

Multivariate Discriminators

INFN School of Statistics 2015
Ischia (Napoli, Italy)

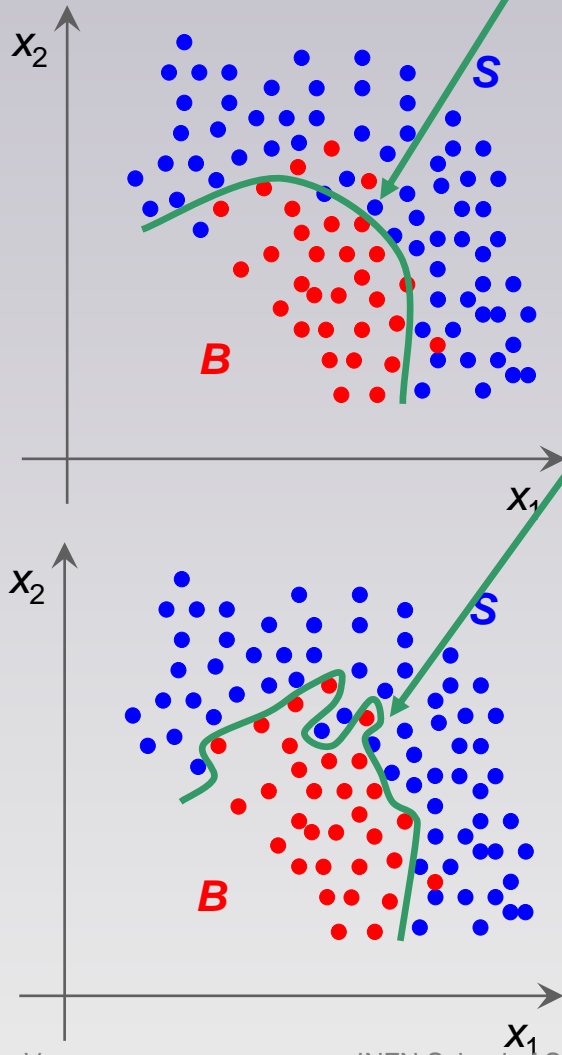
Helge Voss



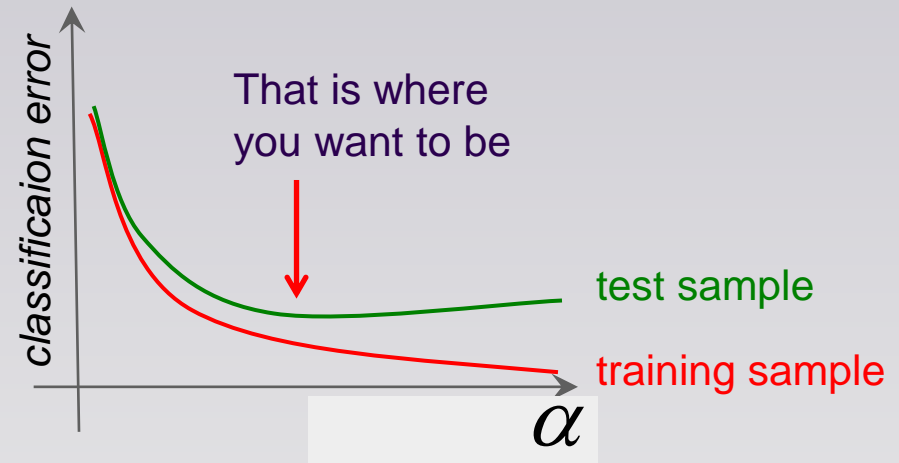
MAX-PLANCK-INSTITUT FÜR KERNPHYSIK IN HEIDELBERG

Overtraining

- it seems intuitive that this boundary will give better results on another statistically independent data set than that one



- e.g. stop training before you learn statistical fluctuations in the data
- Include 'regularisation' in the model
- verify on independent "test" sample



- overtraining is concern for every "tunable parameter" α of classifiers: Smoothing parameter, n-nodes...

Cross Validation

- classifiers have tuning parameters “ α ” \rightarrow choose and control performance
 - #training cycles, #nodes, #layers, regularisation parameter (neural net)
 - smoothing parameter h (kernel density estimator)
 -
- more training data \rightarrow better training results
- division of data set into “training” and “test” and “validation” sample? ☹

Cross Validation: divide the data sample into say 5 sub-sets

Train

Train

Train

Train

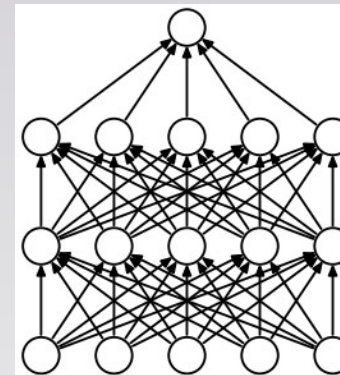
Test

- train 5 classifiers: $y_i(x, \alpha) : i=1, \dots, 5$,
- classifier $y_i(x, \alpha)$ is trained without the i -th sub sample
- calculate the test error: $CV(\alpha) = \frac{1}{N_{\text{events}}} \sum_k^{\text{events}} L(y_i(x_k, \alpha))$ L : loss function
- choose tuning parameter α for which $CV(\alpha)$ is minimum and train the final classifier using all data

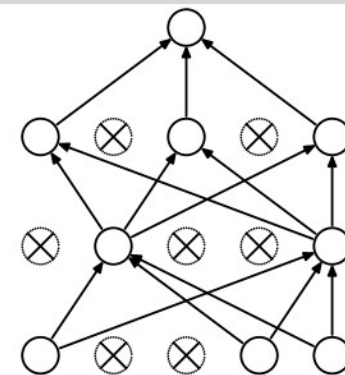
Neural Network Regularisation

How to control model complexity in Neural Networks:

- #nodes and # layers
- early stopping → very first (old) ‘regularizer’ used
 - Start with small random weights → sigmoid approximately linear → essentially a linear model → stop before it deviates too much from that
- Weight decay:
 - add ‘regularizing’ term to the loss function $L = L + \frac{1}{2} \sum w^2$
 - Favour small weights → i.e. simpler models
- Dropout
 - Randomly remove nodes during each training step
 - Essentially a large model averaging procedure like ‘bagging’



(a) Standard Neural Net



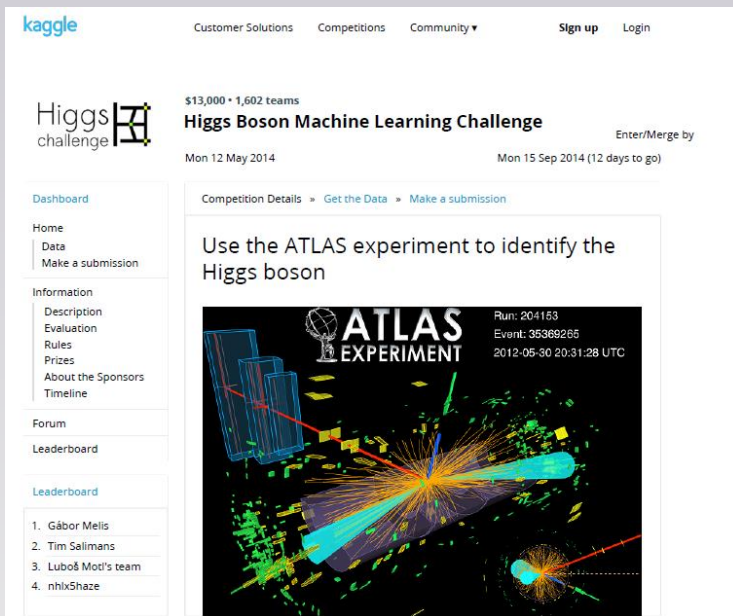
(b) After applying dropout.

