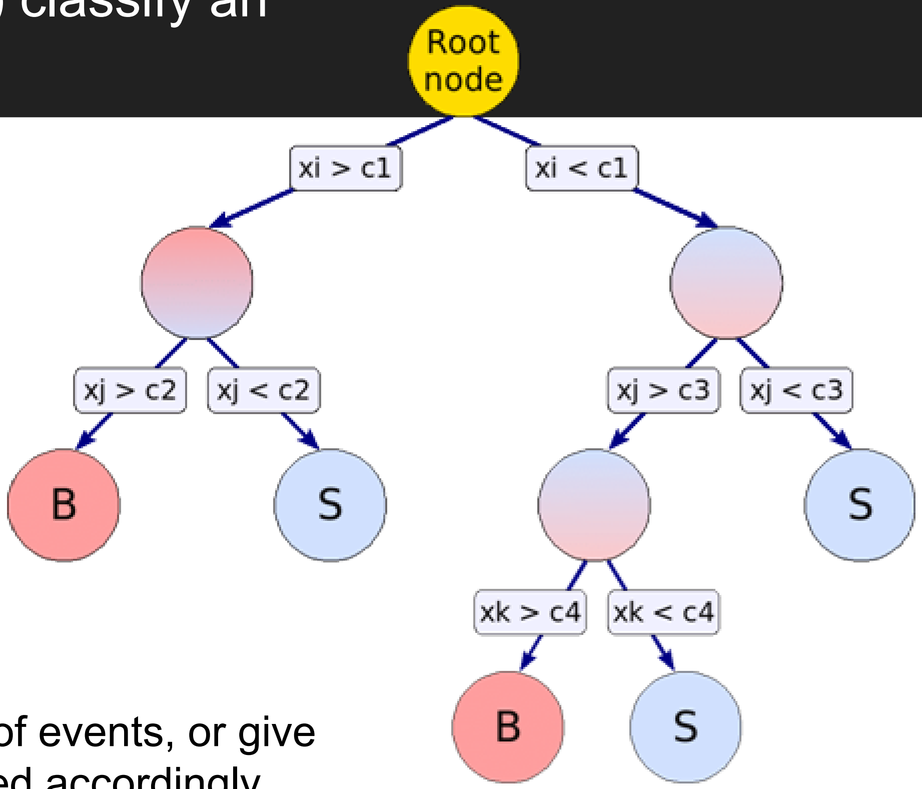
with: $A(x) = \left(1 + e^{-x}\right)^{-1}$

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

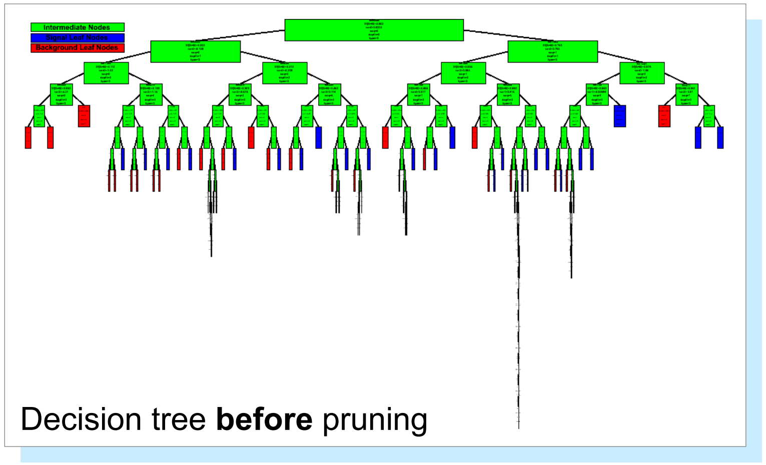
- Sequential application of cuts splits the data into nodes, where the final nodes (leafs) classify an event as signal or background



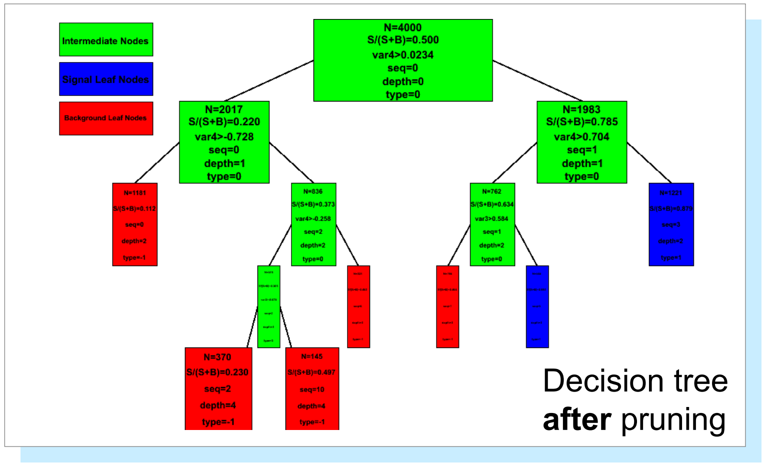
- Growing a decision tree:
  - Start with Root node
  - Split training sample according to cut on best variable at this node
  - Splitting criterion: e.g., maximum "Gini-index": purity $\times$ (1– purity)
  - Continue splitting until min. number of events or max. purity reached
  - Classify leaf node according to majority of events, or give weight; unknown test events are classified accordingly

- Why not multiple branches (splits) per node ?
  - Fragments data too quickly; also: multiple splits per node = series of binary node splits

Decision Trees



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 ($\rightarrow$ 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 $\rightarrow$ 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

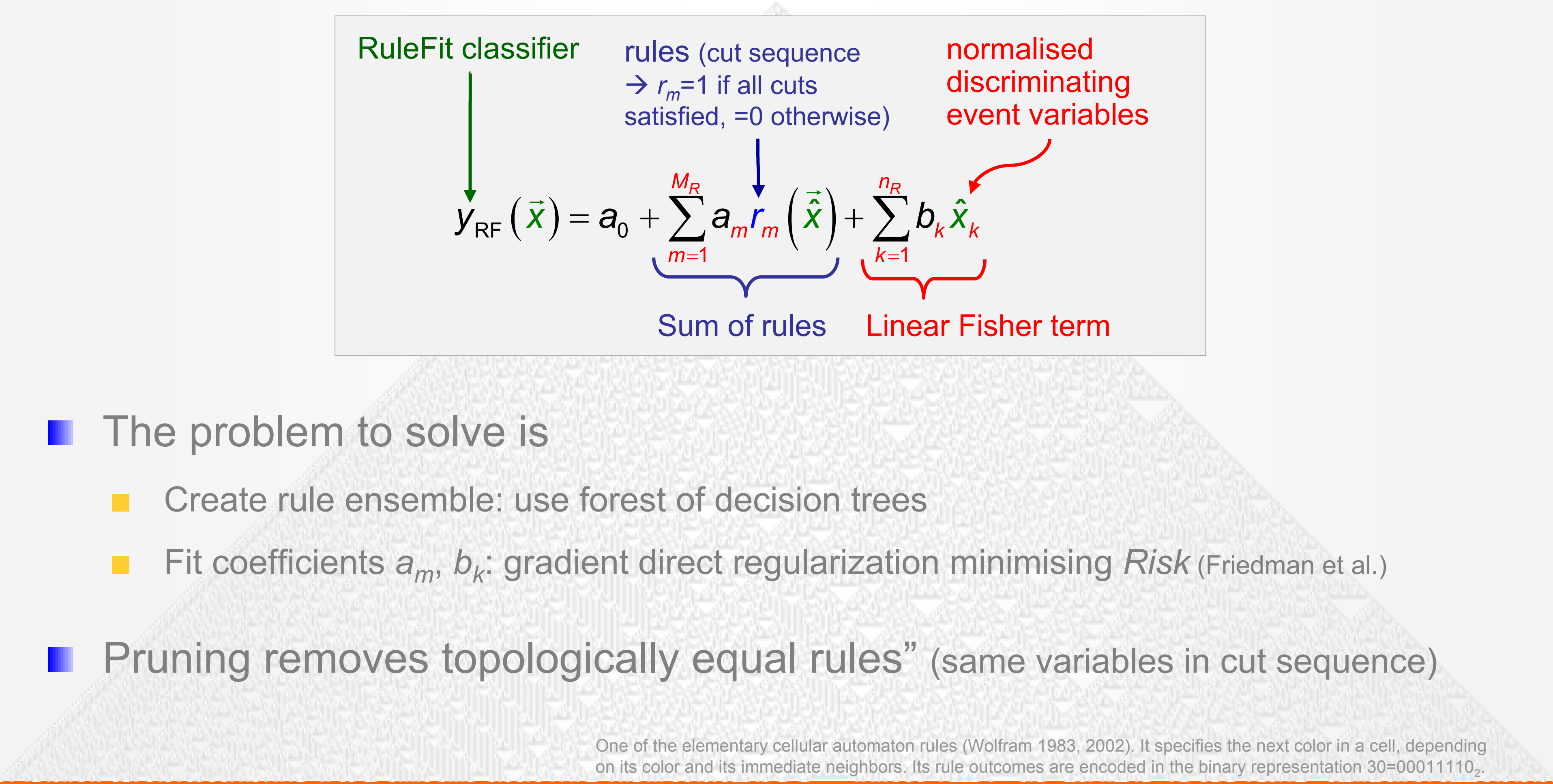
RuleFit classifier

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

normalised discriminating event variables

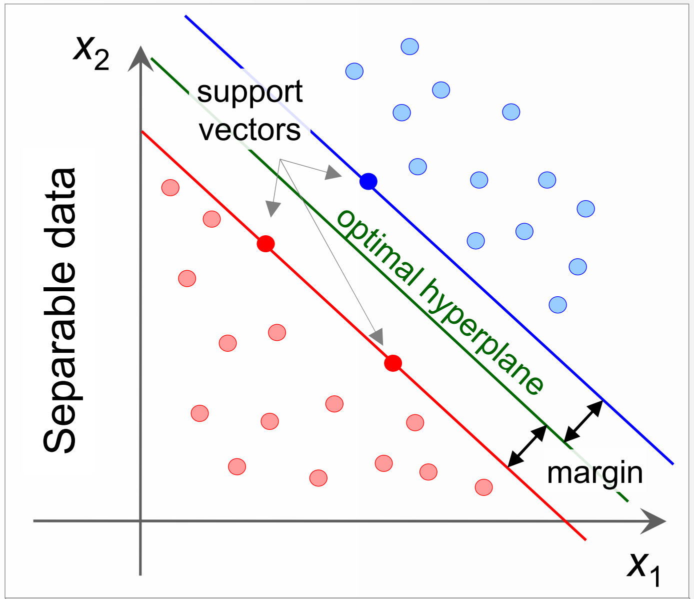
$$y_{\mathrm{RF}}\left(\vec{x}\right) = a_0 + \underbrace{\sum_{m=1}^{M_R} a_m r_m\left(\hat{\vec{x}}\right)}_{\text{Sum of rules}} + \underbrace{\sum_{k=1}^{n_R} b_k \hat{x}_k}_{\text{Linear Fisher term}}$$

- 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)

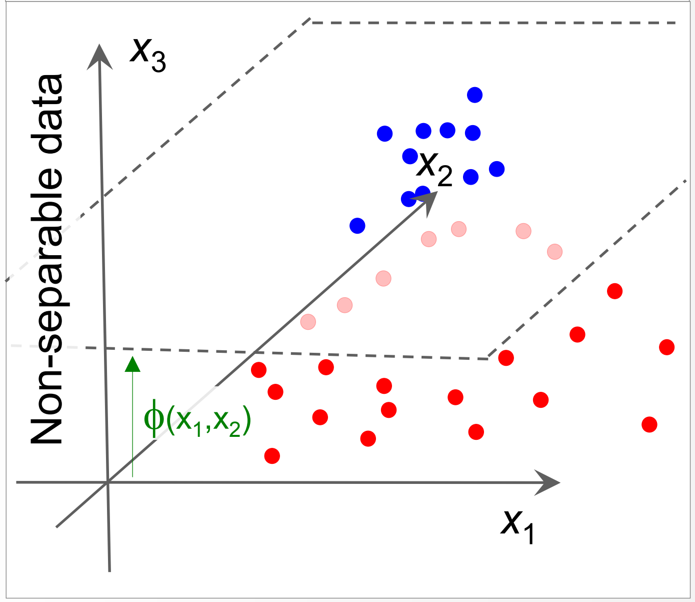

One of the elementary cellular automaton rules (Wolfram 1983, 2002). It specifies the next color in a cell, depending on its color and its immediate neighbors. Its rule outcomes are encoded in the binary representation $30=00011110_2$.

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

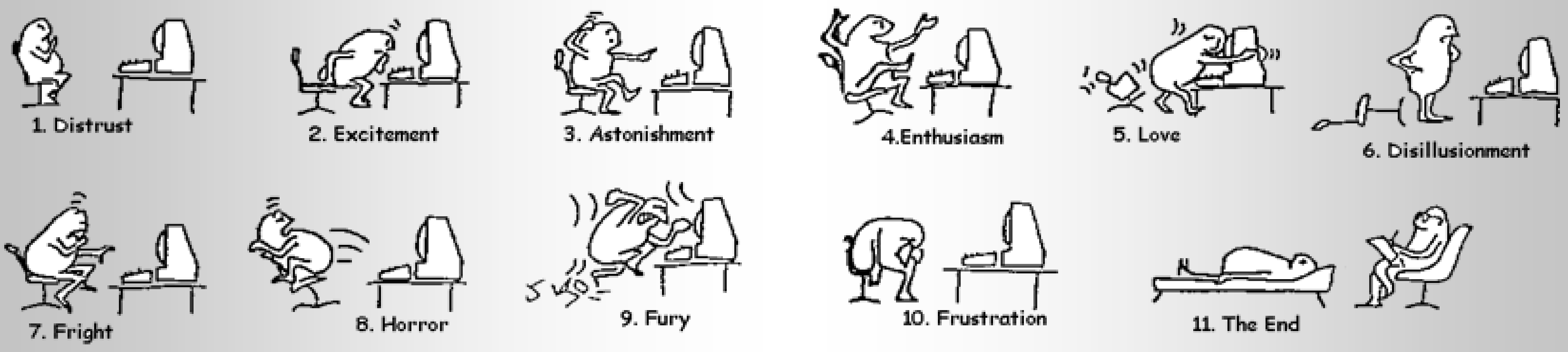


Using *T*MVA

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

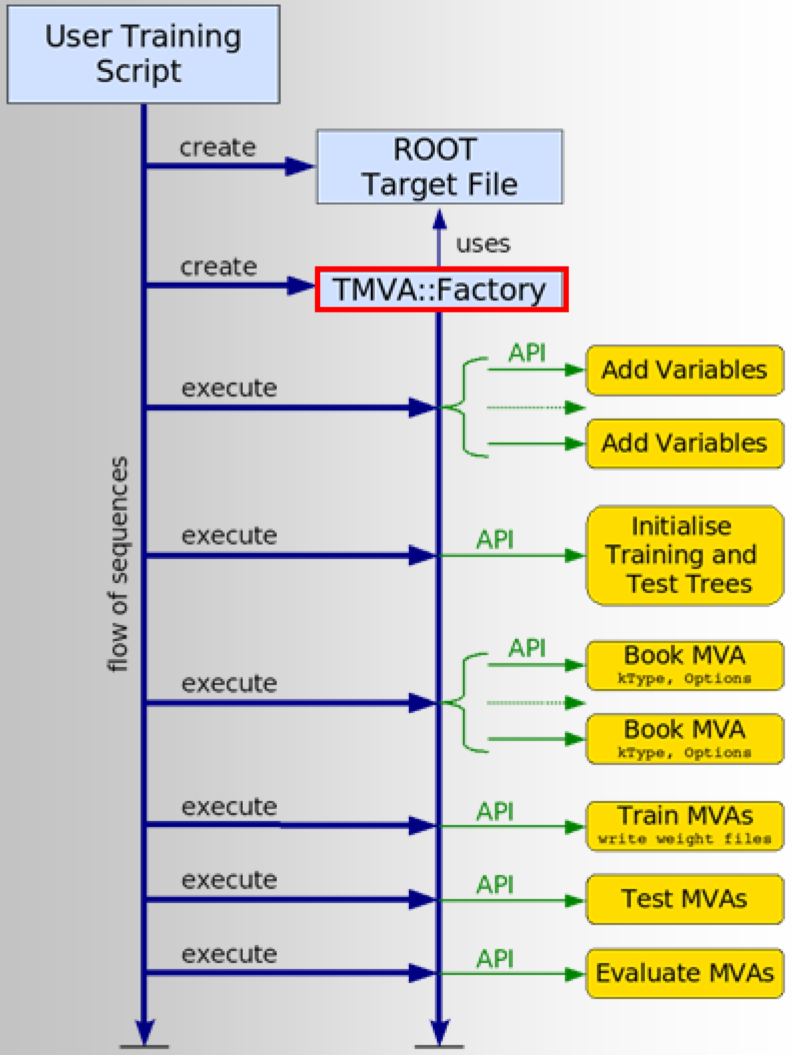
1. *Training phase*: training, testing and evaluation of classifiers using data samples with known signal and background composition
2. *Application phase*: using selected trained classifiers to classify unknown data samples

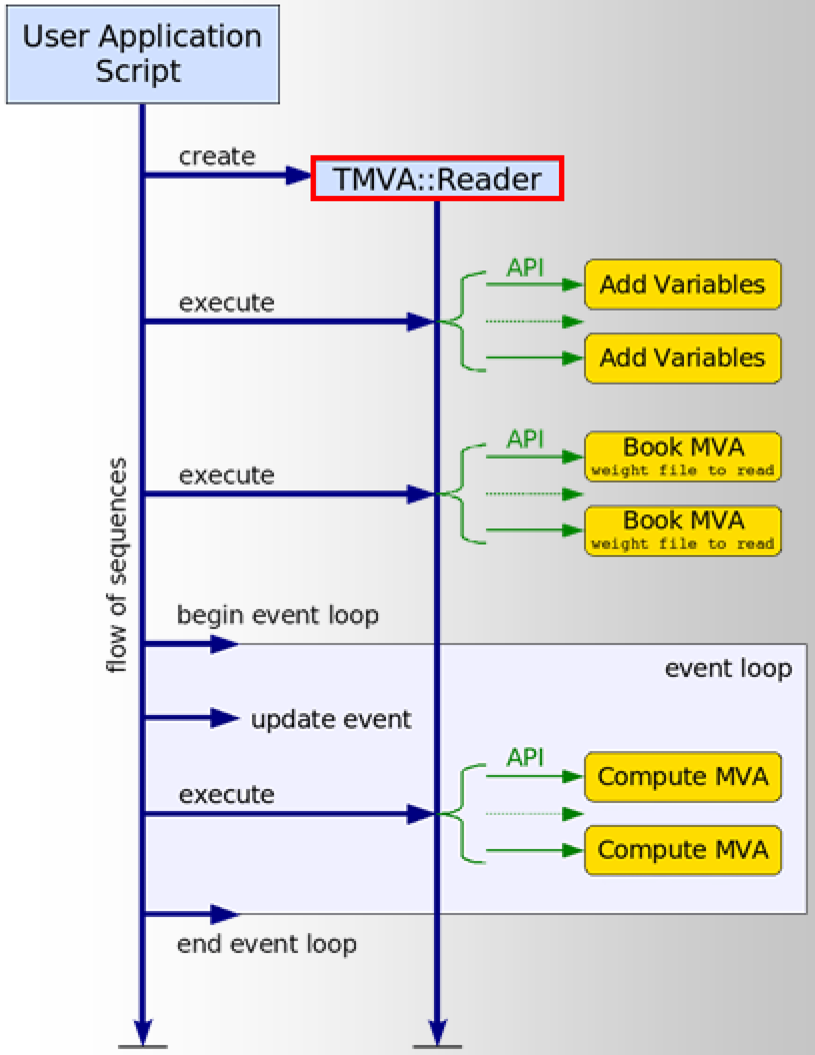
➧ Illustration of these steps with toy data samples