What is the best Network Architecture?

We (TMVA) used to say:

- Typically in high-energy physics, non-linearities are reasonably simple,
 - 1 layer with a larger number of nodes probably enough
 - still worth trying 2 (3?) layers (and less nodes in each layer)

But Higgs ML Challenge:



The screenshot shows the Kaggle website for the Higgs Boson Machine Learning Challenge. The page includes a navigation bar with 'Customer Solutions', 'Competitions', 'Community', 'Sign up', and 'Login'. The challenge title is 'Higgs Boson Machine Learning Challenge' with a prize of '\$13,000 • 1,602 teams'. It started on 'Mon 12 May 2014' and ends on 'Mon 15 Sep 2014 (12 days to go)'. The main content area features a description: 'Use the ATLAS experiment to identify the Higgs boson' and a visualization of the ATLAS experiment detector. A sidebar on the left contains a 'Dashboard' with links to 'Home', 'Data', 'Make a submission', 'Information', 'Forum', and 'Leaderboard'. The 'Leaderboard' section lists the top four teams: 1. Gábor Melis, 2. Tim Salimans, 3. Luboš Motl's team, and 4. nhixShaze.

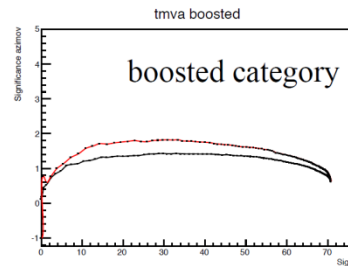
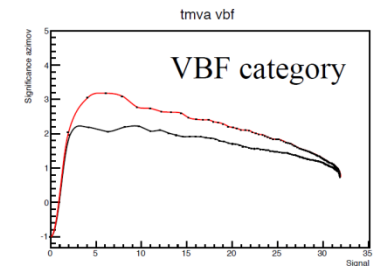
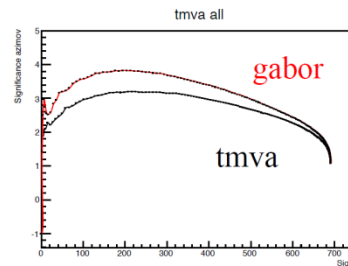
- Won by a 'Deep Neural Network'
 - well... 3 hidden layers with 600 nodes each!
- Did it win because if 'learned' categories, of different event classes (i.e 1 or 3-prong tau decays e.t.c) that noone seemed to look at explicitly?
- Or was it just "collecting efficiency" in corners that requires highly flexible classifiers, hence VERY careful tuning using extensive cross validation?

David Rousseau tried to answer this: BUT:
 TMVA was untuned ! And Gabor spent MANY MANY hours tuning !!
 → that is more than just an ‘unfair’ comparison.

Compare figure of merits:
 (larger is better)

TMVA untuned:	3
TMVA tuned (Eckhard)	3.6
XGBoost tuned	3.7
Gabor	3.8

TMVA vs Gabor



- vbf, boosted categories as is ATLAS note (no ATLAS insider information)
- tmva, gabor are trained without categories, on full 30 variables (not directly comparable to ATLAS analysis)
- (also significance is simple asimov, no bin, no systematics (and fake tau missing))
- Gabor improves more significantly in VBF categories (2 jets → events more complex)

David Rousseau HiggsML visits CERN, 19th May 2015

Overview

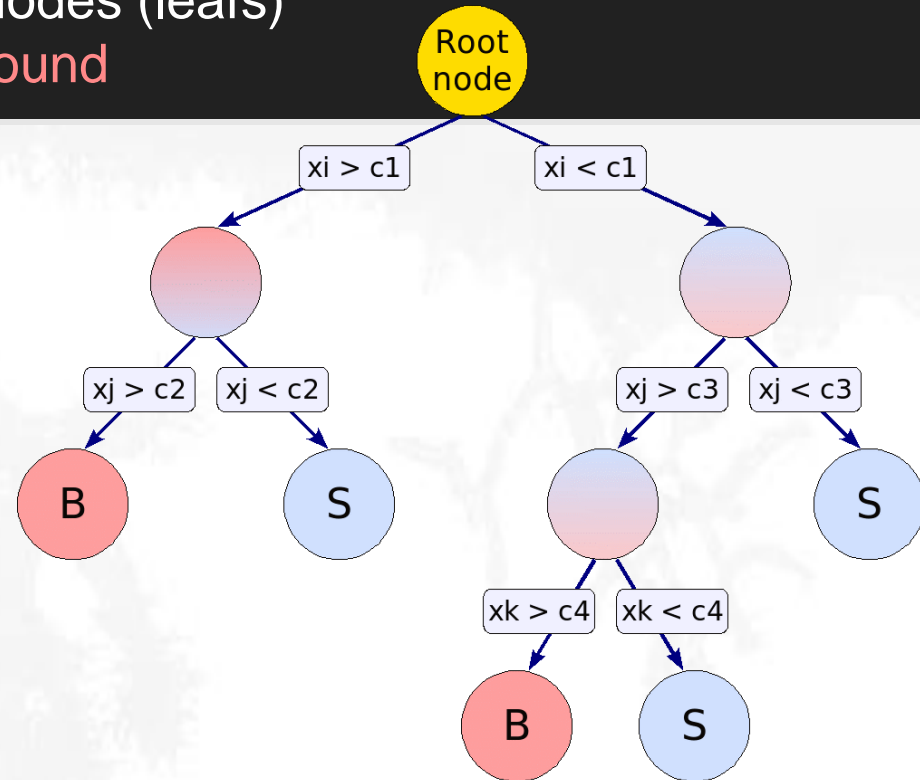
- **Multivariate classification/regression algorithms (MVA)**
 - what they are
 - how they work
- **Overview over some classifiers**
 - **Multidimensional Likelihood (kNN : k-Nearest Neighbour)**
 - **Projective Likelihood (naïve Bayes)**
 - **Linear Classifier**
 - **Non linear Classifiers**
 - Neural Networks
 - **Boosted Decision Trees**
 - Support Vector Machines
- **General comments about:**
 - **Overtraining**
 - **Systematic errors**

Boosted Decision Trees

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

- Each branch \rightarrow one standard “cut” sequence
 - easy to interpret, visualised
 - independent of monotonous variable transformations, immune against outliers
 - weak variables are ignored (and don't (much) deteriorate performance)
- Disadvantage \rightarrow very sensitive to statistical fluctuations in training data

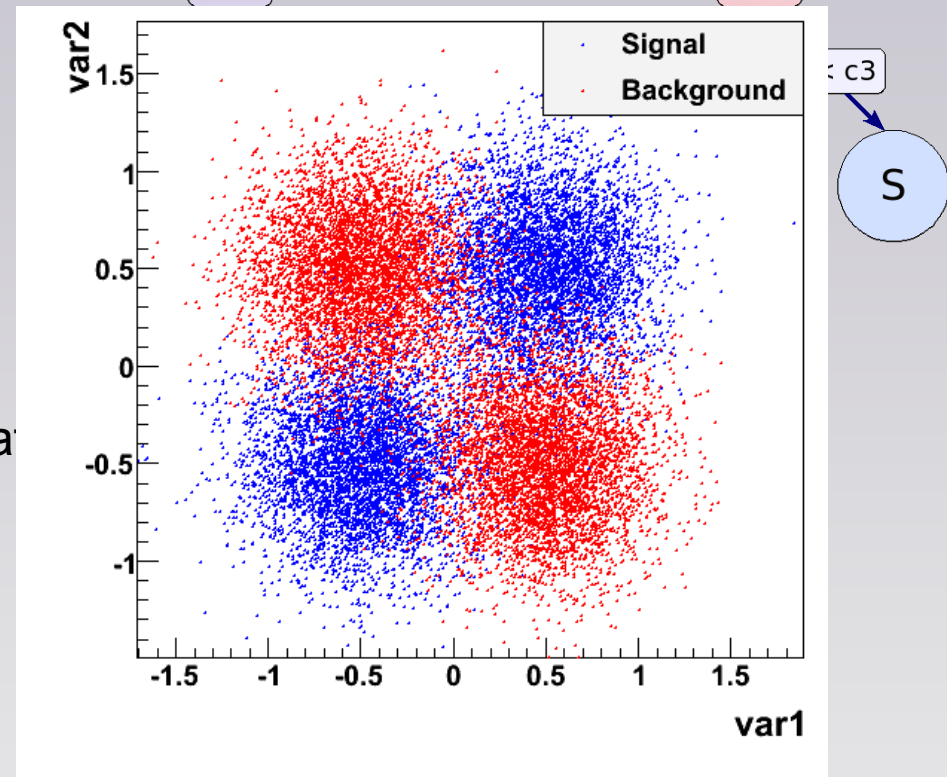
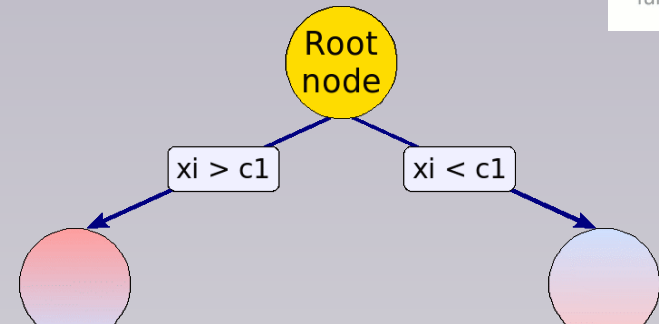
- Boosted Decision Trees (1996): combine a whole forest of Decision Trees, derived from the same sample, e.g. using different event weights.
 - overcomes the stability problem
 - increases performance



\rightarrow became popular in HEP since MiniBooNE, B.Roe et.a., NIM 543(2005)

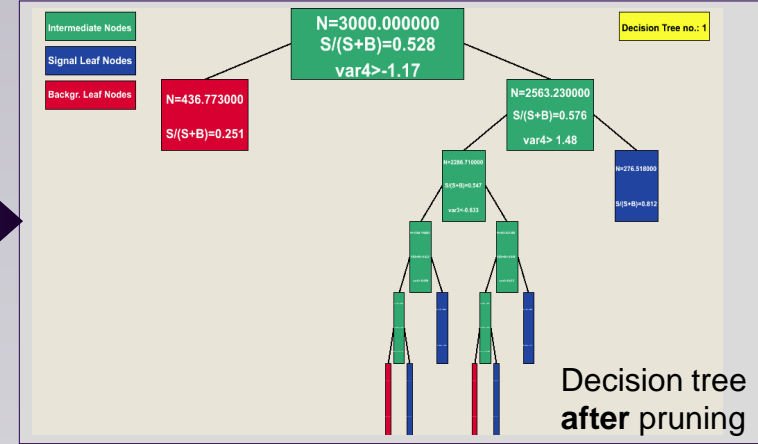
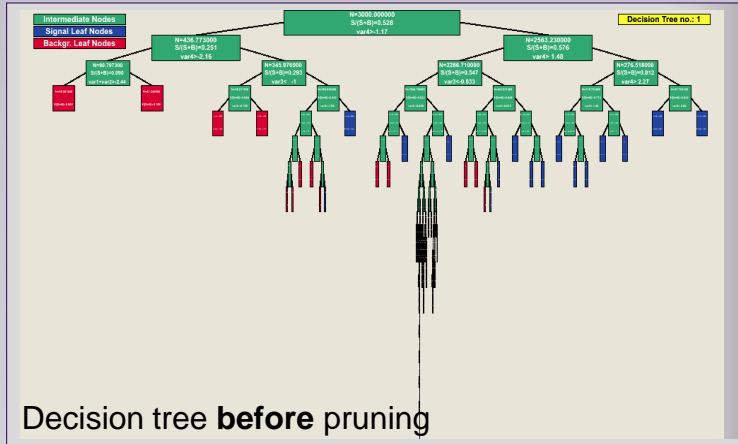
Growing a Decision Tree

- start with training sample at the root node
- split training sample at node into two, using a cut in the one variable that gives best separation gain
- continue splitting until:
 - minimal #events per node
 - maximum number of nodes
 - maximum depth specified
 - (a split doesn't give a minimum separation gain) → not a good idea → see “chessboard”
- leaf-nodes classify **S**, **B** according to the majority of events or give a **S/B** probability



Decision Tree Pruning

- Individual decision trees: Grown to large size and then pruned!