→ *T*MVA tutorial

Code Flow for *Training* and *Application* Phases





→ *TMVA* tutorial

Code Flow for *Training* and *Application* Phases

User Training Script
User Application Script

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

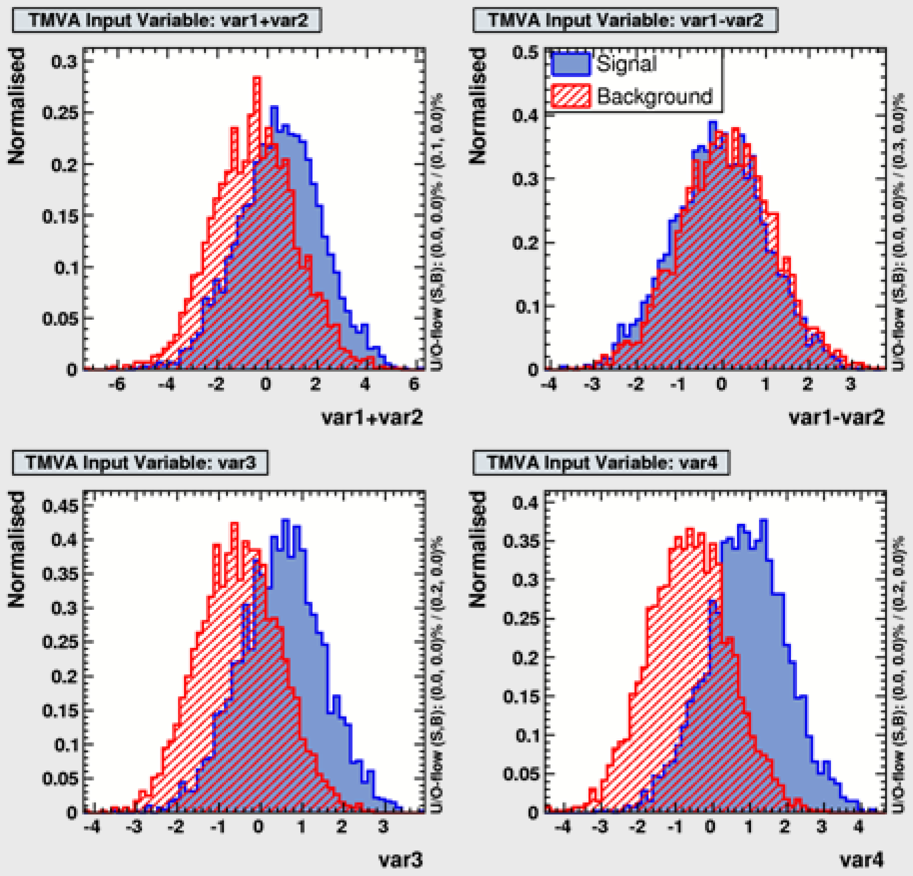
→ *T*MVA tutorial

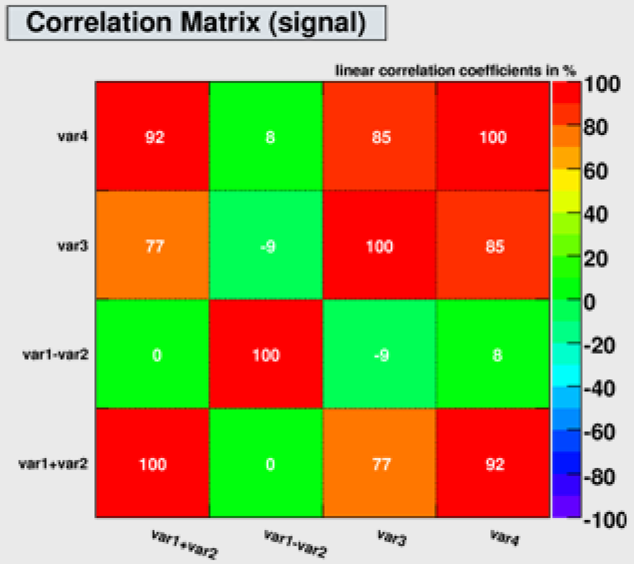
Data Preparation

- 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)

A Toy Example (idealized)

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

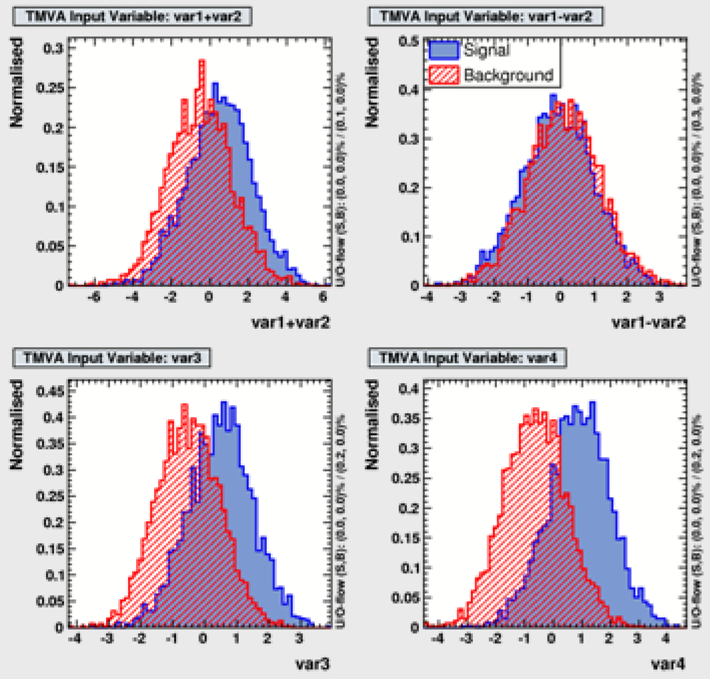


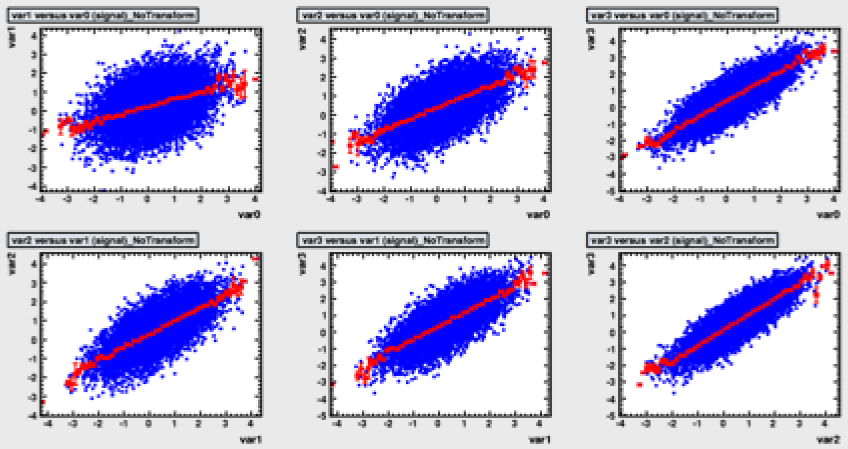


```
---------------------------------------
Rank : Variable    : Separation
---------------------------------------
     1 : var4           : 0.606
     2 : var1+var2 : 0.182
     3 : var3           : 0.173
     4 : var1-var2  : 0.014
---------------------------------------
```

Preprocessing the Input Variables

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

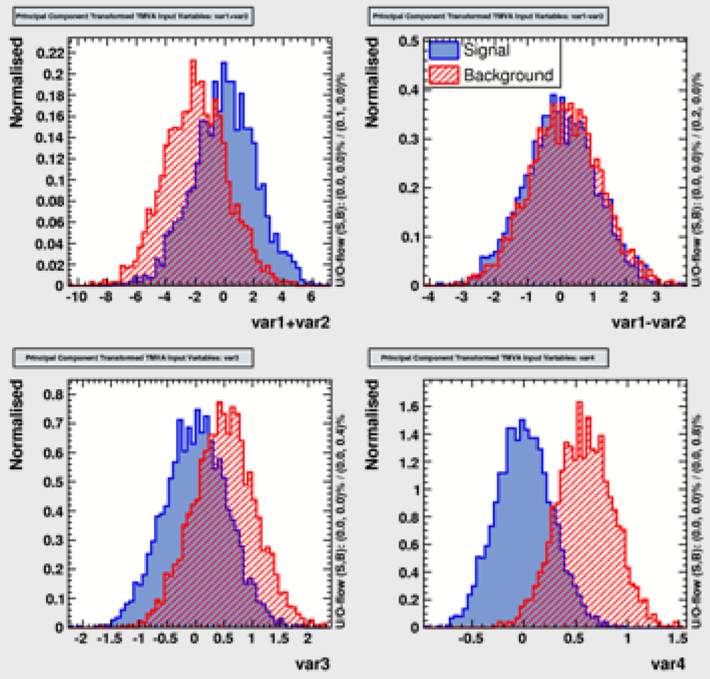


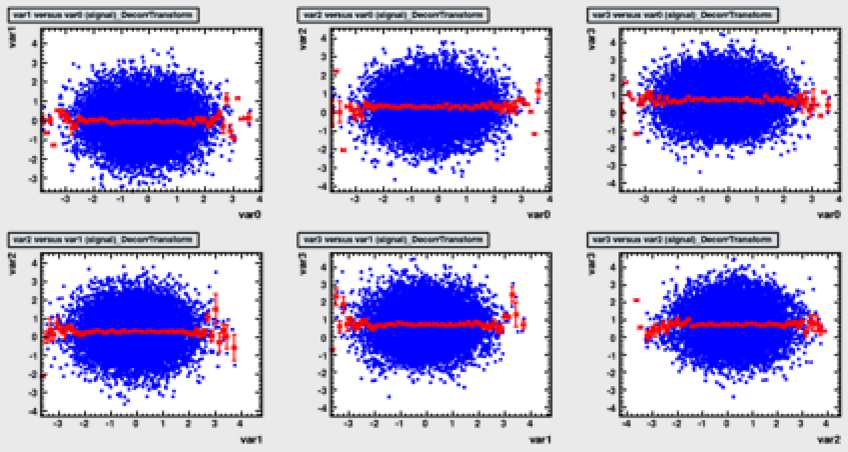


- Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

Preprocessing the Input Variables

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

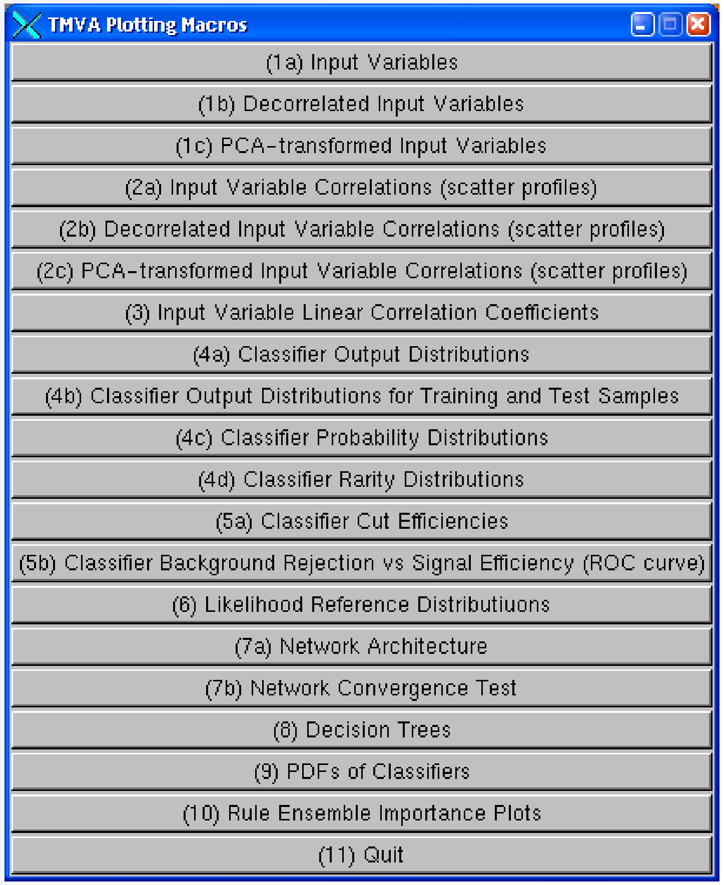




- Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

MVA Evaluation Framework

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



TMVA Plotting Macros

(1a) Input Variables
(1b) Decorrelated Input Variables
(1c) PCA-transformed Input Variables
(2a) Input Variable Correlations (scatter profiles)
(2b) Decorrelated Input Variable Correlations (scatter profiles)
(2c) PCA-transformed Input Variable Correlations (scatter profiles)
(3) Input Variable Linear Correlation Coefficients
(4a) Classifier Output Distributions
(4b) Classifier Output Distributions for Training and Test Samples
(4c) Classifier Probability Distributions
(4d) Classifier Rarity Distributions
(5a) Classifier Cut Efficiencies
(5b) Classifier Background Rejection vs Signal Efficiency (ROC curve)
(6) Likelihood Reference Distributiuons
(7a) Network Architecture
(7b) Network Convergence Test
(8) Decision Trees
(9) PDFs of Classifiers
(10) Rule Ensemble Importance Plots
(11) Quit

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

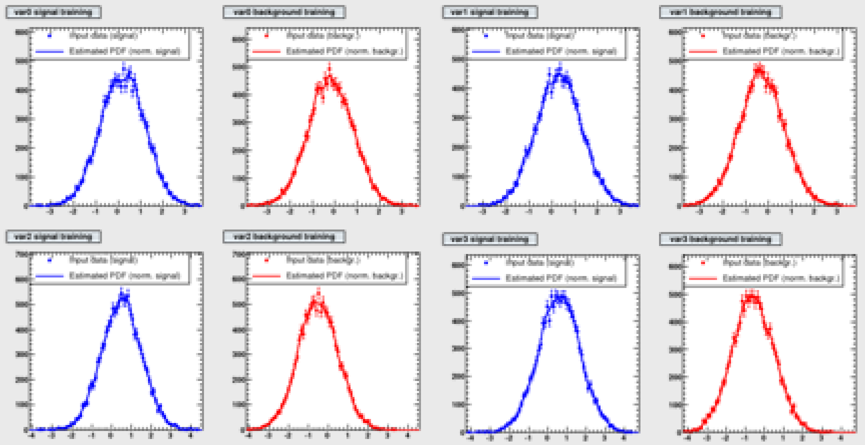
B rejection versus S efficiency

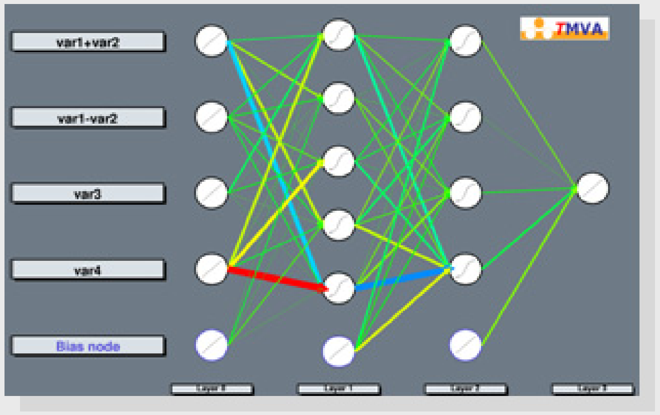
Classifier-specific plots:
- Likelihood reference distributions
- Classifier PDFs (for probability output and Rarity)
- Network architecture, weights and convergence
- Rule Fitting analysis plots

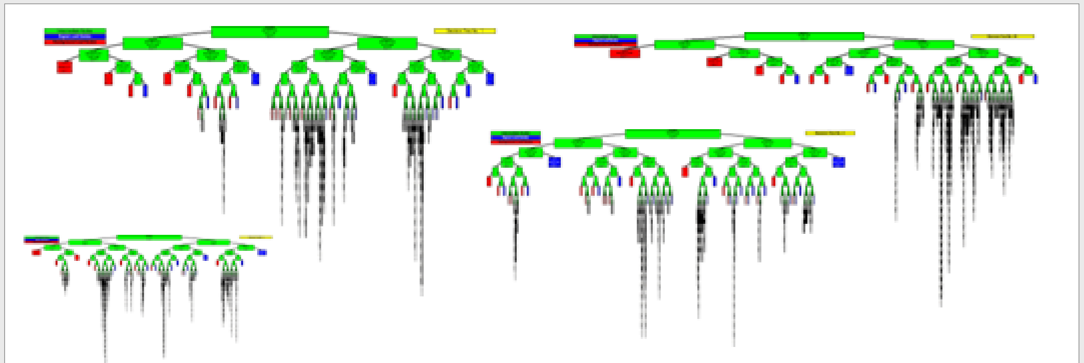
- Visualise decision trees

Evaluating the Classifier Training (I)

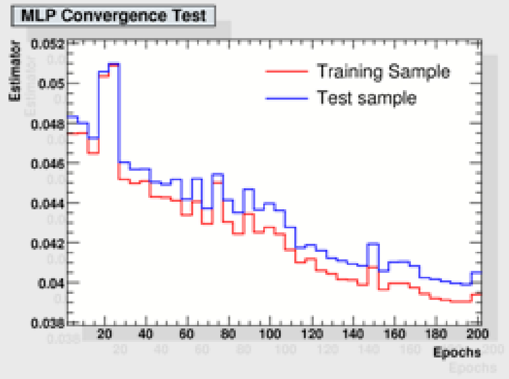
- Projective likelihood PDFs, MLP training, BDTs, …