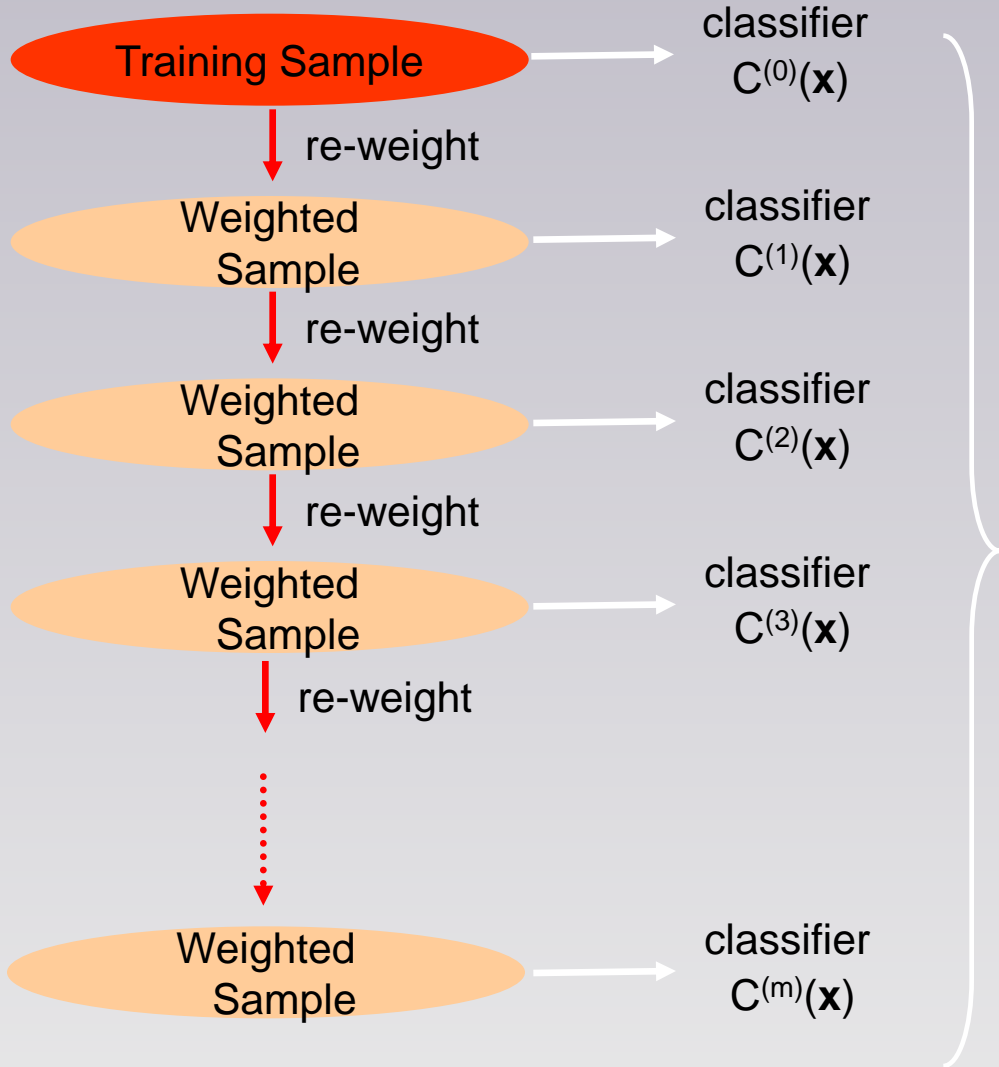


- Pruning algorithms are developed and applied on individual trees
 - optimally pruned single trees are not necessarily optimal in a forest !
- Boosted Decision Trees are better limited in growth size (max depth) straight way
 - optimal depth: “typically small $O(<5)$ depending on ‘interaction’ between variables, i.e. terms in ANOVA ‘expansion’):

$$\eta(X) = \sum_i \eta_i(X_i) + \sum_{ij} \eta_{ij}(X_i, X_j) + \sum_{ijk} \eta_{ijk}(X_i, X_j, X_k) + \dots$$
 with $\eta(X)$ begin the target (discriminating) function

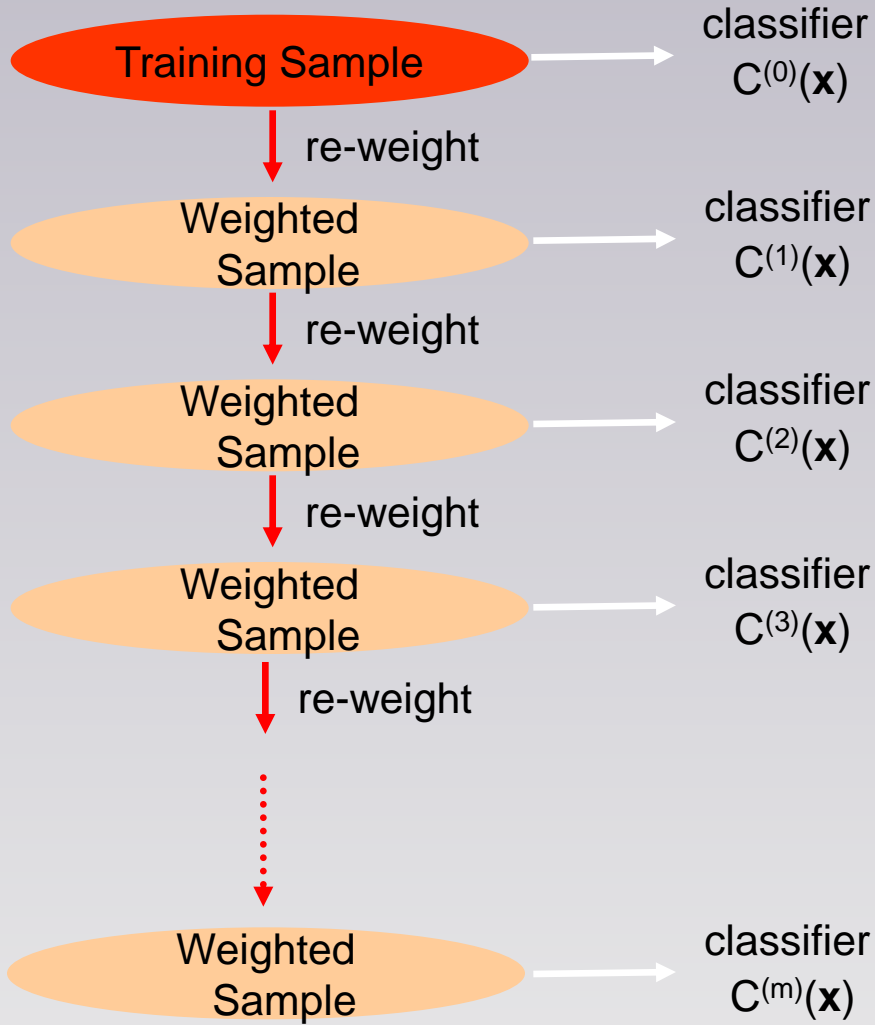
XGBoost apparently has some ‘new’ regularizing scheme → probably worth looking at

Boosting



$$y(\mathbf{x}) = \sum_i^{N_{\text{Classifier}}} w_i C^{(i)}(\mathbf{x})$$

Adaptive Boosting (AdaBoost)



- AdaBoost re-weights events misclassified by previous classifier by:

$$\frac{1 - f_{\text{err}}}{f_{\text{err}}} \quad \text{with :}$$

$$f_{\text{err}} = \frac{\text{misclassified events}}{\text{all events}}$$

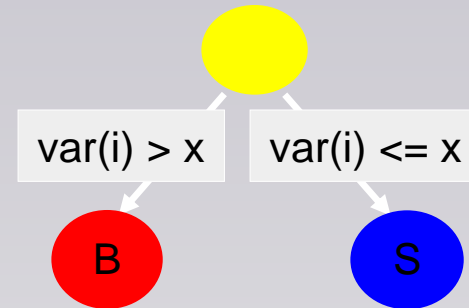
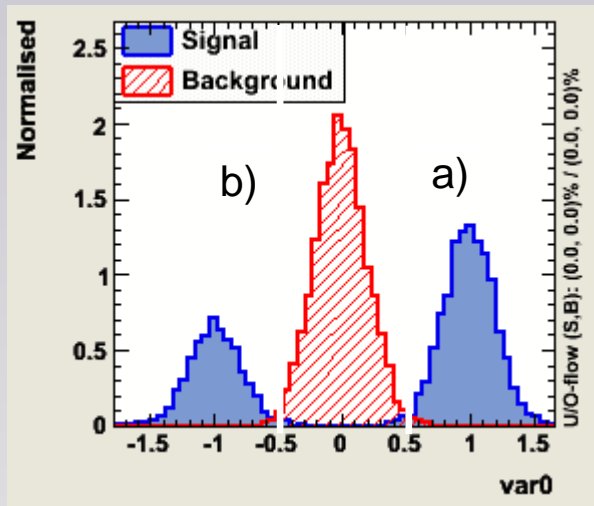
- AdaBoost weights the classifiers also using the error rate of the individual classifier according to:

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

AdaBoost: A simple demonstration

The example: (somewhat artificial...but nice for demonstration) :

- Data file with three “bumps”
- Weak classifier (i.e. one single simple “cut” ↔ decision tree stumps)



- a) $\text{Var0} > 0.5 \rightarrow \epsilon_{\text{sig}}=66\% \epsilon_{\text{bkg}} \approx 0\%$ misclassified events in total 16.5%
or
b) $\text{Var0} < -0.5 \rightarrow \epsilon_{\text{sig}}=33\% \epsilon_{\text{bkg}} \approx 0\%$ misclassified events in total 33%

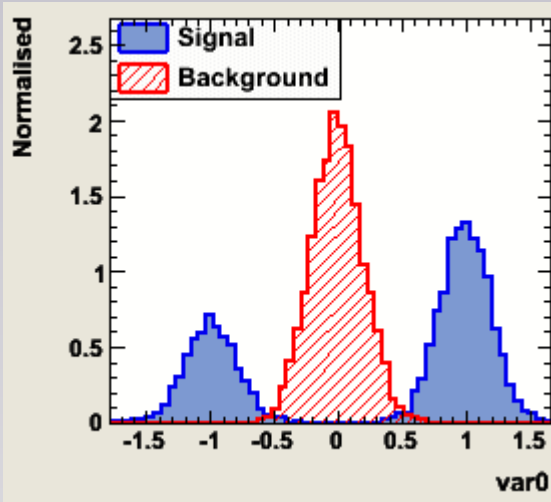
the training of a single decision tree stump will find “cut a)”

AdaBoost: A simple demonstration

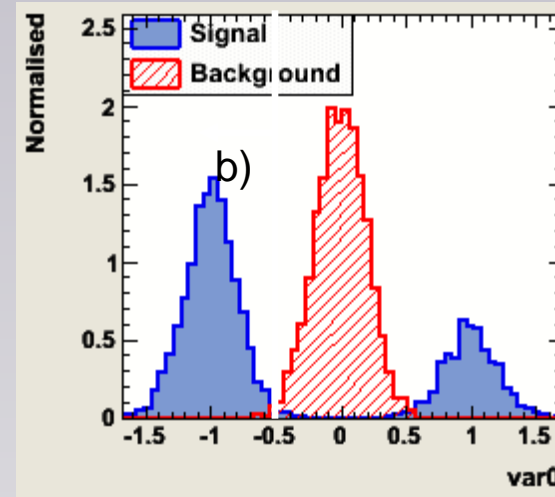
The first “tree”, choosing cut a) will give an error fraction: $\text{err} = 0.165$

→ before building the next “tree”: weight wrong classified training events by $(1 - \text{err}/\text{err}) \approx 5$

→ the next “tree” sees essentially the following data sample:

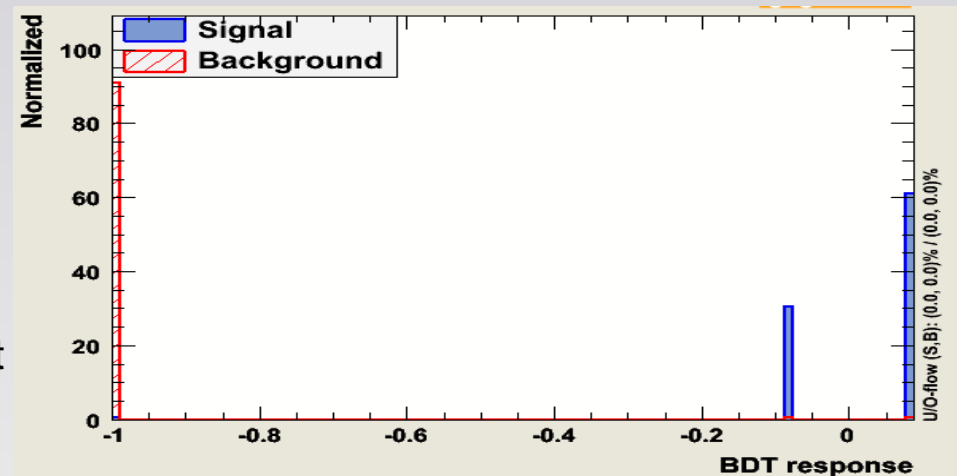


re-weight



.. and hence will
chose: “cut b)”:
 $\text{Var0} < -0.5$

The combined classifier: Tree1 + Tree2
the (weighted) average of the response to
a test event from both trees is able to
separate signal from background as
good as one would expect from the most
powerful classifier



“A Statistical View of Boosting” (Friedman 1998 et.al)

Boosted Decision Trees: two different interpretations

1. give events that are “difficult to categorize” more weight and average afterwards the results of all classifiers that were obtained with different weights
2. see each Tree as a “base classifier” of an additive classifier ensemble

$$y(x; \alpha, \theta) = \sum_i^{nTrees} \alpha_i T_i(x; \theta)$$

→ boosting: “greedy” (*i.e.* each step just adds a new classifier to the ensemble w/o modifying the current ensemble) optimization of $y(x; \alpha, \theta)$ w.r.t a specific ‘loss function’

→ AdaBoost: “exponential loss function” $L = \exp(-y^{train} y(x; \alpha, \theta))$ where $y^{train} = 1$ (signal), $y^{train} = -1$ (backgr.)

→ Optimizes ‘log odds’, *i.e.* $y^{best}(x) = \frac{1}{2} \log \frac{P(S|x)}{P(B|x)}$

$$E[L] = E[e^{-yy(x)}] = P(S|x)e^{-y(x)} + P(B|x)e^{y(x)}; \frac{dE[L]}{dy(x)} = -P(S|x)e^{-y(x)} + P(B|x)e^{y(x)} \equiv 0 \rightarrow y(x) = \frac{1}{2} \log \frac{P(S|x)}{P(B|x)};$$

Gradient Boost

- AdaBoost: Exponential loss $L = \exp(-y^{train}y(x; \alpha, \theta)) \rightarrow$ sensitive to outliers
- Gradient Boost:
 - implement “boosting” with arbitrary “loss functions” by approximating the gradient of the loss function
 - Which loss function actually gives us the proper ‘bernulli probability’ distribution for the fitted P(S)
 - Binomial log-likelihood loss $\ln(1 + \exp(-2y^{train}y(\alpha, x))) \rightarrow$ more well behaved loss function, (the corresponding “GradientBoost” is implemented in TMVA)

Bagging and Randomised Trees

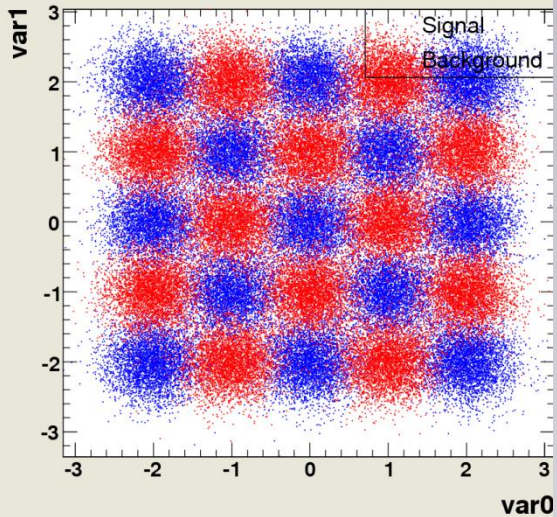
- Bagging:
 - combine trees grown from “bootstrap” samples
(i.e re-sample training data with replacement)

- Randomised Trees: (Random Forest: trademark L.Breiman, A.Cutler)
 - combine trees grown with:
 - random bootstrap (or subsets) of the training data only
 - consider at each node only a random subsets of variables for the split
 - NO Pruning (despite possibly larger trees than AdaBoost) !