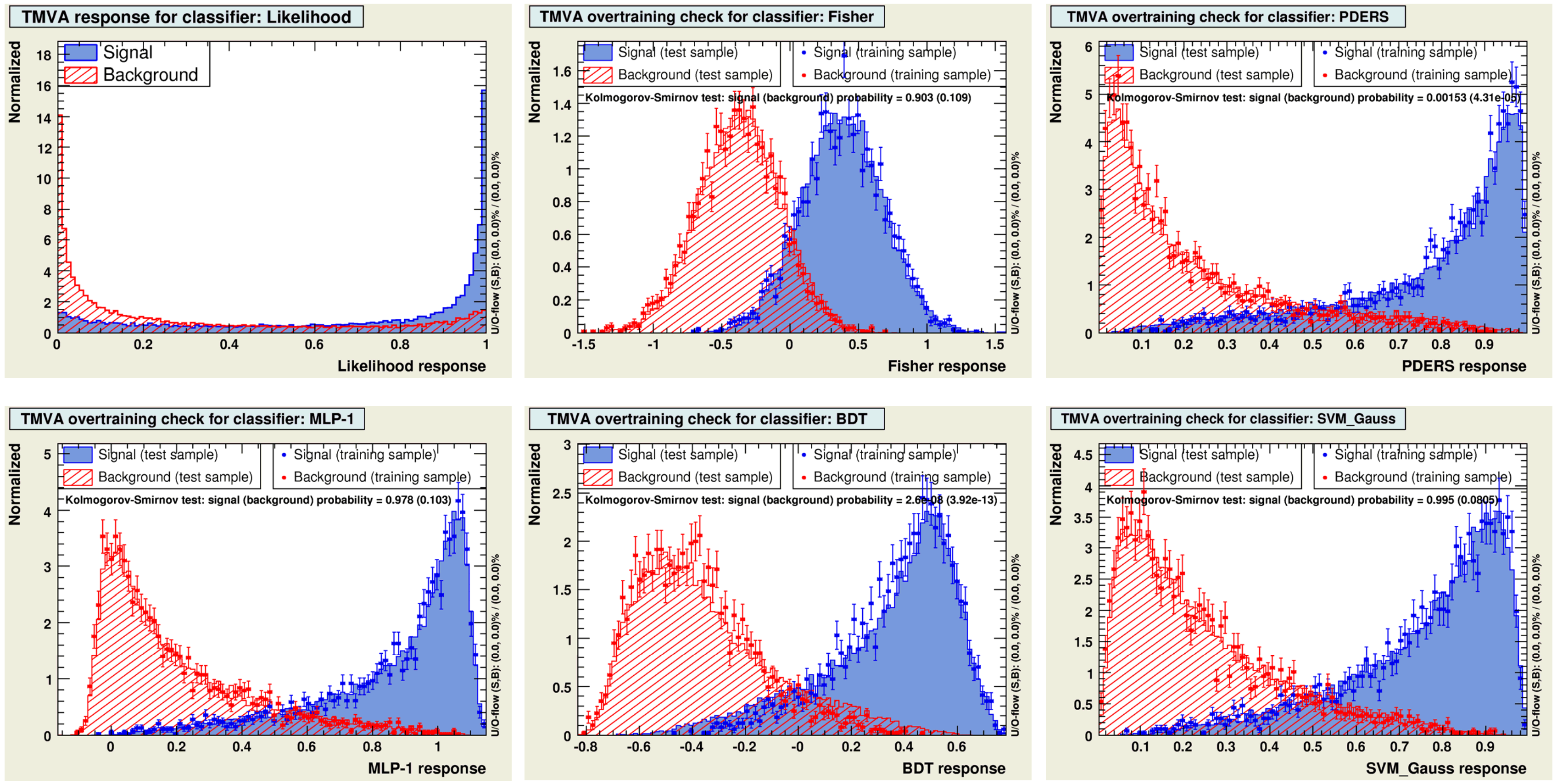


average no. of nodes before/after pruning: 4193 / 968



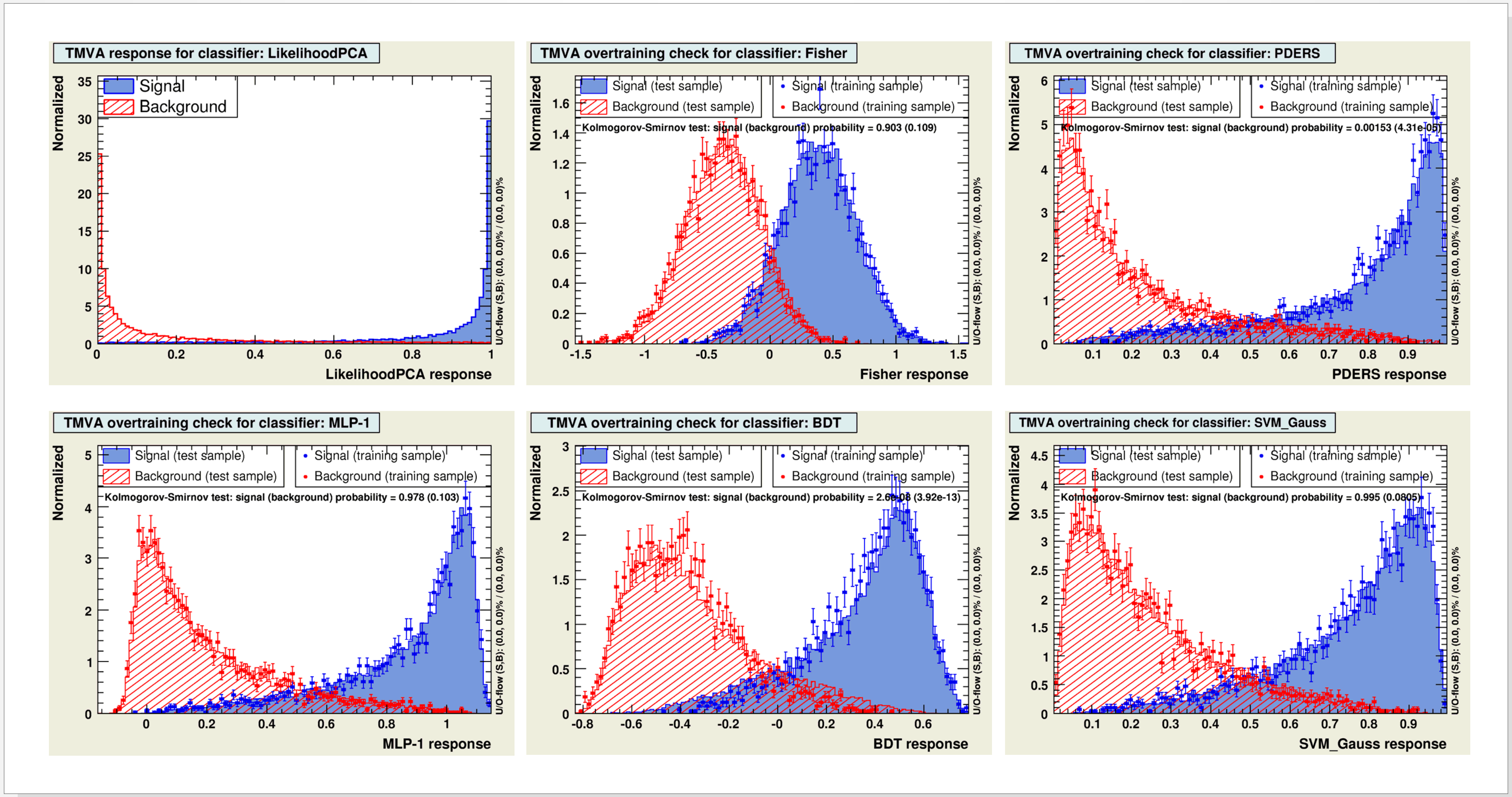
Testing the Classifiers

- Classifier output distributions for independent test sample:



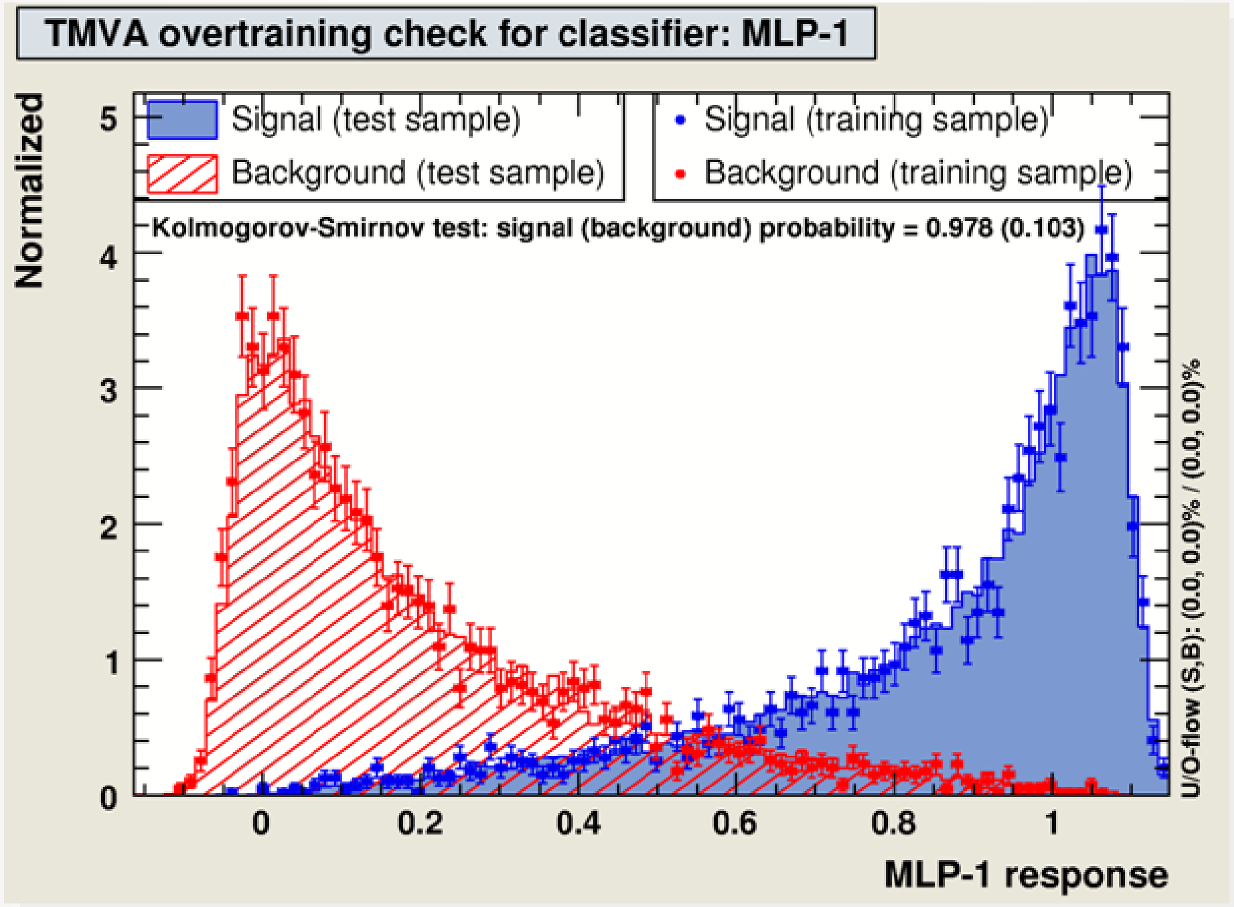
Testing the Classifiers

- Classifier output distributions for independent test sample:



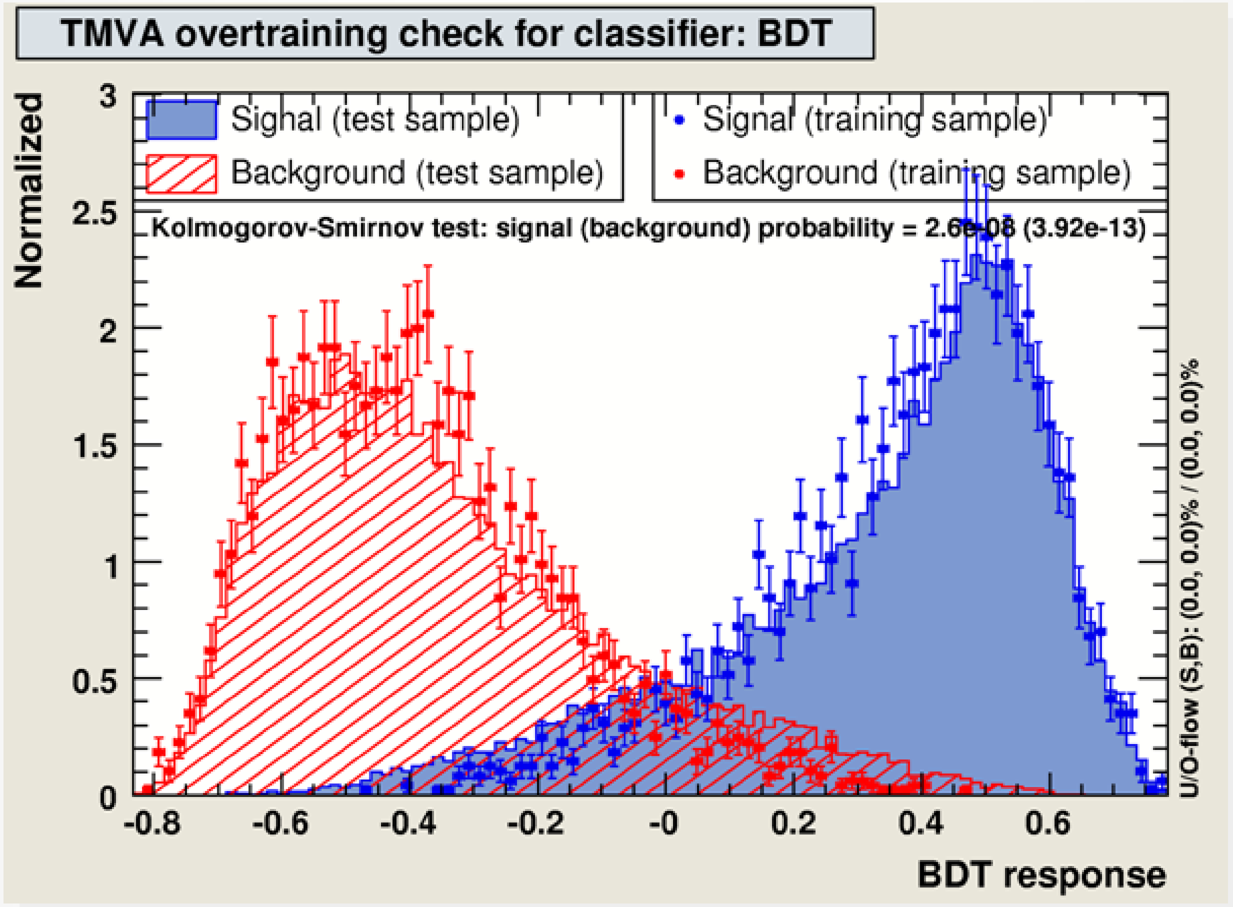
Evaluating the Classifier Training (II)

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



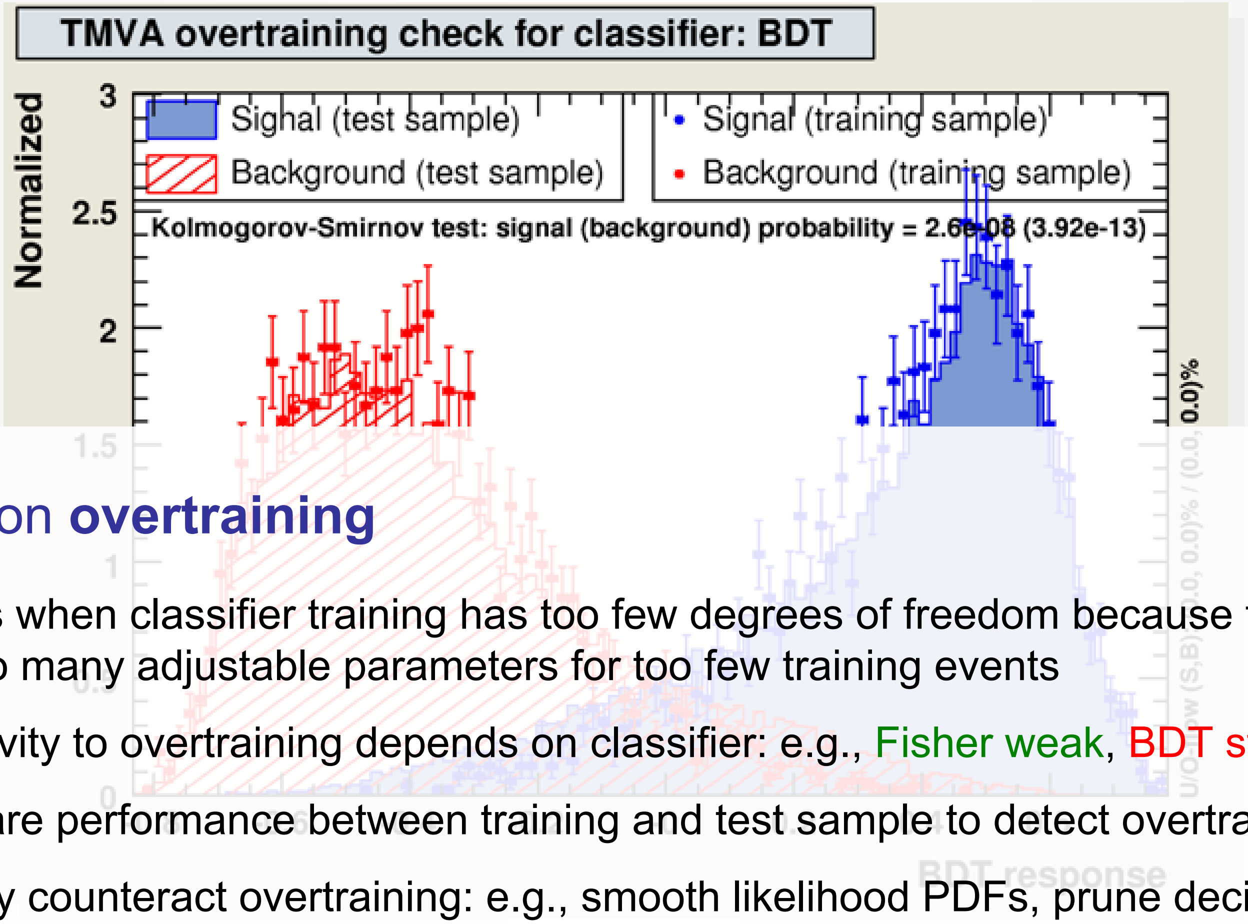
Evaluating the Classifier Training (II)

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



Evaluating the Classifier Training (II)

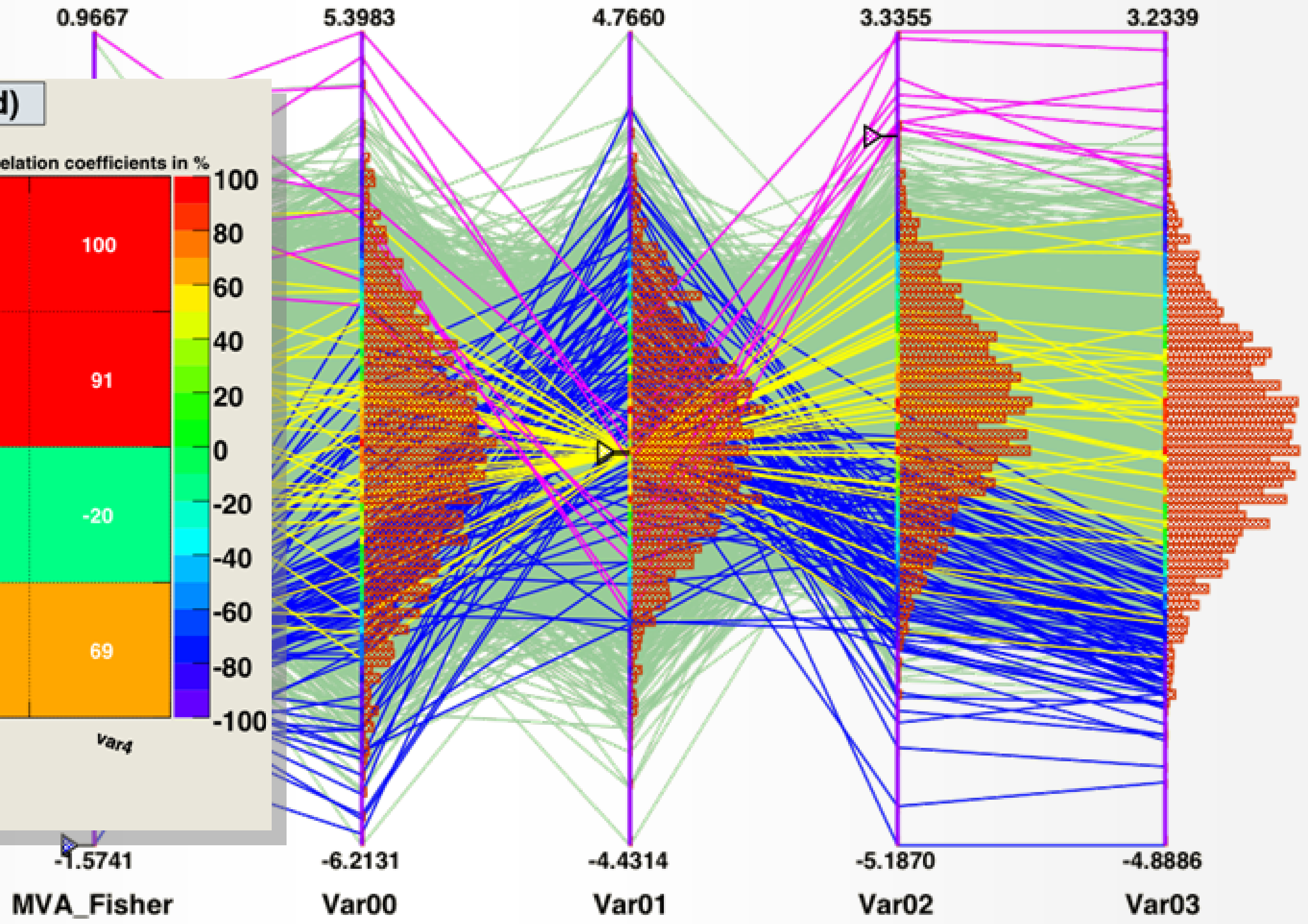
- Check for overtraining: classifier output for test *and* training samples …

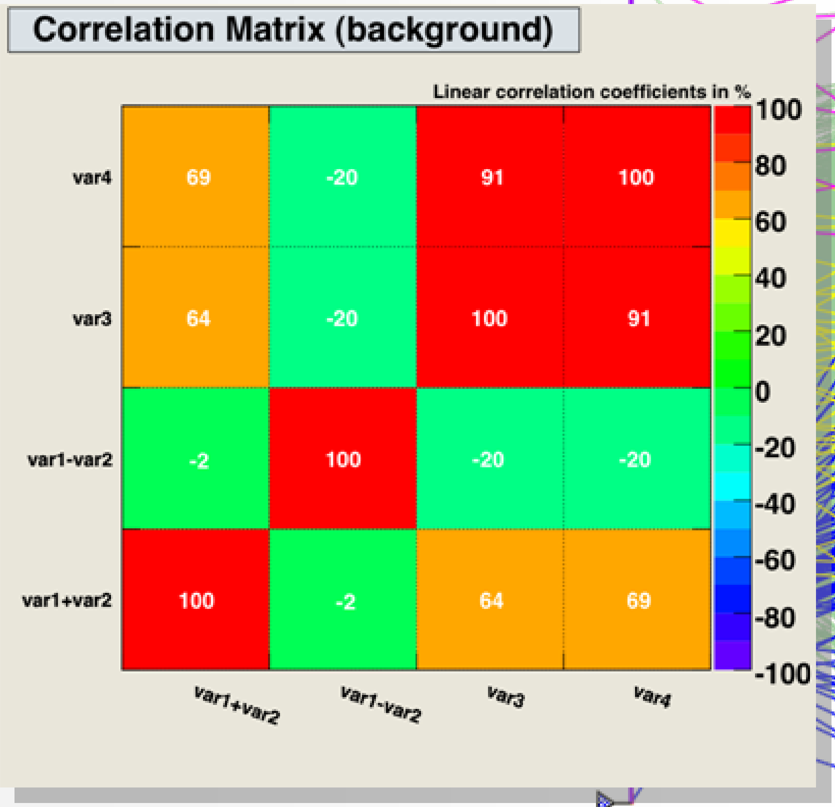


- Remark on **overtraining**
  - Occurs when classifier training has too few degrees of freedom because the classifier has too many adjustable parameters for too few training events
  - Sensitivity to overtraining depends on classifier: e.g., Fisher weak, BDT strong
  - Compare performance between training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

Evaluating the Classifier Training (III)