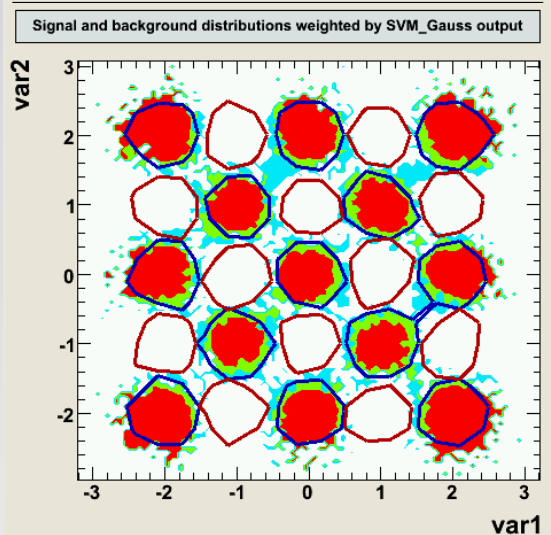
- or any “combination” of Bagging/Randomising/Boosting
- These combined classifiers work surprisingly well, are very stable and almost perfect “out of the box” classifiers

A 'Chess Board' Toy MC

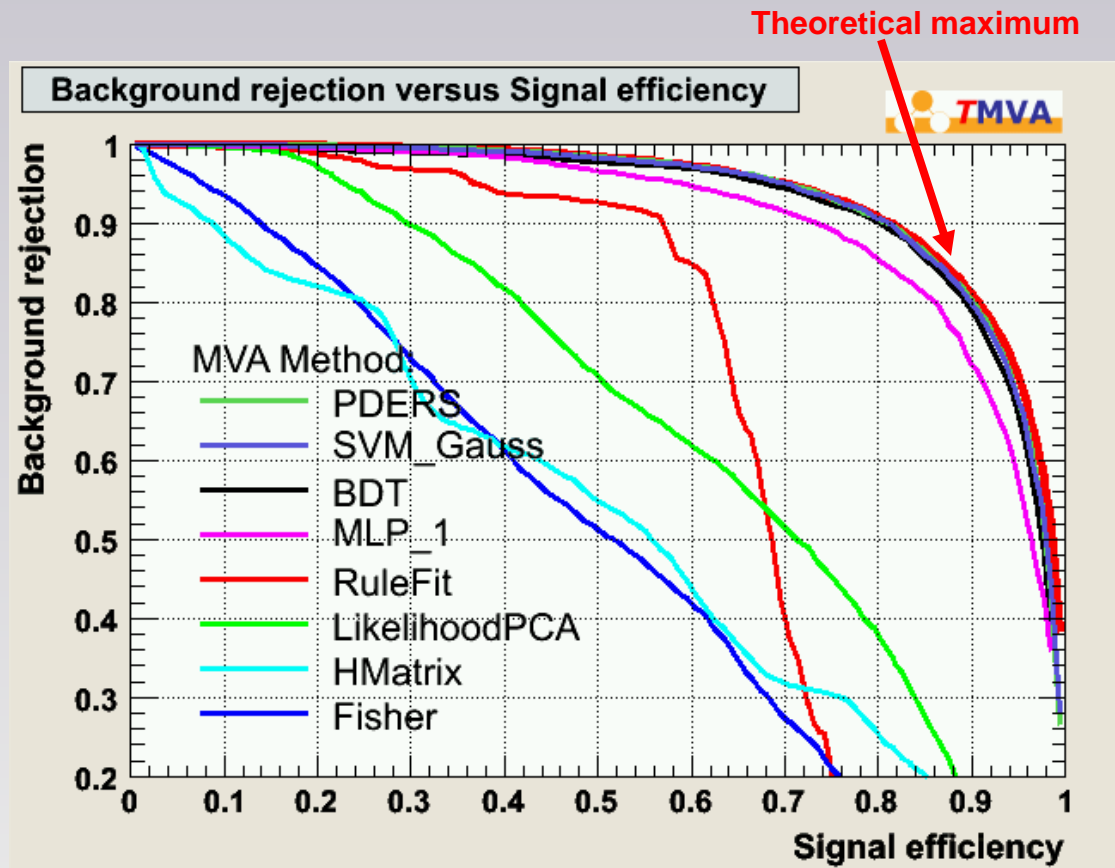
Event Distribution



Events weighted by SVM response

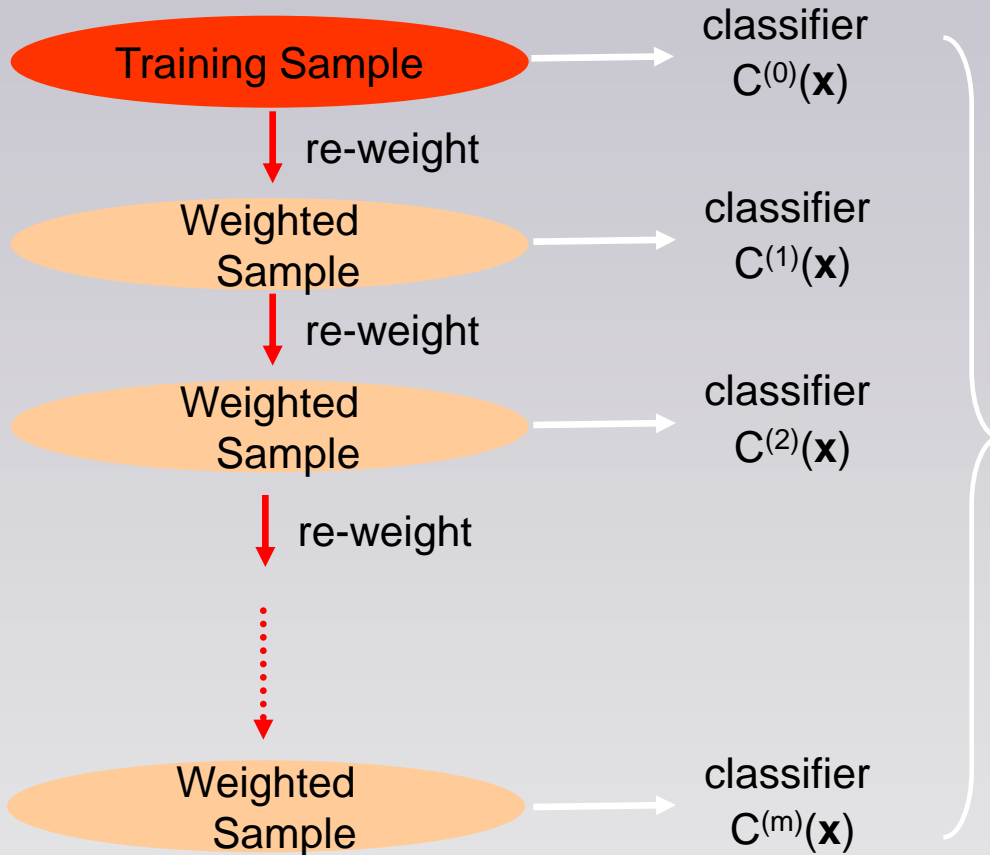


- Performance achieved without parameter adjustments: nearest Neighbour and BDTs are best “out of the box”
- After some parameter tuning, also SVM und ANN(MLP) perform



Generalised Classifier Boosting

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



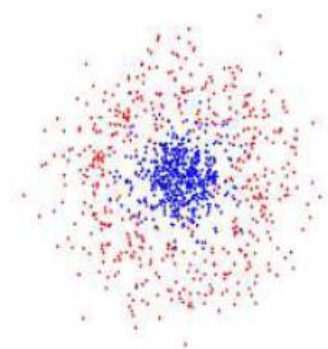
$$y(x) = \sum_i^{N_{\text{Classifier}}} \log \left(\frac{1 - f_{\text{err}}^{(i)}}{f_{\text{err}}^{(i)}} \right) C^{(i)}(x)$$

Response is weighted sum
of each classifier response

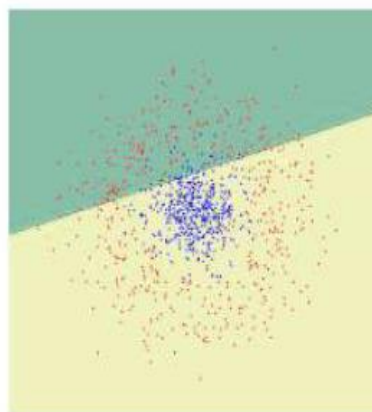
AdaBoost On a linear Classifier (e.g. Fisher)

J. Sochman, J. Matas, `cmp.felk.cvut.cz`

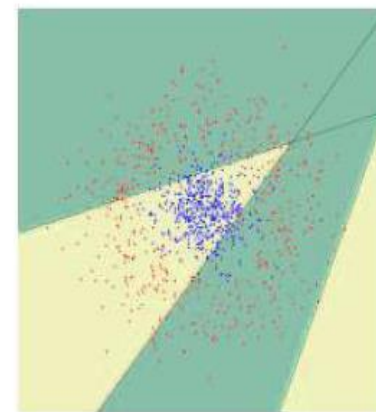
Start with a problem for which a linear classifier is weak:



$t = 1$

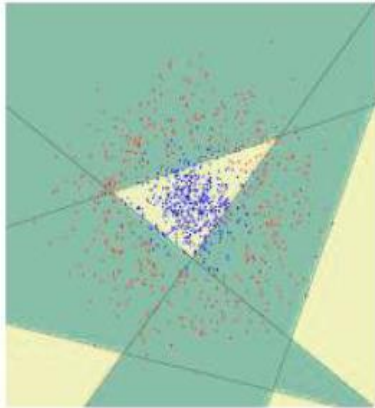


$t = 3$

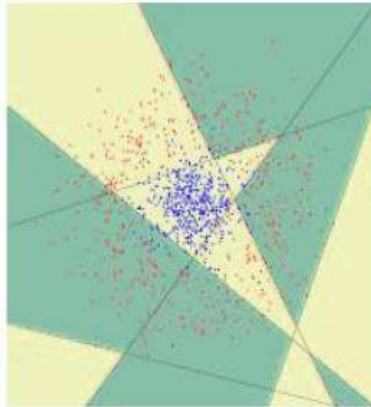


AdaBoost On a linear Classifier (e.g. Fisher)

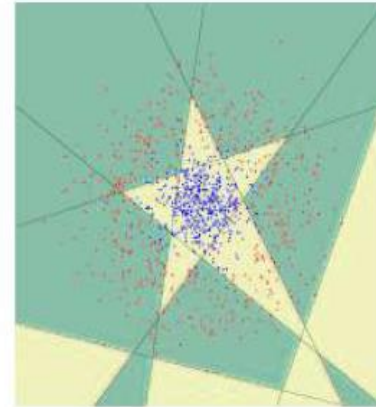
$t = 5$



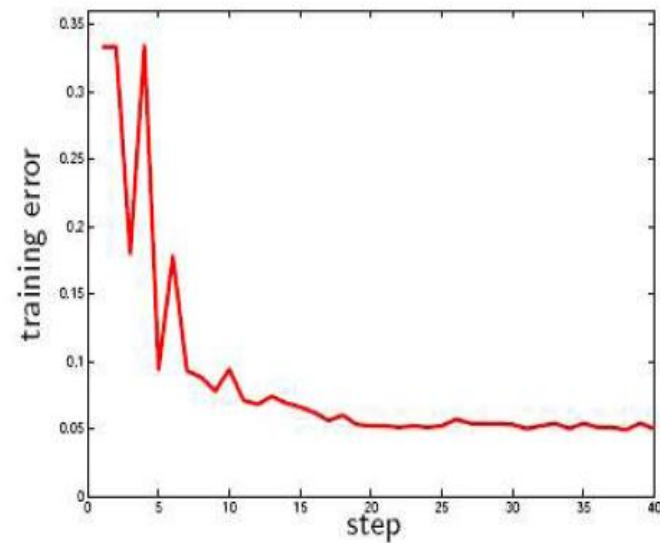
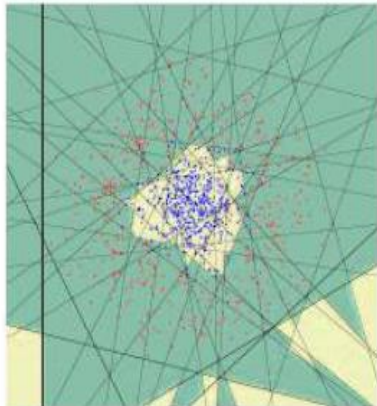
$t = 6$



$t = 7$



$t = 40$



Support Vector Machines

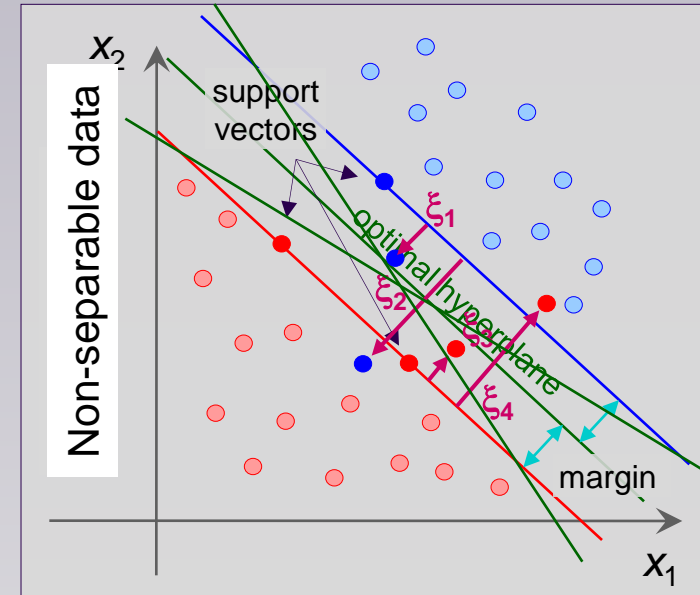
- Neural Networks are complicated by finding the proper optimum “weights” for best separation power by “wiggly” functional behaviour of the piecewise defined separating hyperplane
- KNN (multidimensional likelihood) suffers disadvantage that calculating the MVA-output of each test event involves evaluation of ALL training events
- If Boosted Decision Trees in theory are always weaker than a perfect Neural Network