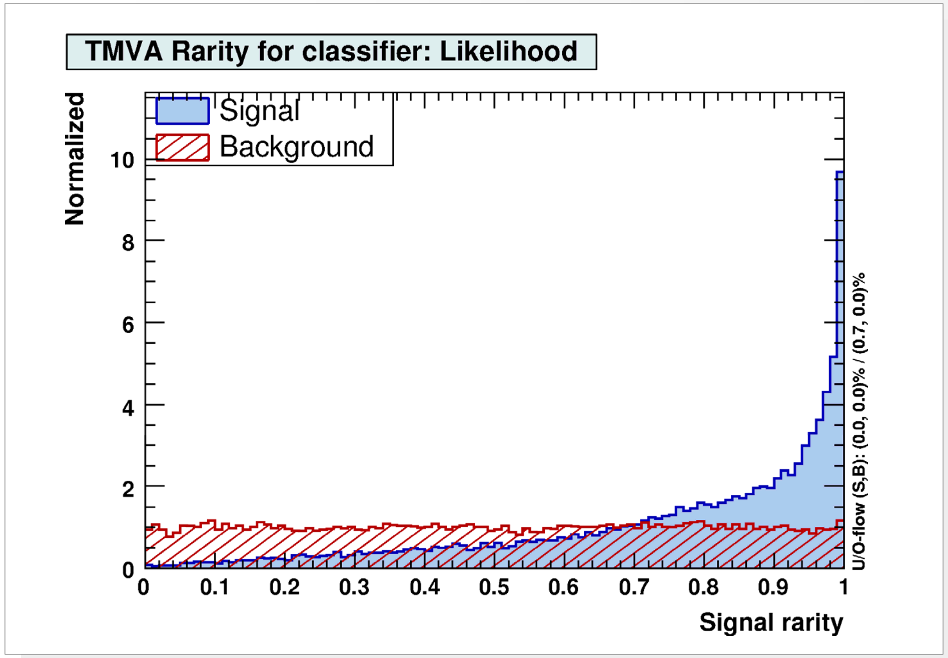
- *Parallel Coordinates* (ROOT class)





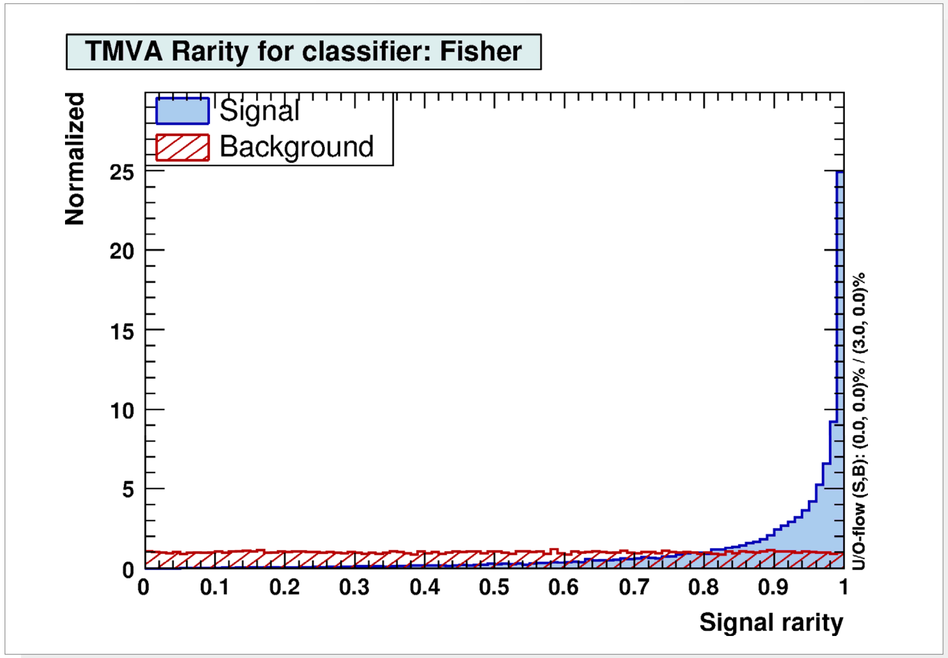
Evaluating the Classifier Training (IV)

- There is no unique way to express the performance of a classifier
  → several benchmark quantities computed by *T*MVA
  - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
  - The *Separation*: $\frac{1}{2}\int \frac{\left(\hat{y}_S(y)-\hat{y}_B(y)\right)^2}{\hat{y}_S(y)+\hat{y}_B(y)}dy$
  - "Rarity" implemented (background flat): $R(y)=\int_{-\infty}^{y}\hat{y}(y')dy'$
  - Other quantities … see Users Guide



Evaluating the Classifier Training (IV)

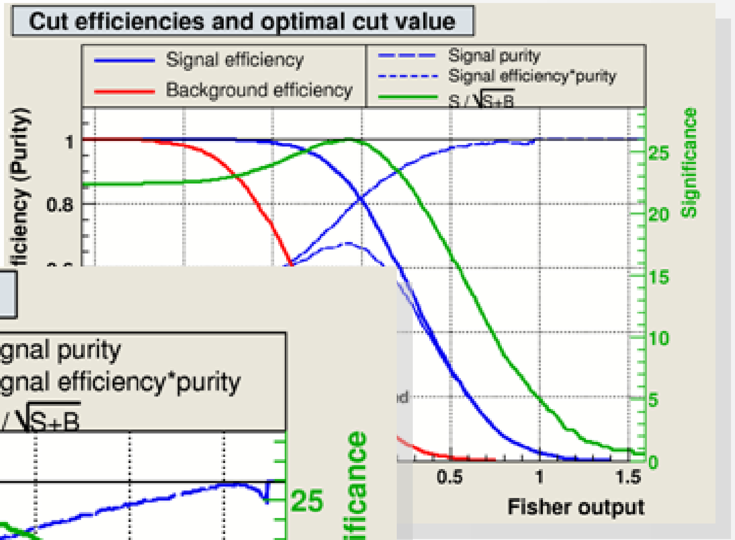
- There is no unique way to express the performance of a classifier → several benchmark quantities computed by *T*MVA
  - Signal eff. at various background effs. (= 1 – rejection) when cutting on classifier output
  - The *Separation*: $\frac{1}{2}\int\frac{(\hat{y}_S(y)-\hat{y}_B(y))^2}{\hat{y}_S(y)+\hat{y}_B(y)}dy$
  - "Rarity" implemented (background flat): $R(y)=\int_{-\infty}^{y}\hat{y}(y')dy'$
  - Other quantities … see Users Guide

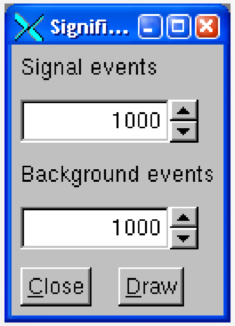


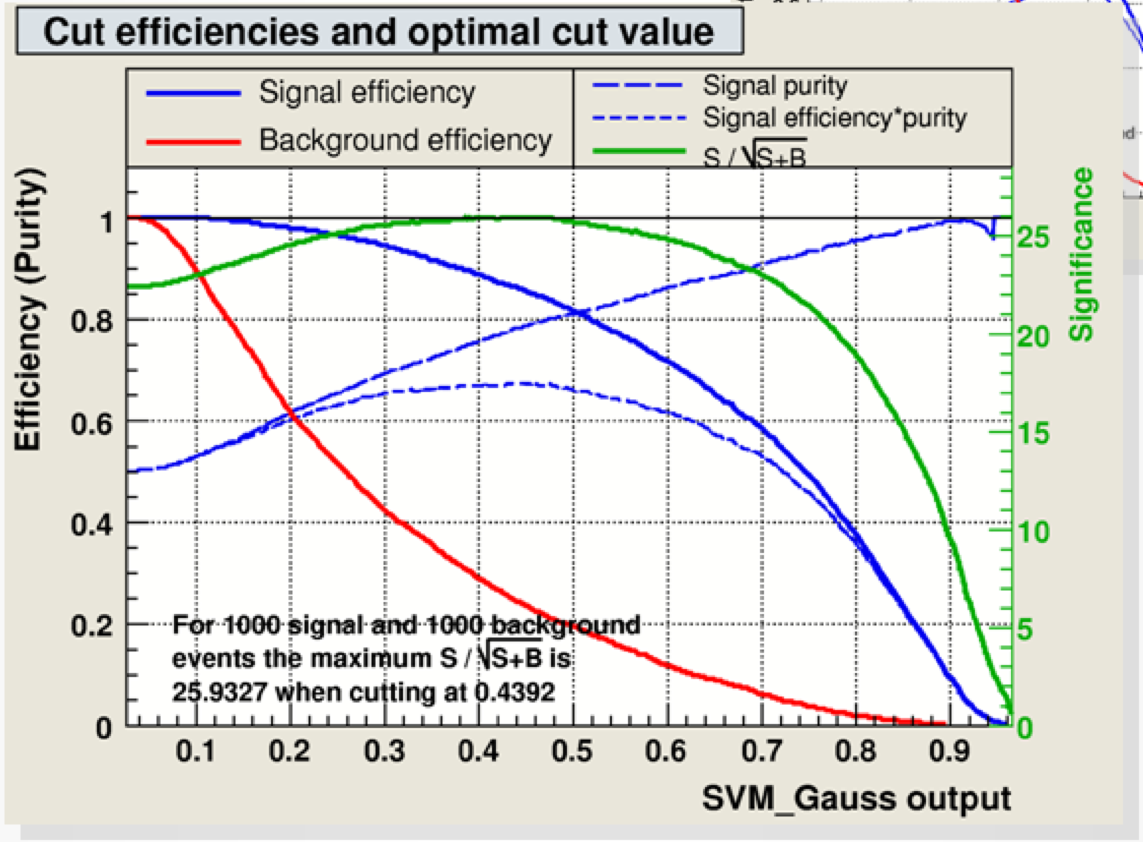
Evaluating the Classifier Training (V)

- Optimal cut for each classifiers …

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







For 1000 signal and 1000 background events the maximum S / √S+B is 25.9327 when cutting at 0.4392

Evaluating the Classifiers Training (VI) (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 Training (VII) (taken from TMVA output…)

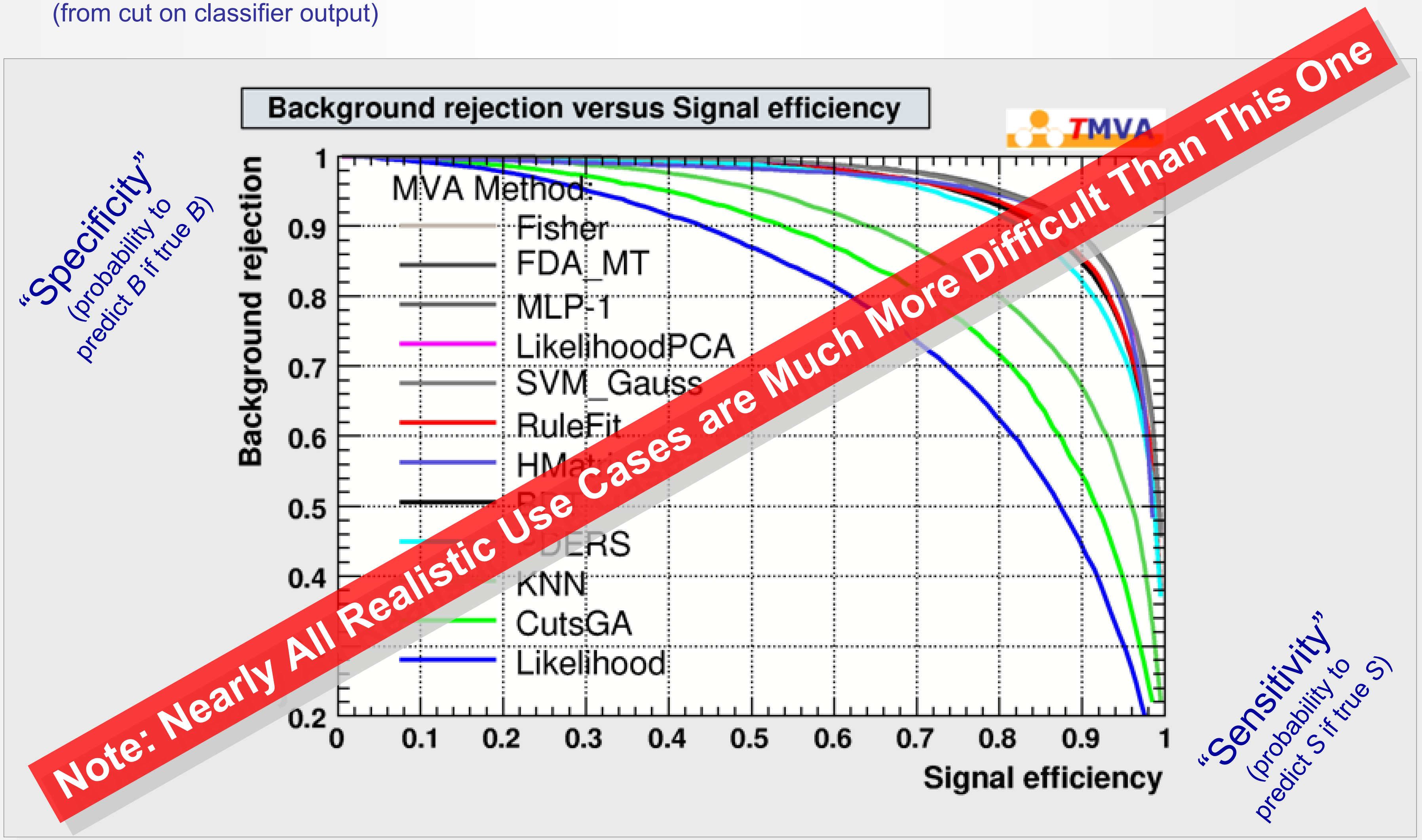
Better classifier

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

Check for over-training

Receiver Operating Characteristics (ROC) Curve

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

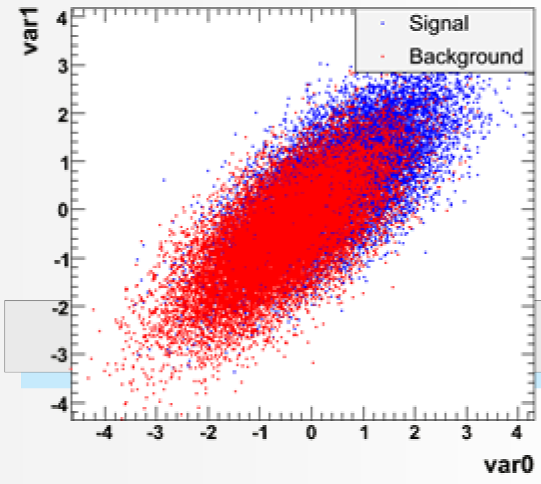


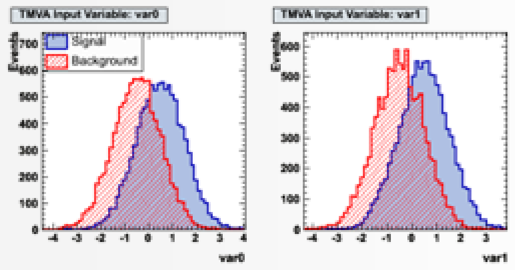
More Toy Examples

More Toys: Linear-, Cross-, Circular Correlations

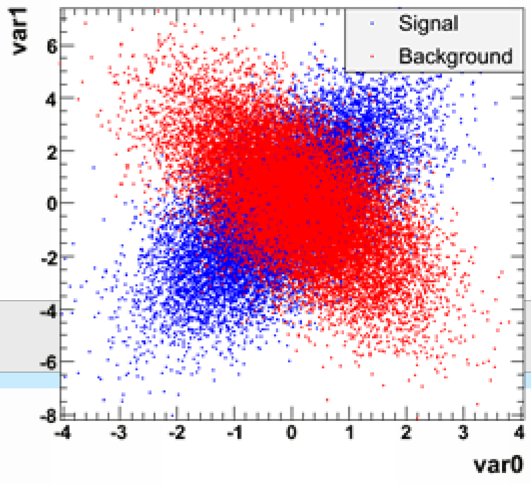
- Illustrate the behaviour of linear and nonlinear classifiers

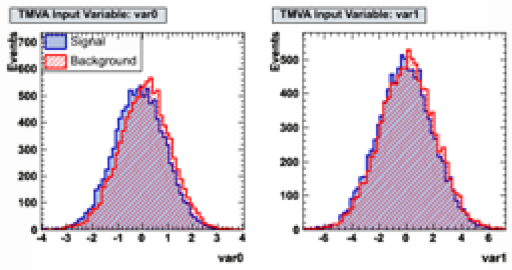
Linear correlations
(same for signal and background)



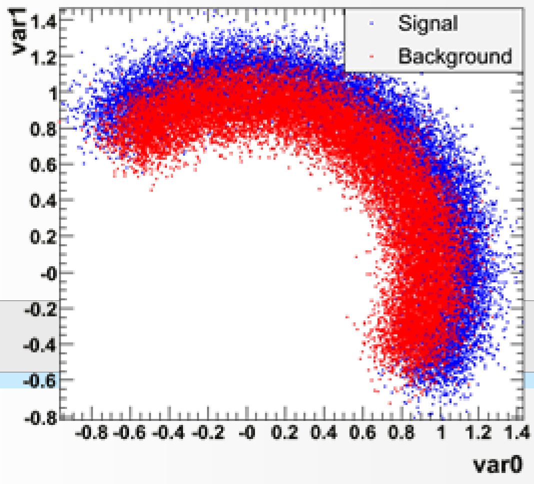


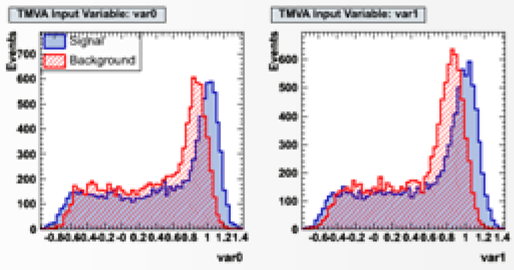
Linear correlations
(opposite for signal and background)





Circular correlations
(same for signal and background)

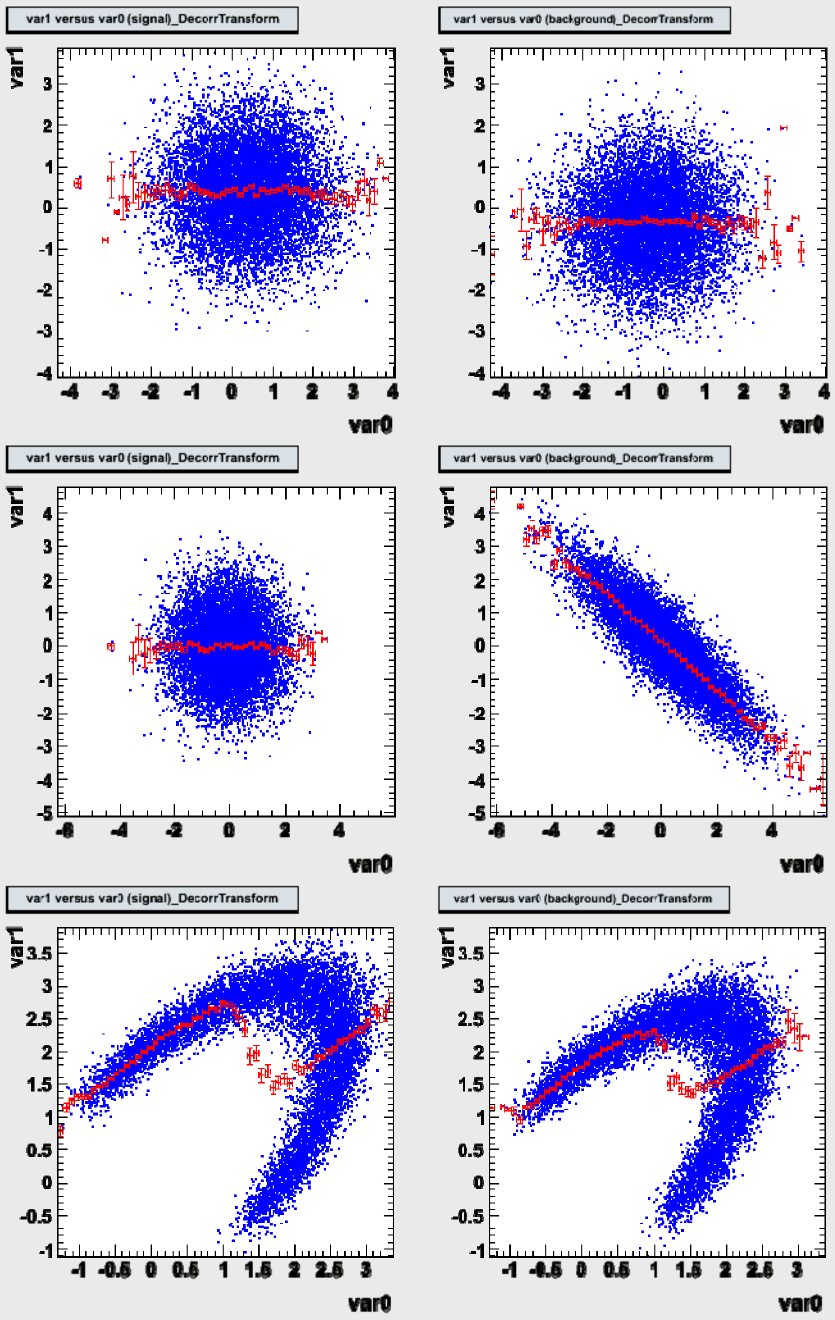




- How does linear decorrelation affect strongly nonlinear cases ?

Original correlations

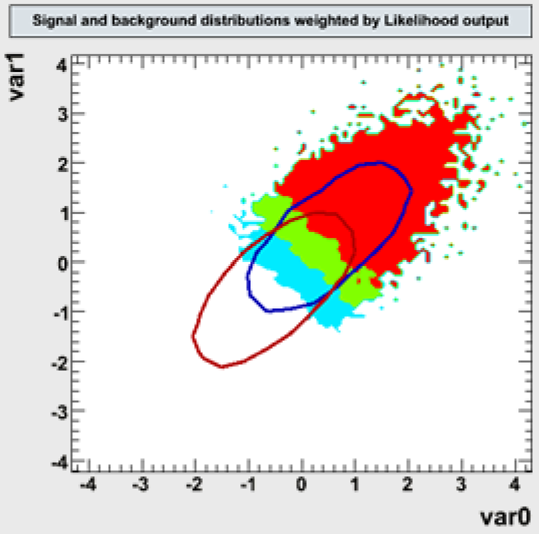
SQRT decorrelation

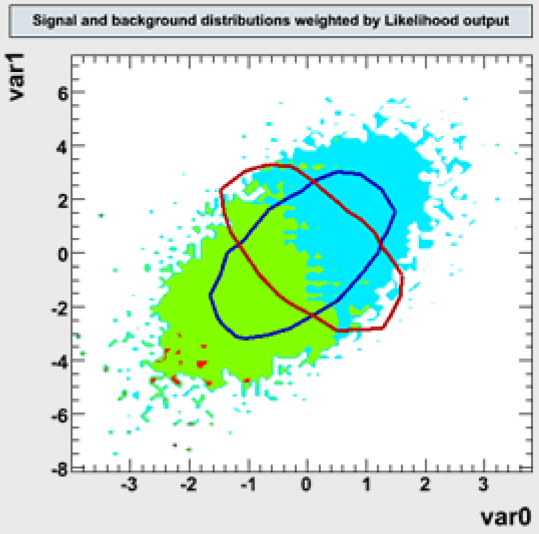


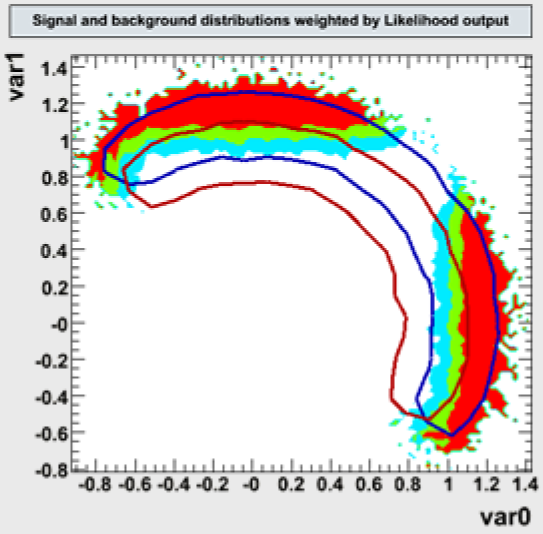
Weight Variables by Classifier Output

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

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





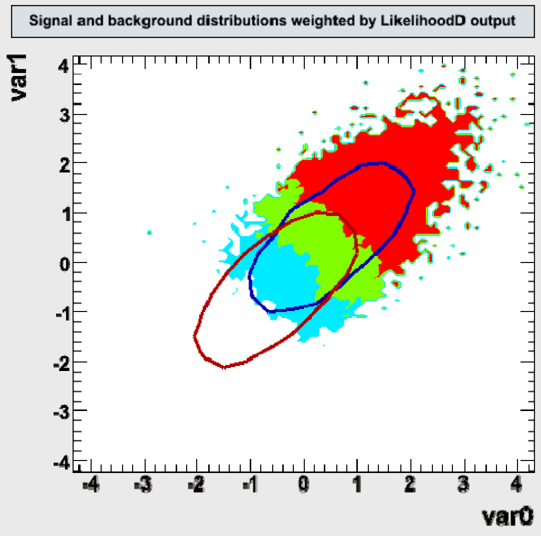


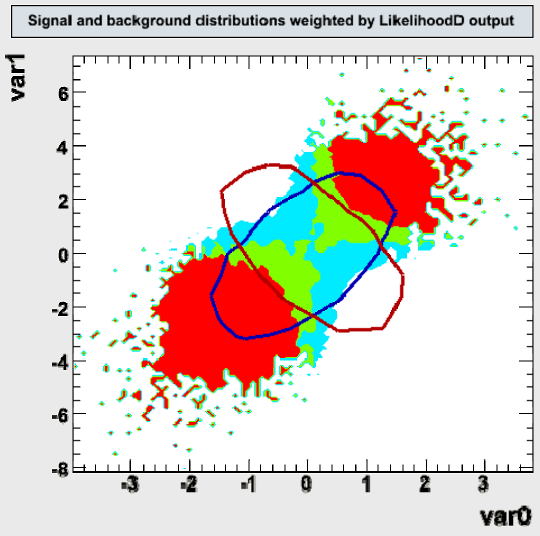
Likelihood

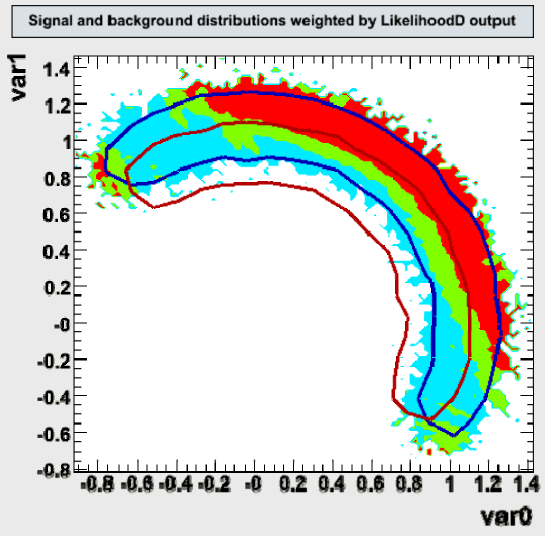
Weight Variables by Classifier Output

- 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 →

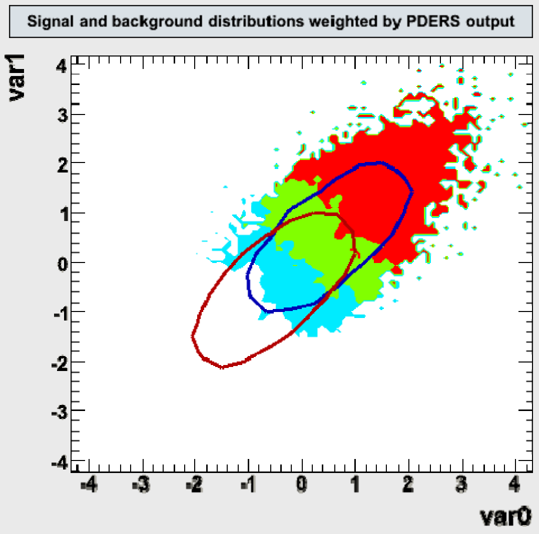
Weight Variables by Classifier Output

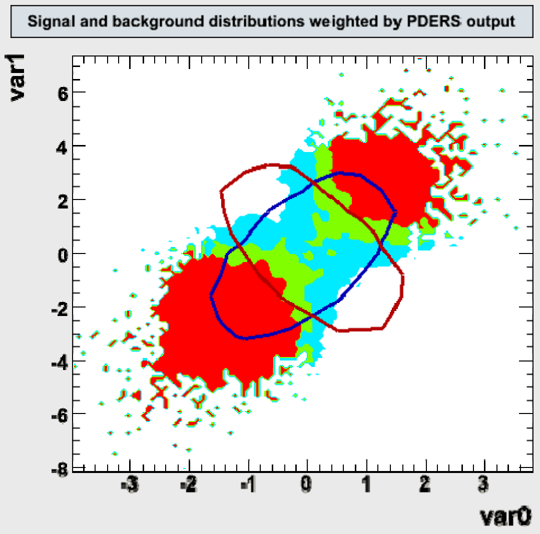
- How well do the classifier resolve the various correlation patterns ?

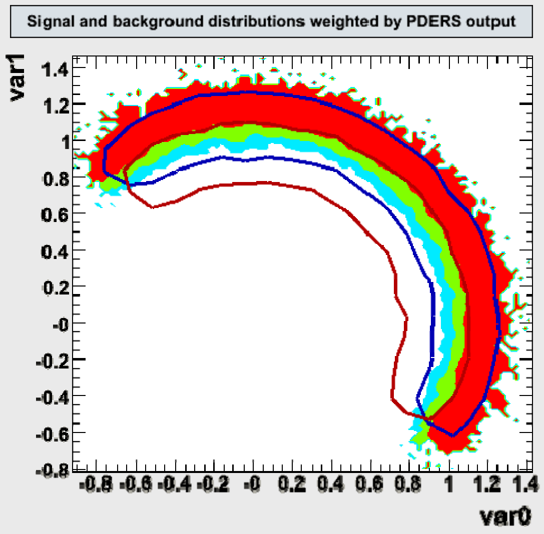
Linear correlations
(same for signal and background)

Cross-linear correlations
(opposite for signal and background)

Circular correlations
(same for signal and background)







← PDERS →

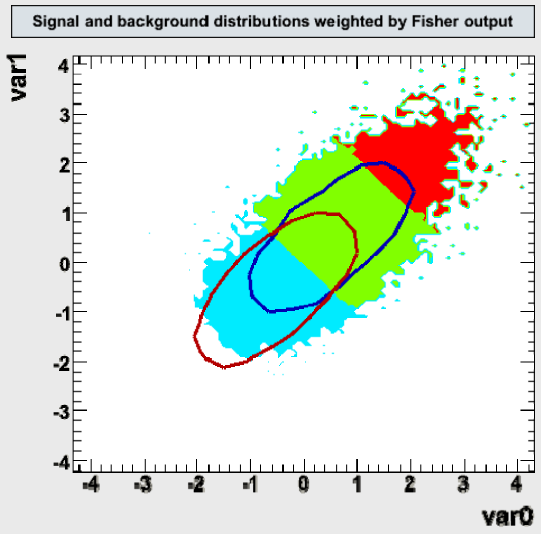
Weight Variables by Classifier Output

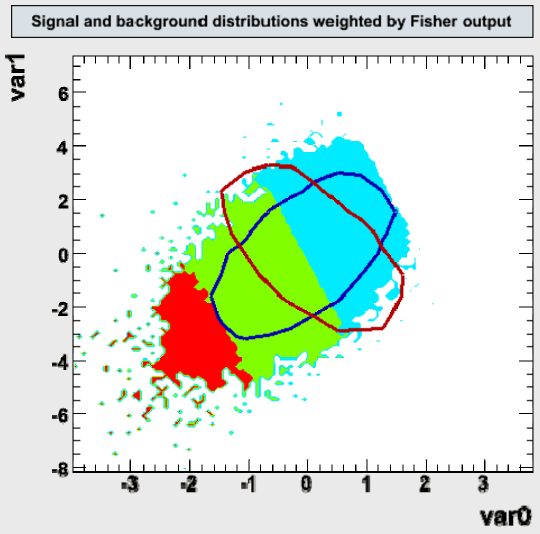
- How well do the classifier resolve the various correlation patterns ?

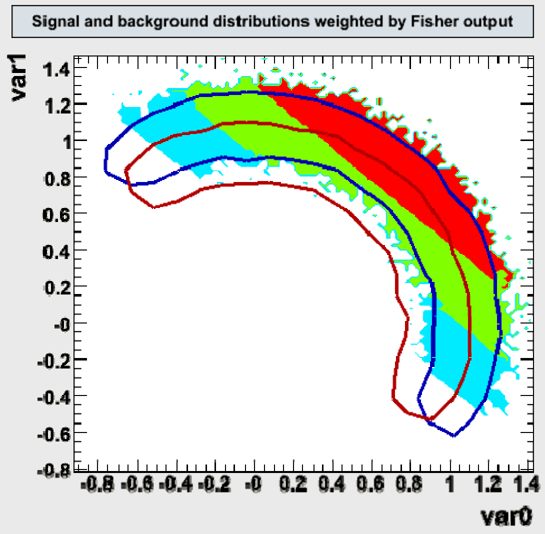
Linear correlations
(same for signal and background)

Cross-linear correlations
(opposite for signal and background)

Circular correlations
(same for signal and background)





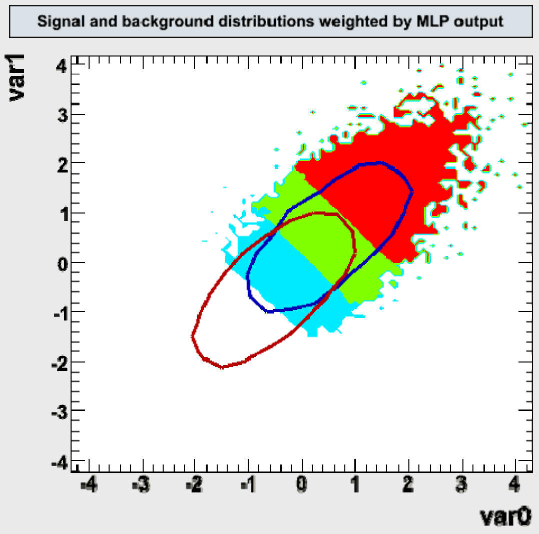


← Fisher →

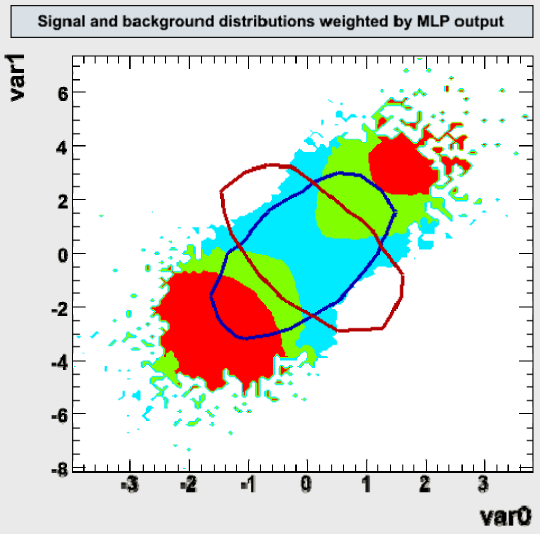
Weight Variables by Classifier Output

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

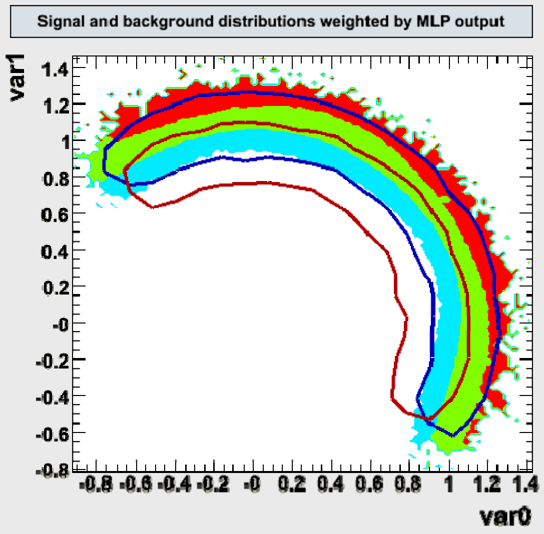
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



Circular correlations
(same for signal and background)

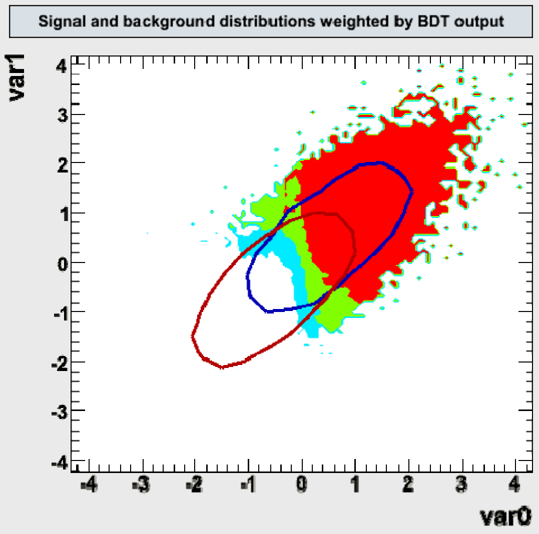


MLP

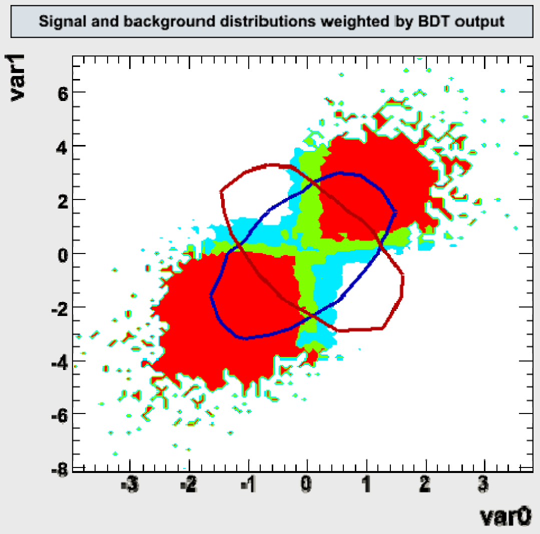
Weight Variables by Classifier Output

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

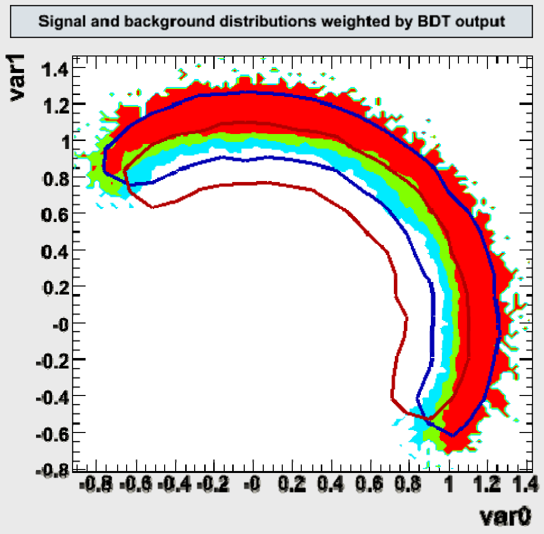
Linear correlations
(same for signal and background)



Cross-linear correlations
(opposite for signal and background)



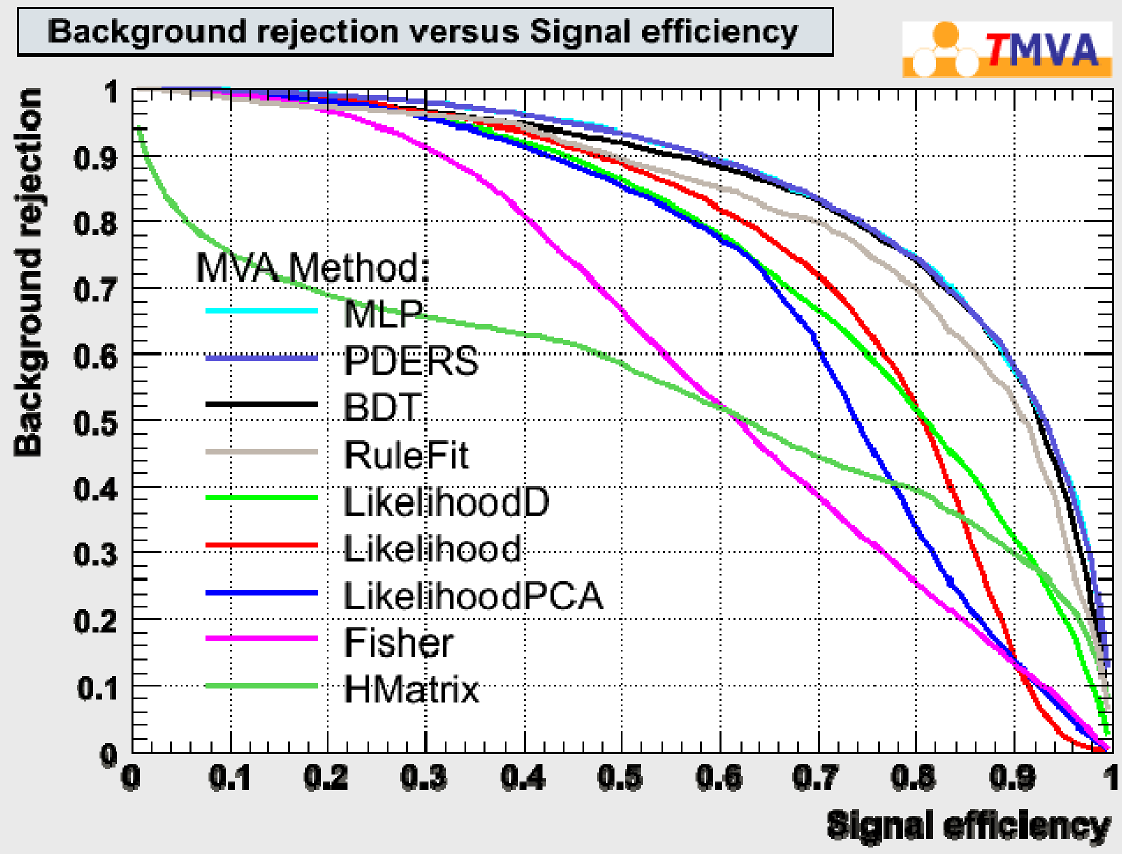
Circular correlations
(same for signal and background)



← BDT →

Final Classifier Performance

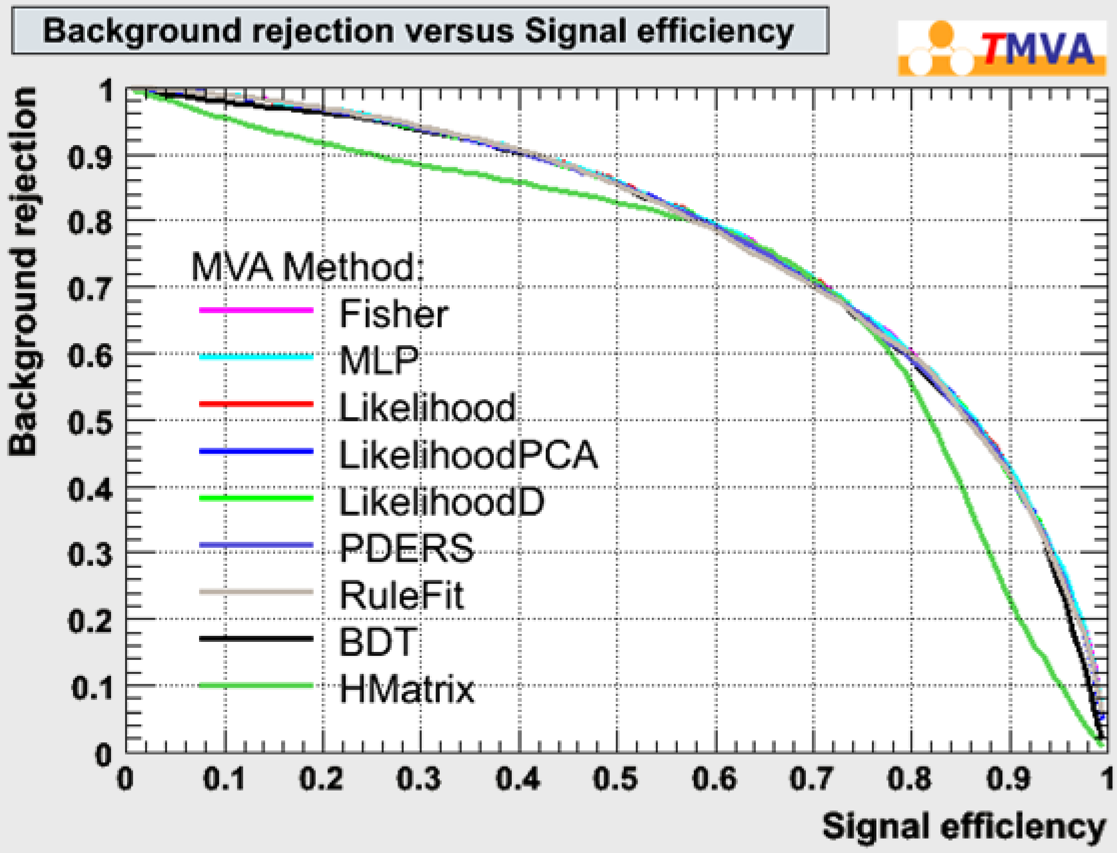
- Background rejection versus signal efficiency curve:



Circular Example

Final Classifier Performance

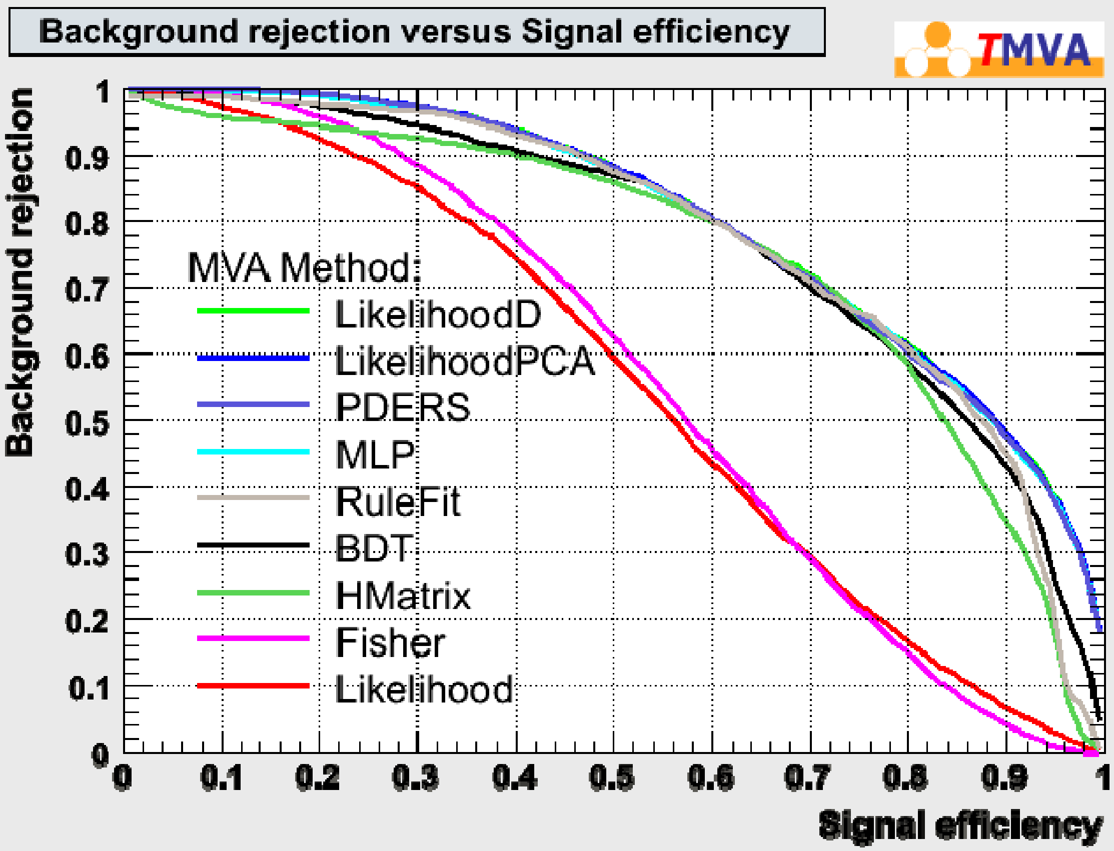
- Background rejection versus signal efficiency curve:



Linear Example

Final Classifier Performance

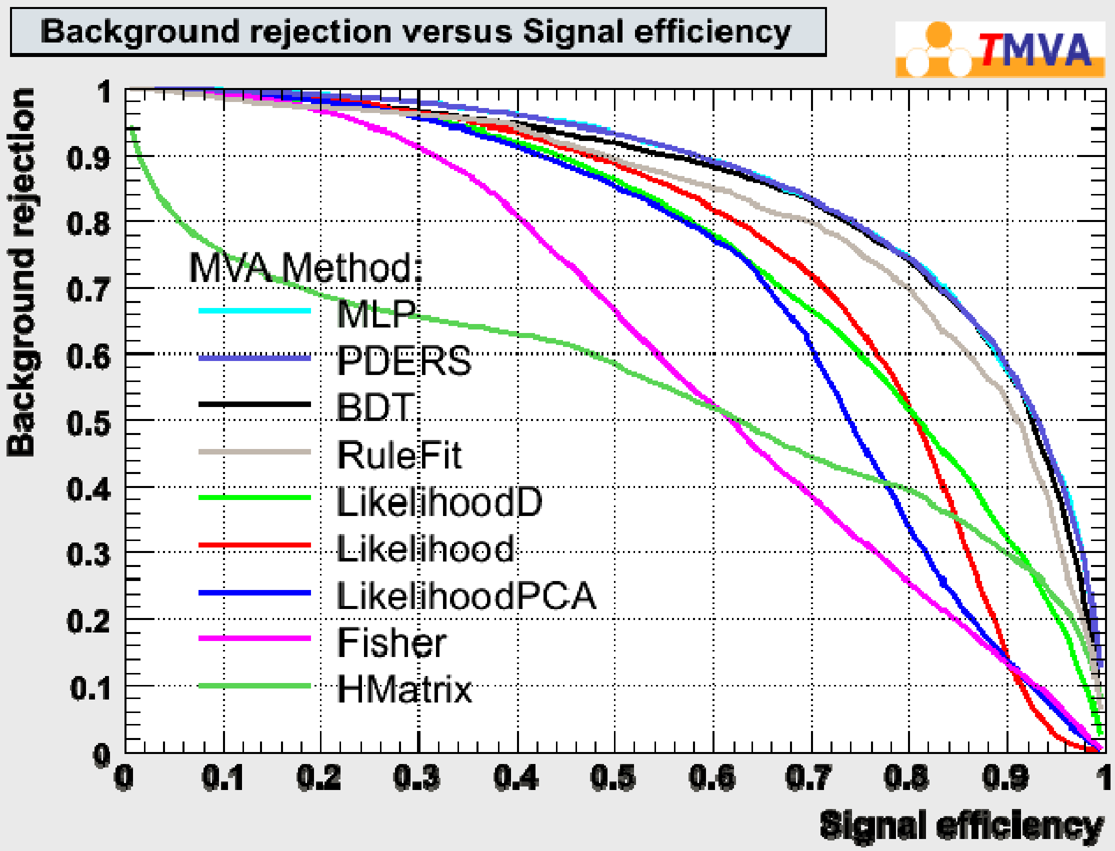
- Background rejection versus signal efficiency curve:



Cross Example

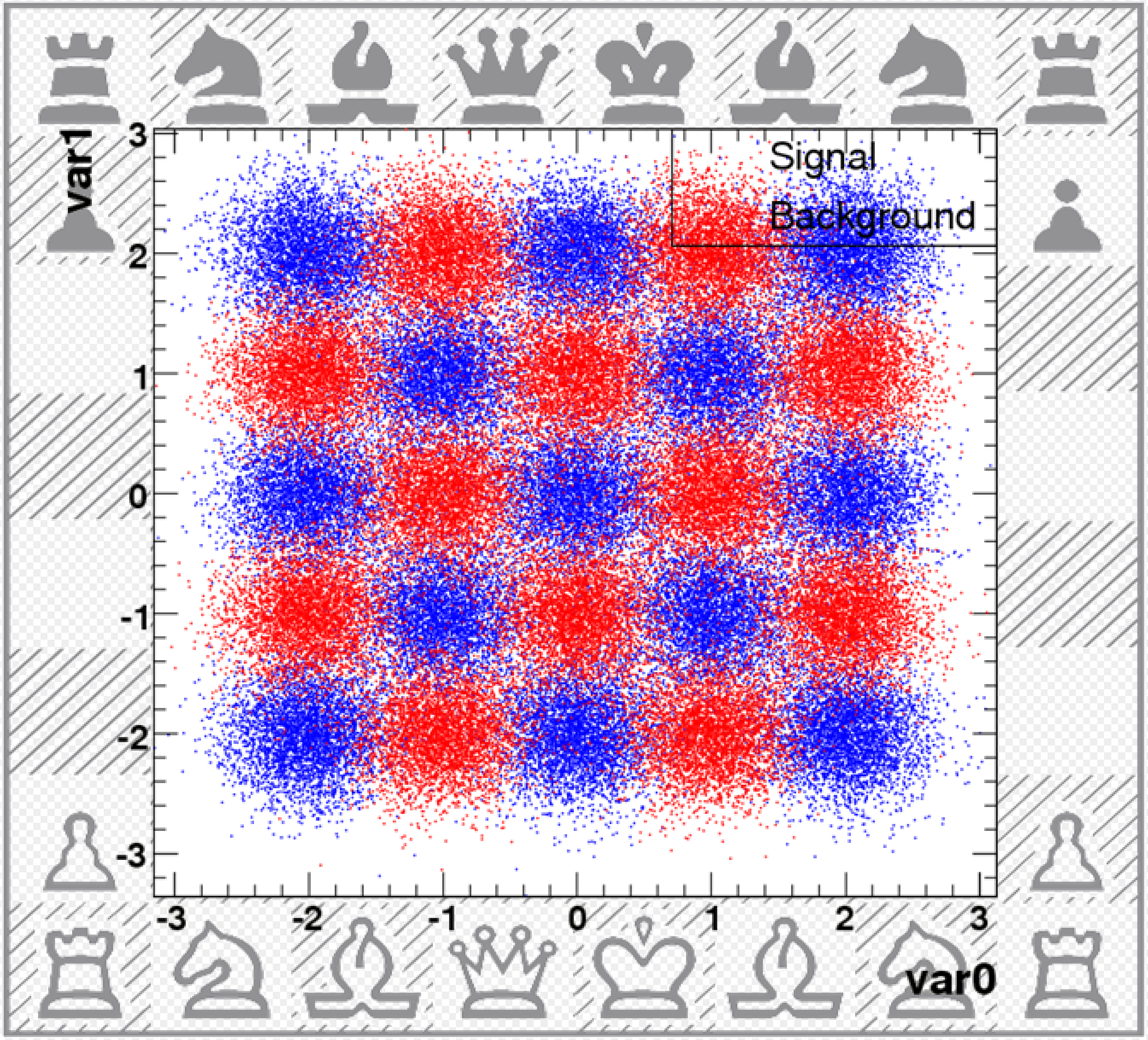
Final Classifier Performance

- Background rejection versus signal efficiency curve:

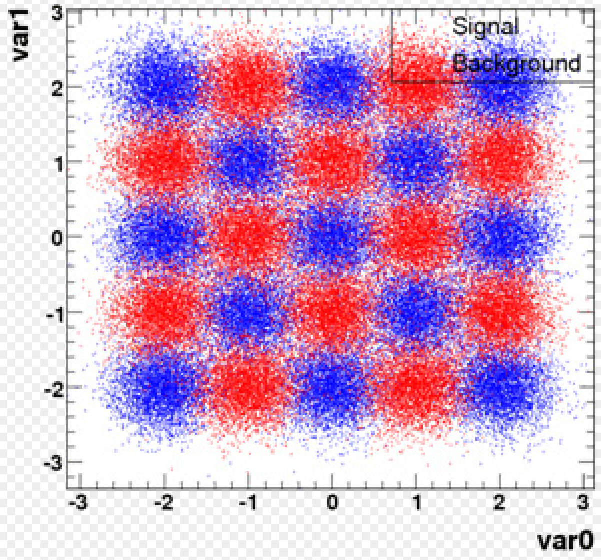


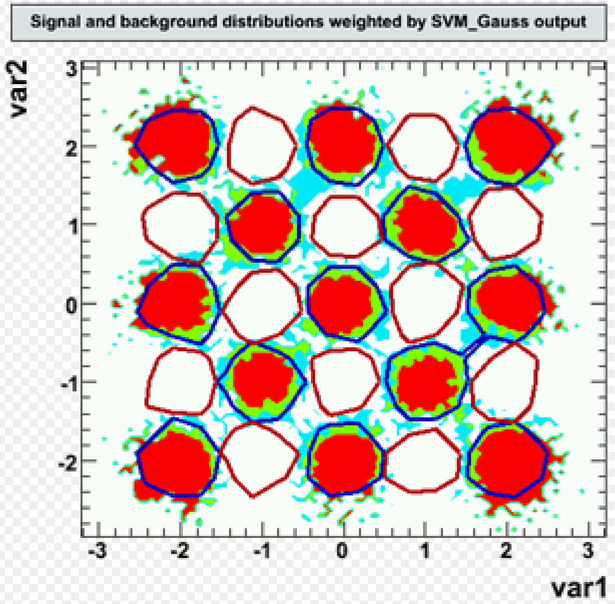
Circular Example

The *Chessboard* Toy

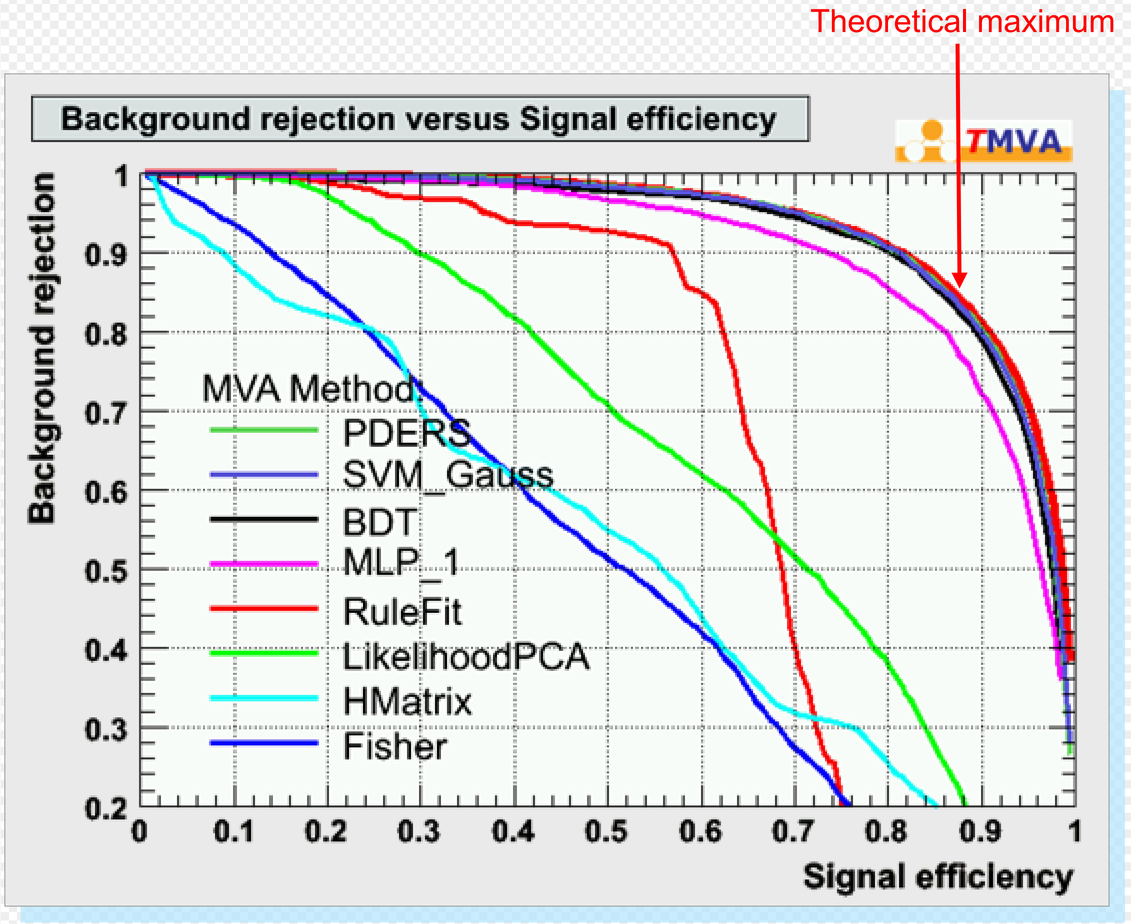


The *Chessboard* Toy





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



Summary

No Single Best Classifier …

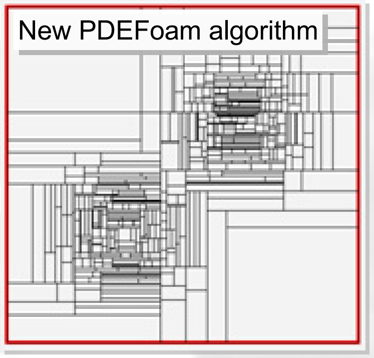
Criteria		Classifiers								
		Cuts	Likeli-hood	PDERS / k-NN	H-Matrix	Fisher	MLP	BDT	RuleFit	SVM
Perfor-mance	no / linear correlations	😐	🙂	🙂	😐	🙂	🙂	😐	🙂	🙂
	nonlinear correlations	😐	☹	🙂	☹	☹	🙂	🙂	😐	🙂
Speed	Training	☹	🙂	🙂	🙂	🙂	😐	☹	😐	☹
	Response	🙂	🙂	☹/😐	🙂	🙂	🙂	😐	😐	😐
Robust -ness	Overtraining	🙂	😐	😐	🙂	🙂	☹	☹	😐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

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

Outlook to TMVA 4

Composite and boost 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 → end 2008



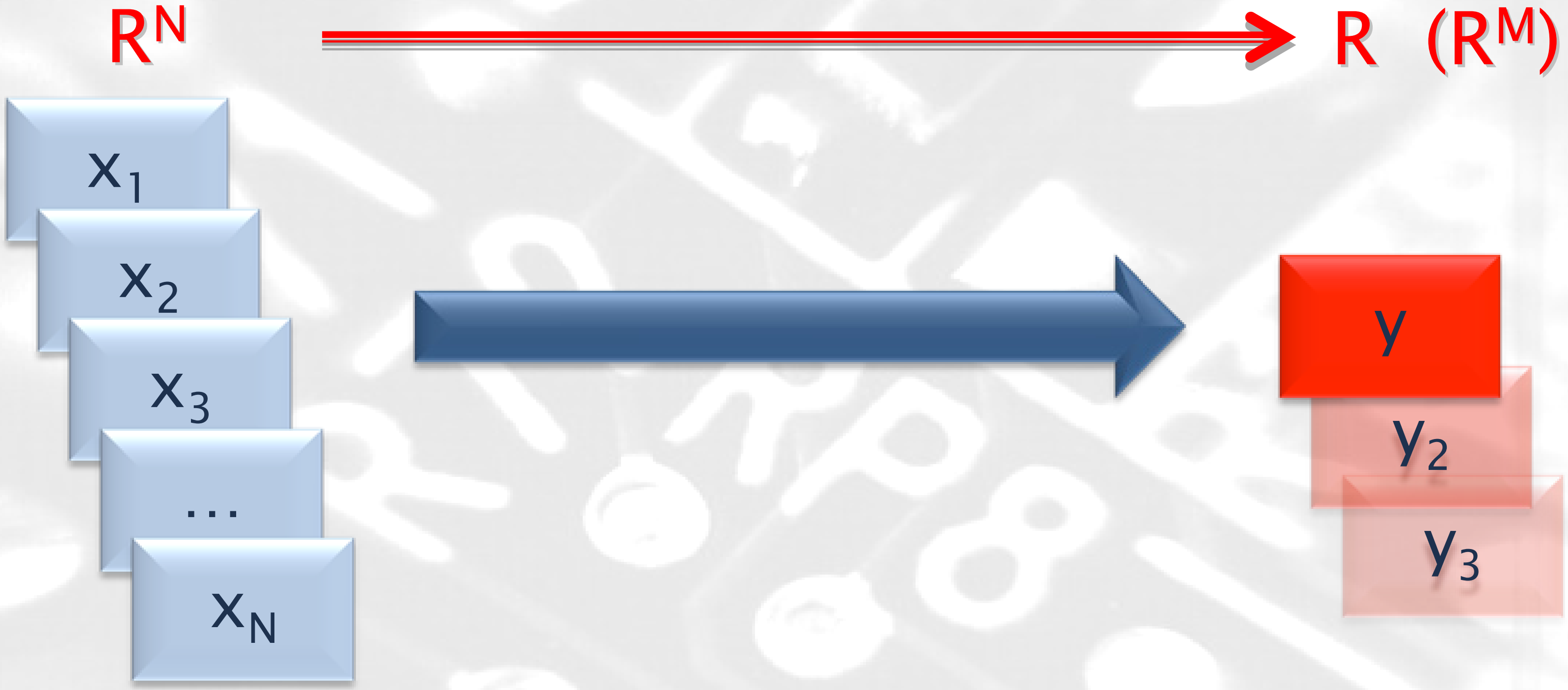
Extend TMVA to multivariate regression

Extend TMVA to multi-class classification

Generalised cross-validation

New and Improved Existing Classification/Regression Methods

Regression Analysis

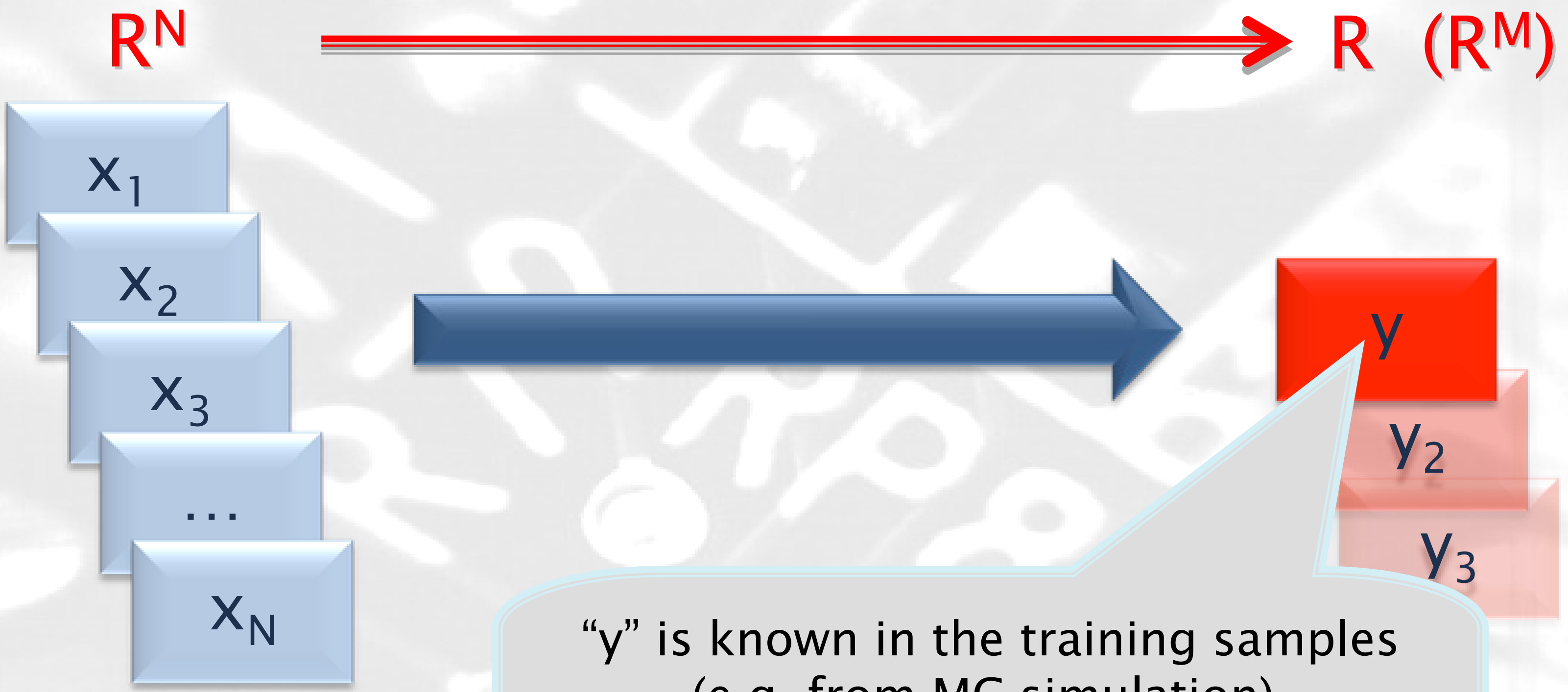


Multiple input variables (predictors)

Use their information (multivariate training)

To predict the outcome of one (or more) dependent variable(s) (target)

Regression Analysis



"y" is known in the training samples (e.g. from MC simulation).

"y" can be estimated for events where "y" is not known (e.g. real data)

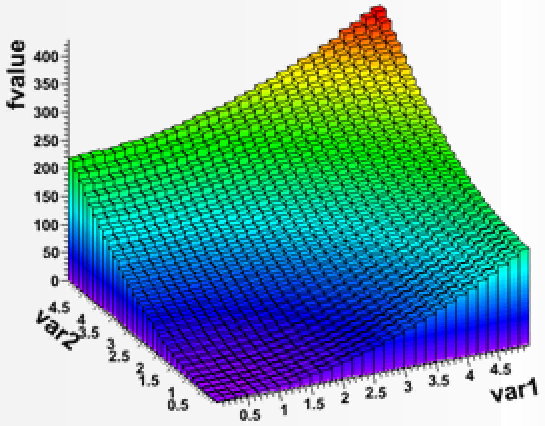
Multiple input variables (predictors)

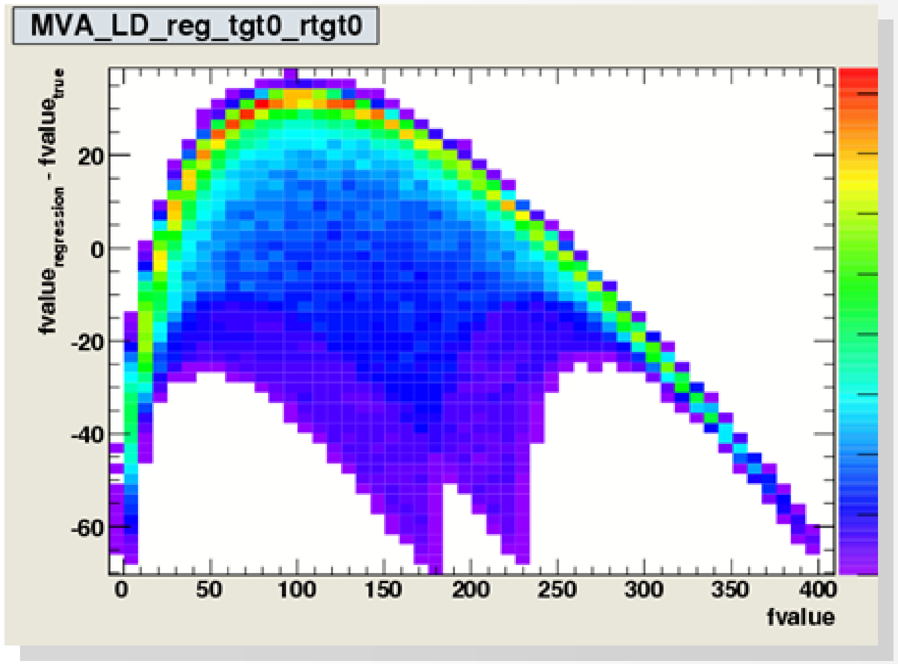
f one
iable(s)

Regression Analysis

- Regression approximates the functional dependence of a target on (x_1,…,x_N)
  - Example: predict the energy correction of jet clusters in calorimeter
- Training: instead of specifying sig/bkgr, provide a regression target
  - Multi-dim target space possible
- Not yet implemented for all methods (so far: LD, PDERS, PDEFoam, MLP, SVM)

Example:
Target as function of 2 input variables

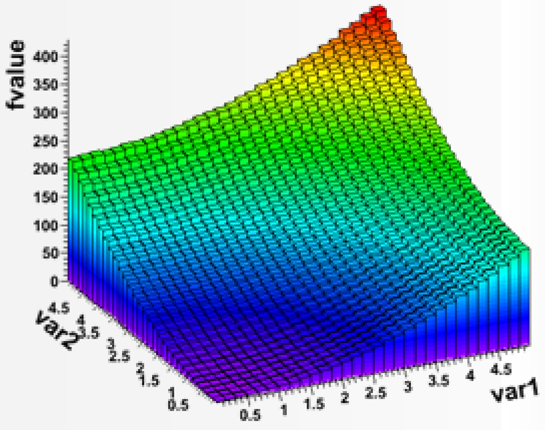


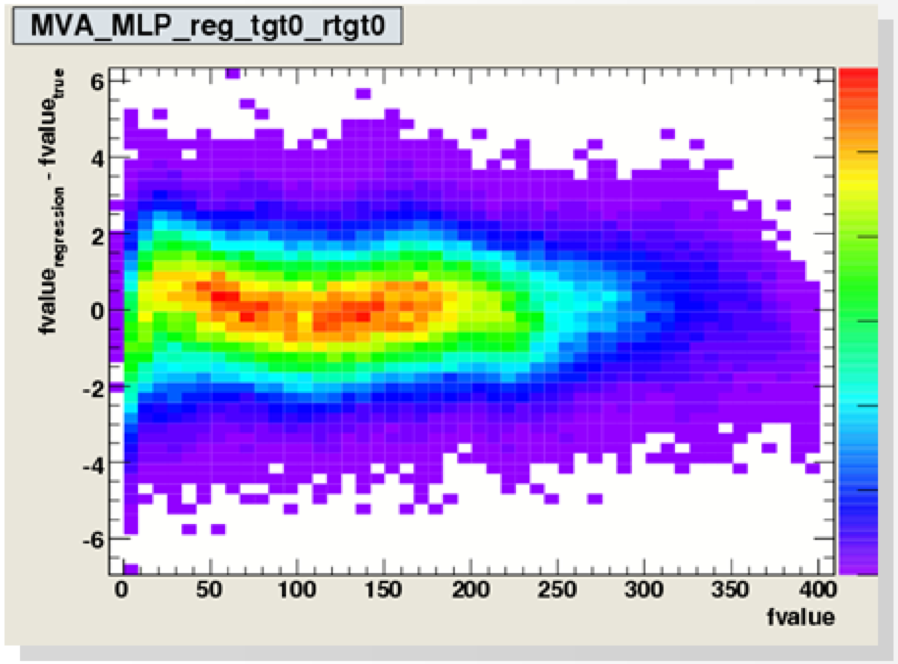


Regression Analysis

- Regression approximates the functional dependence of a target on ($x_1,\ldots,x_N$)
  - Example: predict the energy correction of jet clusters in calorimeter
- Training: instead of specifying sig/bkgr, provide a regression target
  - Multi-dim target space possible
- Not yet implemented for all methods (so far: LD, PDERS, PDEFoam, MLP, SVM)

Example:
Target as function of 2 input variables

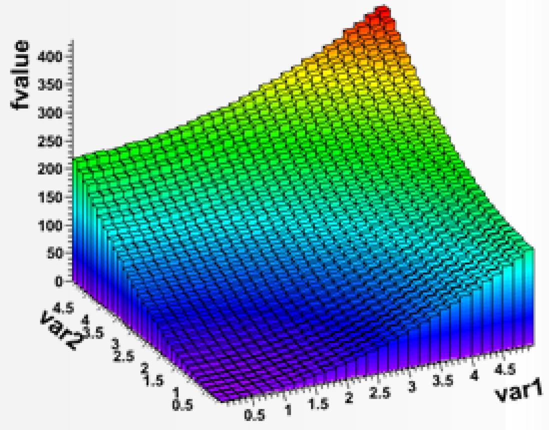


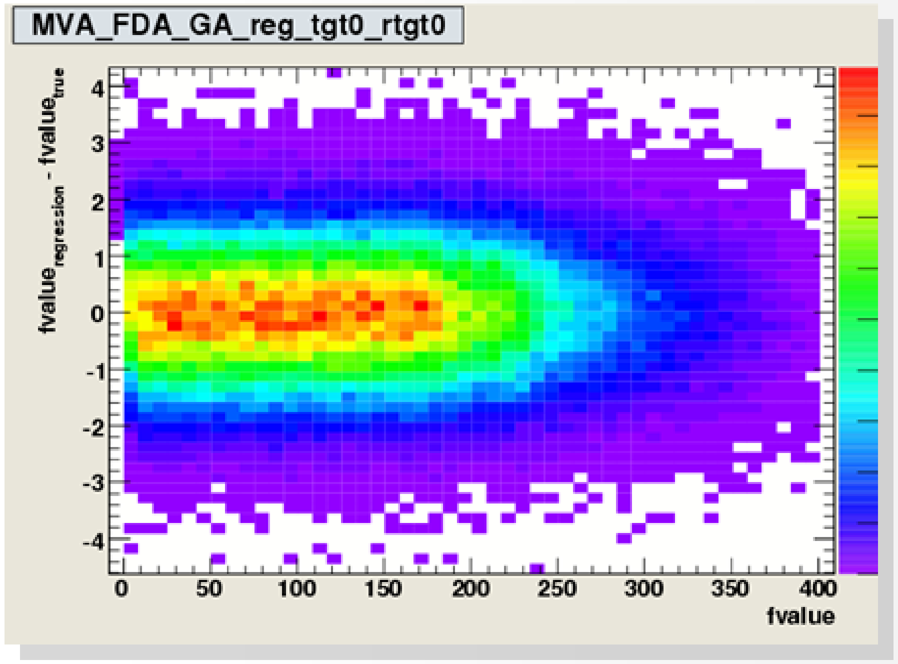


Regression Analysis

- Regression approximates the functional dependence of a target on $(x_1,\ldots,x_N)$
  - Example: predict the energy correction of jet clusters in calorimeter
- Training: instead of specifying sig/bkgr, provide a regression target
  - Multi-dim target space possible
- Not yet implemented for all methods (so far: LD, PDERS, PDEFoam, MLP, SVM)

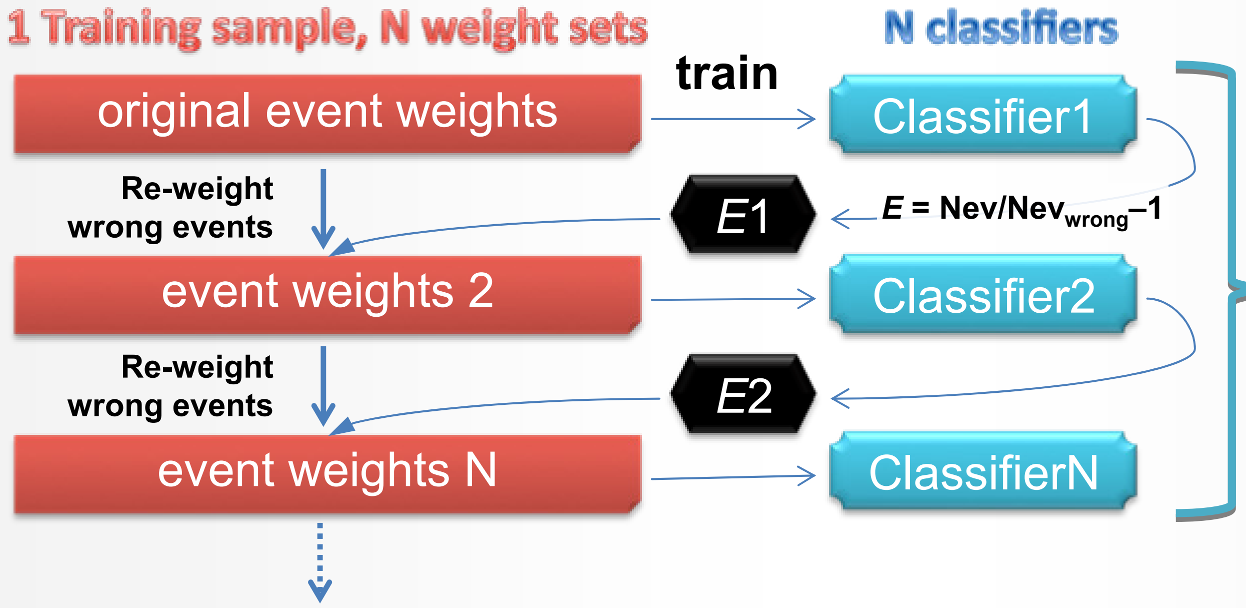
Example:
Target as function of 2 input variables





Generalised Classifier Boosting

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



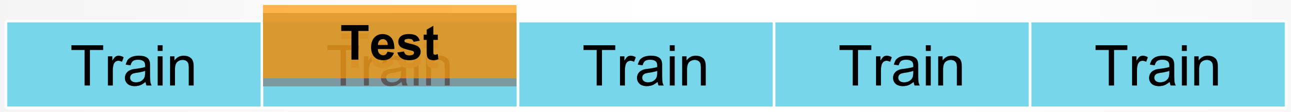
$$y(x) = \sum_{i}^{N_{\text{Classifier}}} \log\left(\frac{1-f_{\text{err}}^{(i)}}{f_{\text{err}}^{(i)}}\right) C^{(i)}(x)$$

Response is weighted sum of each classifier response

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

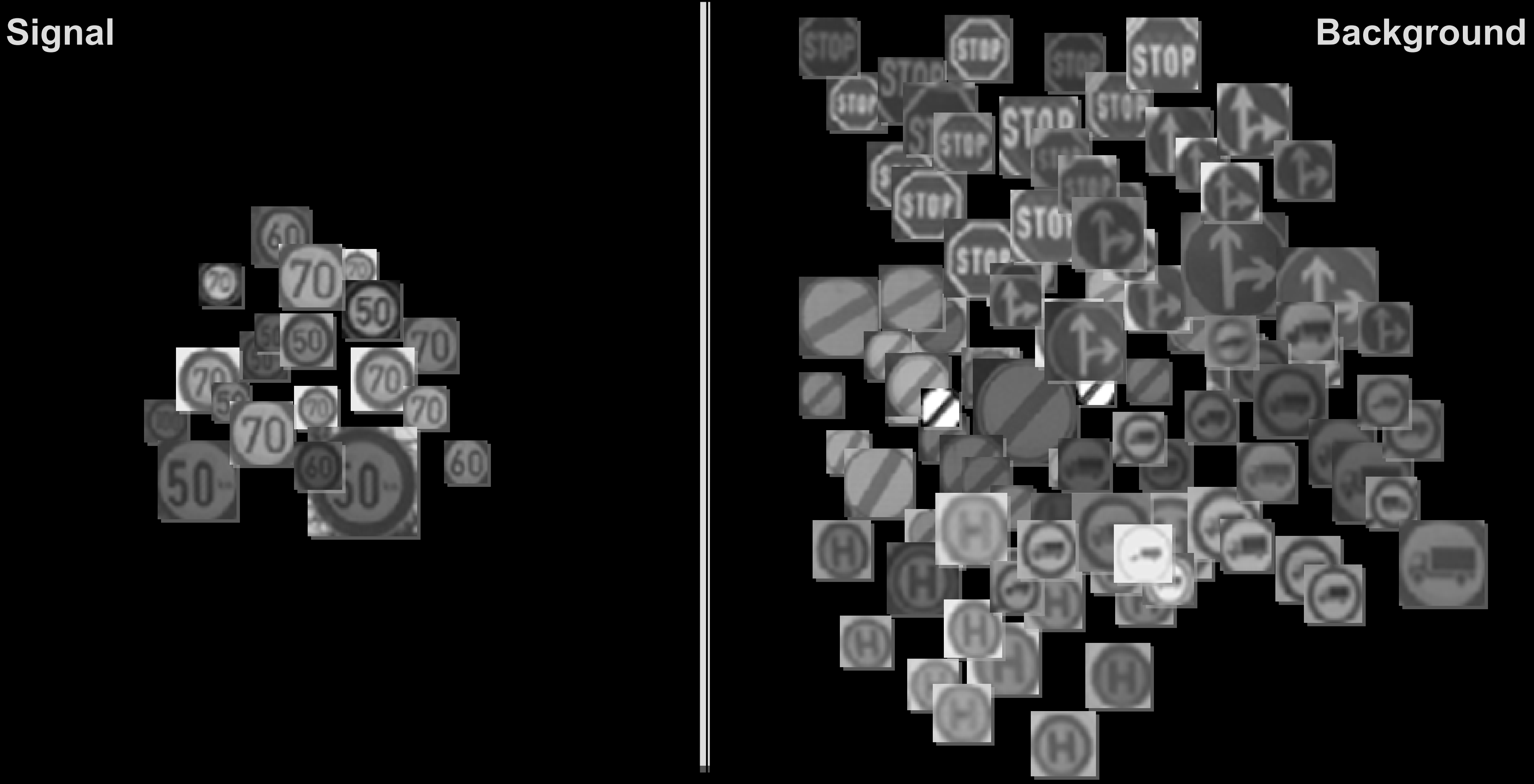
Automated Classifier Tuning via Cross Validation

- Many classifiers have parameters that, being tuned, improves performance
- Method for automated parameter tuning: Cross-Validation
- Special choice of *K*-fold cross-validation:
  - Divide the data sample into *K* sub-sets
  - For set of parameters α train *K* classifiers $C_i(\alpha)$, *i*=1..*K*, omitting each time the *i*-th subset from the training to use as test sample



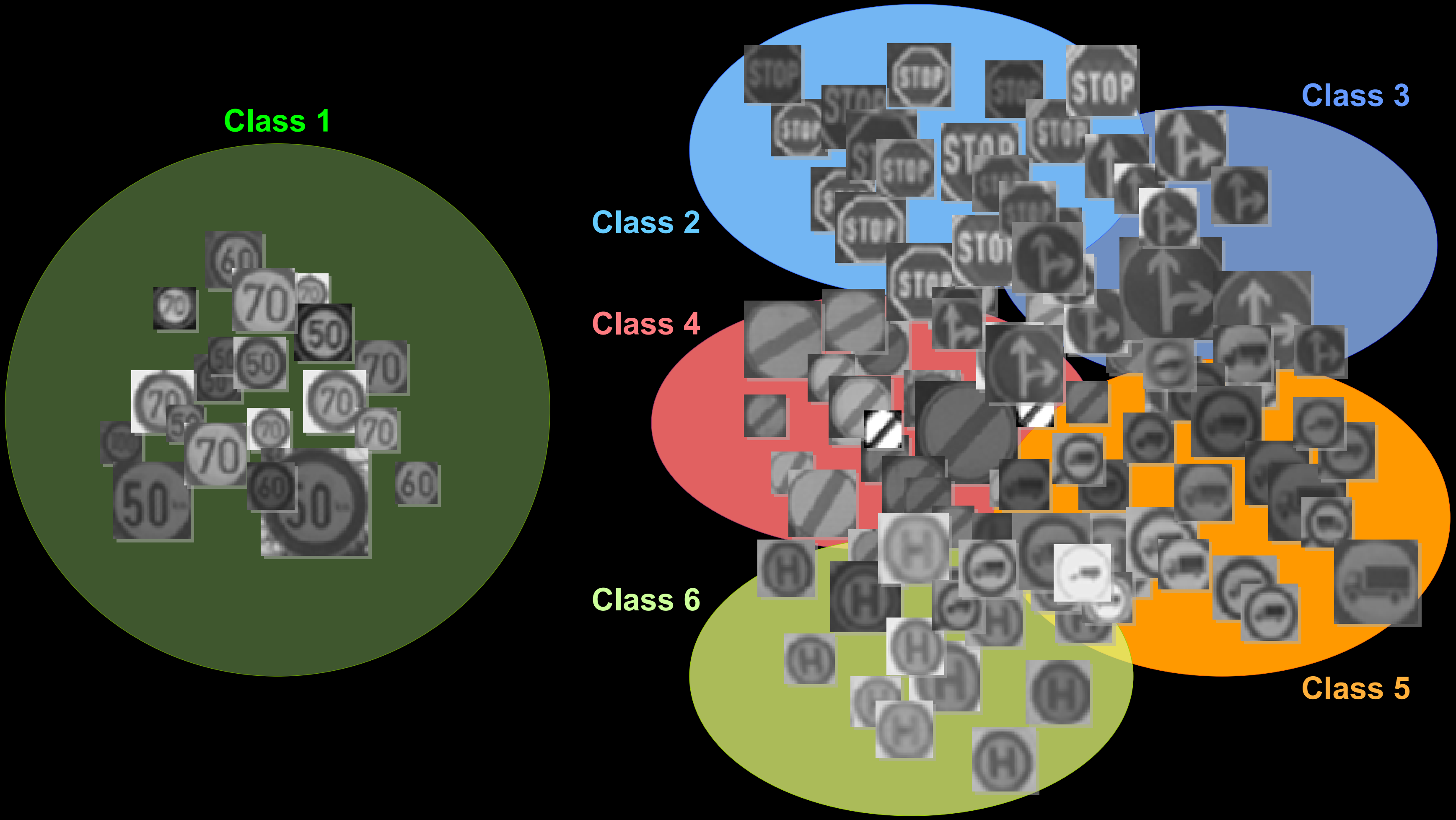
  - Compute performance estimator for each $C_i(\alpha)$ and average among all *K*
  - Choose parameter set α providing the best average estimator

Multi-Class Classification



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

Multi-Class Classification



Multi-class classification – natural extension for many classifiers

*T*MVA 4

- Current stable TMVA version 3.9.6 for ROOT 5.22 (midst of December),
- Moving to TMVA 4. Target for first release: *end of 2008*
  - Not everything at once:
    1) New framework (*done*)
    2) Regression (*PDE, LD, MLP, SVN*) and generic classifier boosting (*done*)
    3) Multi-class classification, automatic classifier tuning (*being prepared*)
    4) Composite classifiers (*not done*)
  - Multi-core some time along the way (*being prepared*)

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 (Cal Tech)
  - Many implementations of individual classifiers exist

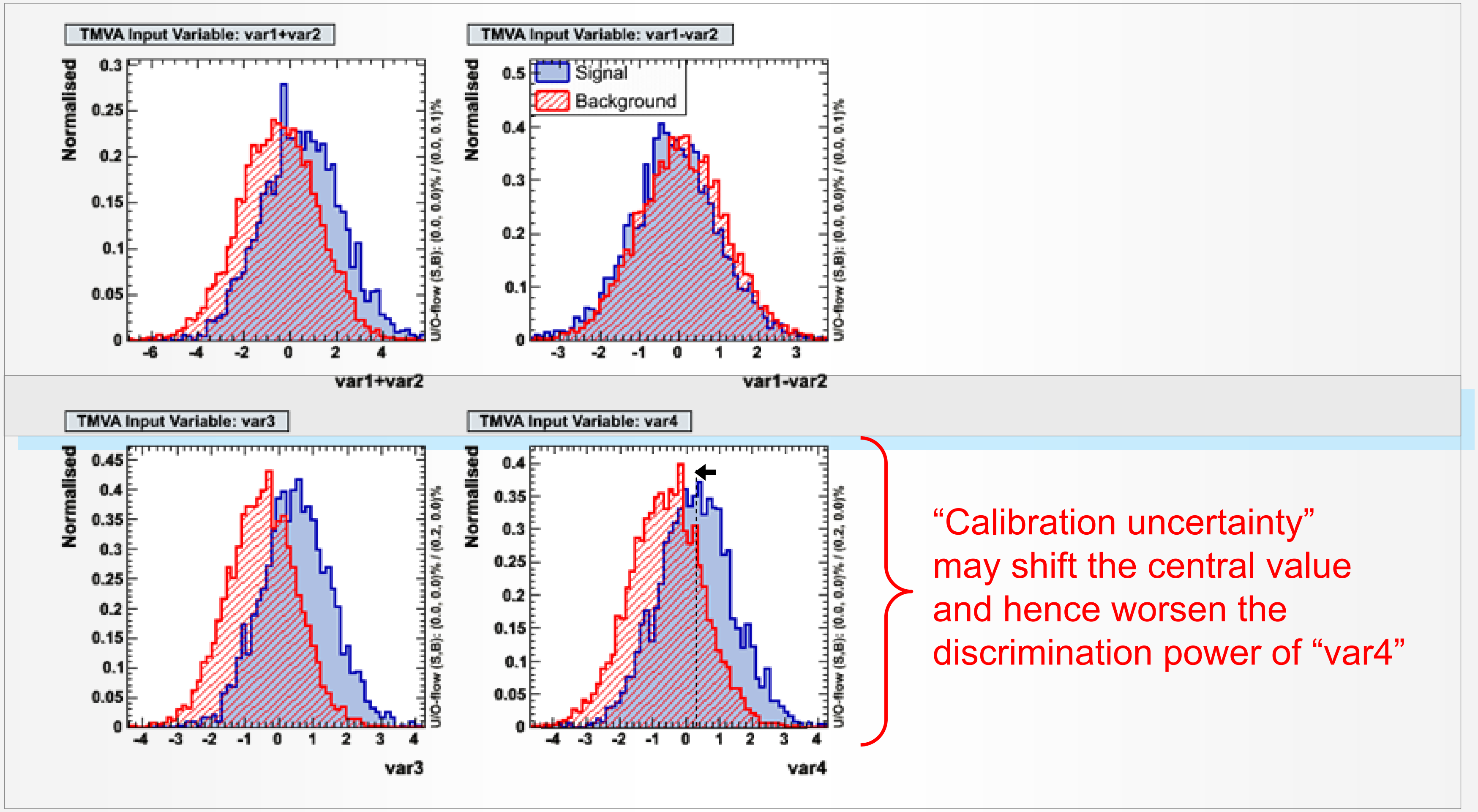
Acknowledgments: The fast development of TMVA would not have been possible without the contribution and feedback from many developers and users to whom we are indebted. We thank in particular the CERN Summer students Matt Jachowski (Stanford) for the implementation of TMVA's new MLP neural network, Yair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

A (brief) Word on

Systematics & Irrelevant Input Variables

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

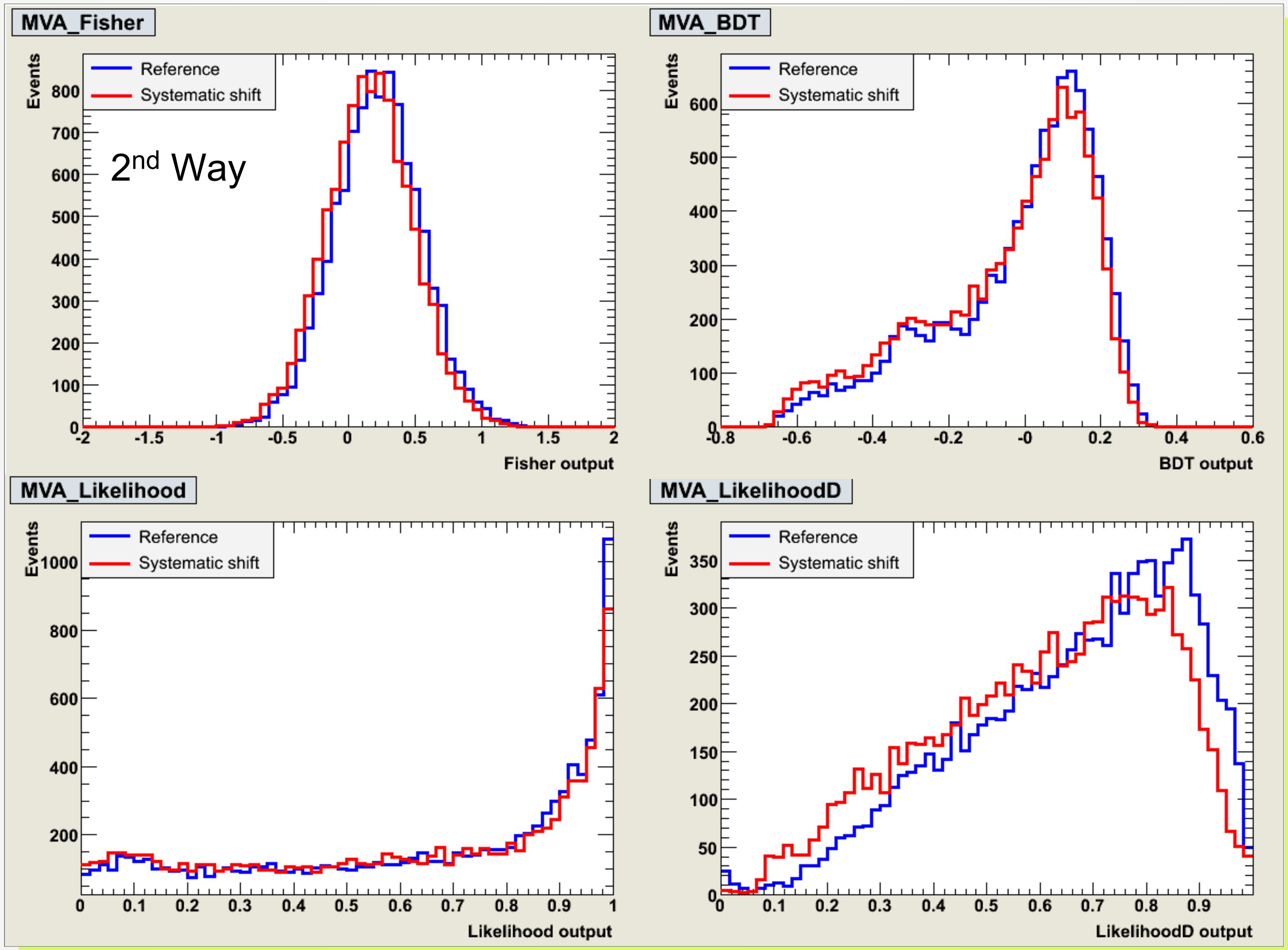
- Assume strongest variable “var4” suffers from systematic uncertainty
- (at least) Two ways to deal with it:

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) !

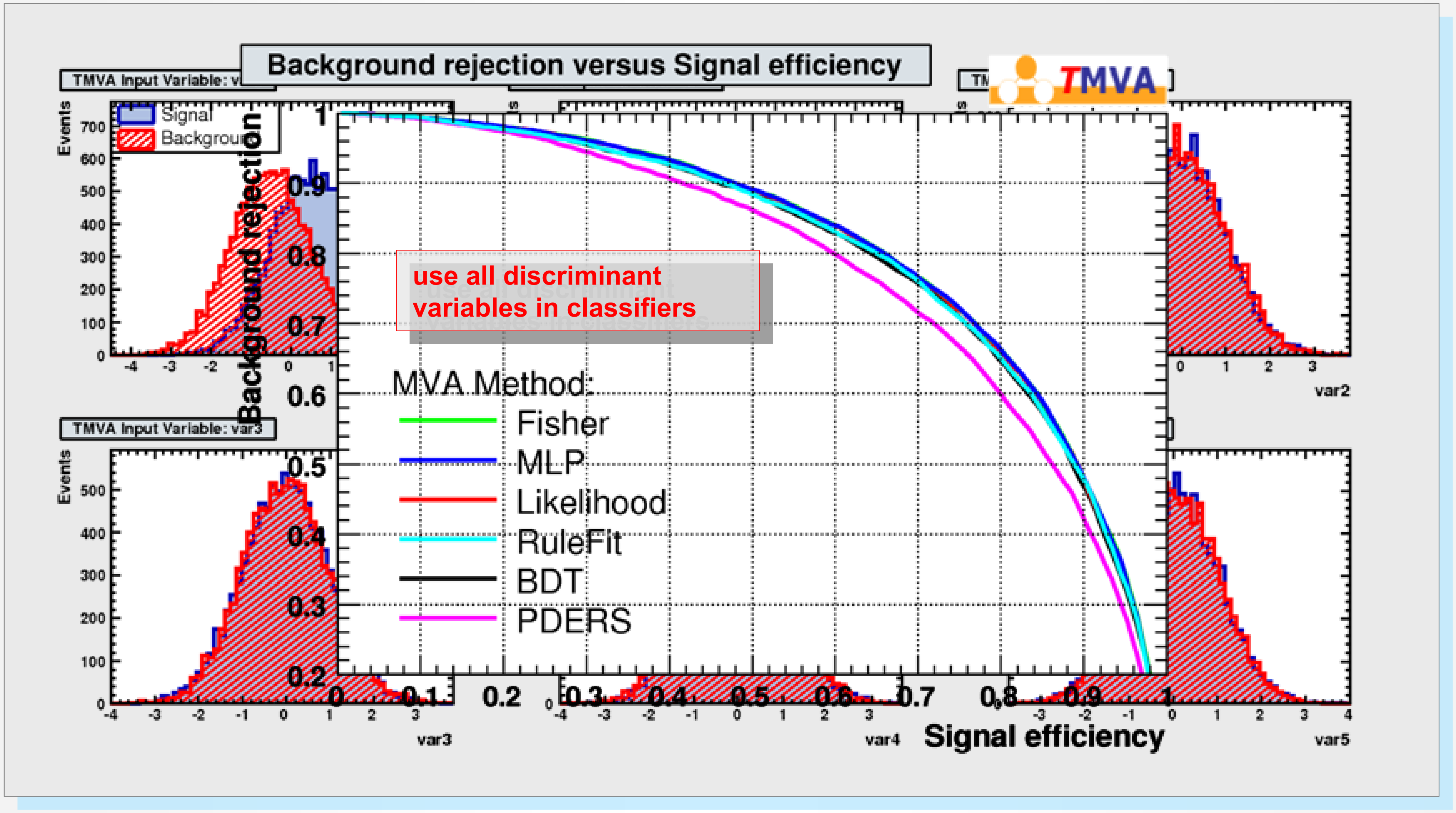
Treatment of Systematic Uncertainties

Classifier output distributions for signal only



Stability with Respect to Irrelevant Variables

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


Background rejection versus Signal efficiency
TMVA
TMVA Input Variable: var3
Signal
Background
Events
Background rejection
Signal efficiency
use all discriminant variables in classifiers
MVA Method:
Fisher
MLP
Likelihood
RuleFit
BDT
PDERS
var2
var3
var4
var5