Support Vector Machine

- There are methods to create linear decision boundaries using only measures of distances (= inner (scalar) products)
 - → leads to quadratic optimisation problem
- The decision boundary in the end is defined only by training events that are closest to the boundary
- suitable variable transformations into a higher dimensional space may allow separation with linear decision boundaries non linear problems
- → Support Vector Machine

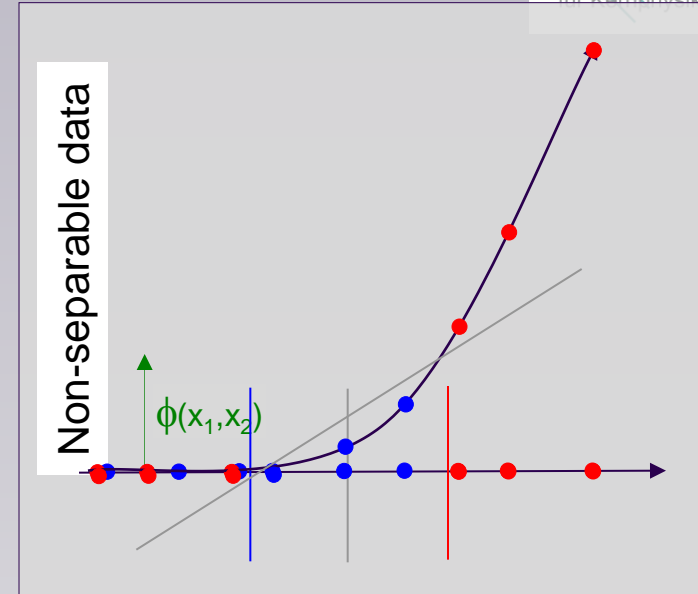
Support Vector Machines

- hyperplane that separates **S** from **B**
 - Linear decision boundary
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - If data non-separable add *misclassification cost* parameter $C \cdot \sum \xi_j$ to minimisation function
 - **Solution of largest margin depends only on inner product of support vectors (distances)**
- quadratic minimisation problem



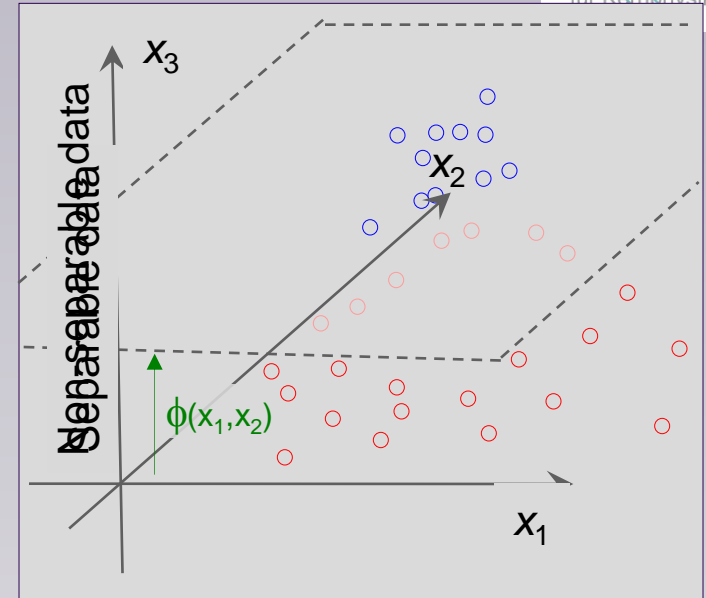
Support Vector Machines

- hyperplane that separates **S** from **B**
 - Linear decision boundary
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - If data non-separable add *misclassification cost* parameter $C \cdot \sum \xi_j$ to minimisation function
 - **largest margin - inner product of support vectors (distances) → quadratic minimisation problem**
- Non-linear cases:
 - Transform variables into higher dimensional feature space where again a linear boundary (hyperplane) can separate the data



Support Vector Machines

- Find hyperplane that best separates signal from background
 - Linear decision boundary
 - Best separation: maximum distance (margin) between closest events (*support*) to hyperplane
 - If data non-separable add *misclassification cost* parameter $C \cdot \sum \xi_j$ to minimisation function
 - **largest margin - inner product of support vectors (distances) → quadratic minimisation problem**
- Non-linear cases:
 - non linear variable transformation → linear separation in transformed feature space
 - no explicit transformation specified → Only its “scalar product” $x \cdot x \rightarrow \Phi(x) \cdot \Phi(x)$ needed.
 - certain *Kernel Functions* can be interpreted as scalar products between transformed vectors in the higher dimensional feature space. e.g.: **Gaussian, Polynomial, Sigmoid**
 - Choose Kernel and fit the hyperplane using the linear techniques developed above
 - Kernel size paramter typically needs careful tuning! (Overtraining!)

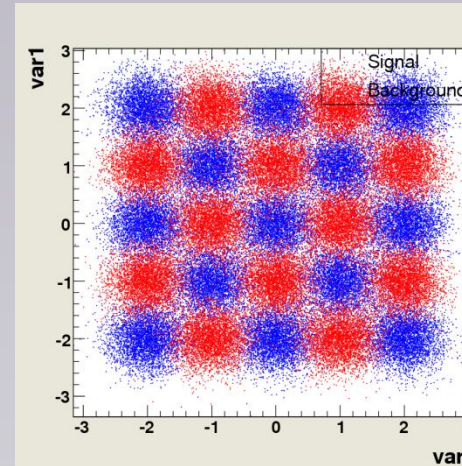


Support Vector Machines

- How does this “Kernel” business work?
- Kernel function == scalar product in “some transformed” variable space
- standard: $\vec{x} \cdot \vec{y} = \sum x_i y_i = |\vec{x}| |\vec{y}| * \cos(\theta)$
 - large if : $\vec{x} \cdot \vec{y}$ are in the same “direction”
 - zero if : $\vec{x} \cdot \vec{y}$ are orthogonal (i.e. point along different axes / dimension)
- e.g. Gauss kernel: $\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x}, \vec{y}) = \exp\left(-\frac{(\vec{x}-\vec{y})^2}{2\sigma^2}\right)$
 - zero if points: \vec{x} and \vec{y} “far apart” in original data space
 - large only in “vicinity” of each other
 - $\sigma <$ distance between training data points:
 - each data point is “lifted” into its “own” dimension
 - full separation of “any” event configuration with decision boundary along coordinate axis
 - well, that would of course be: overtraining

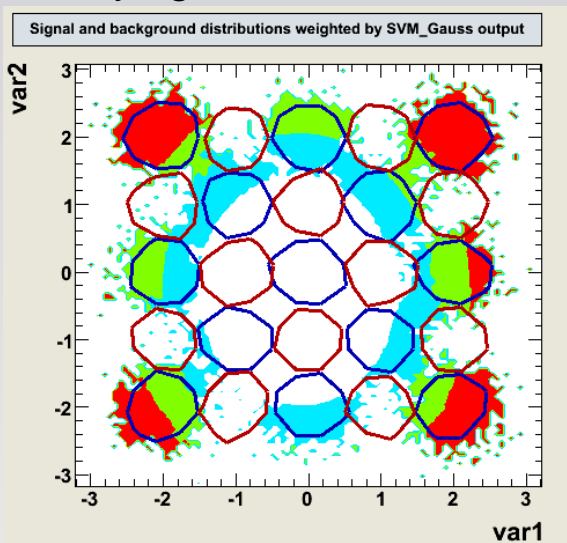
Support Vector Machines

SVM: the Kernel size parameter:
example: Gaussian Kernels

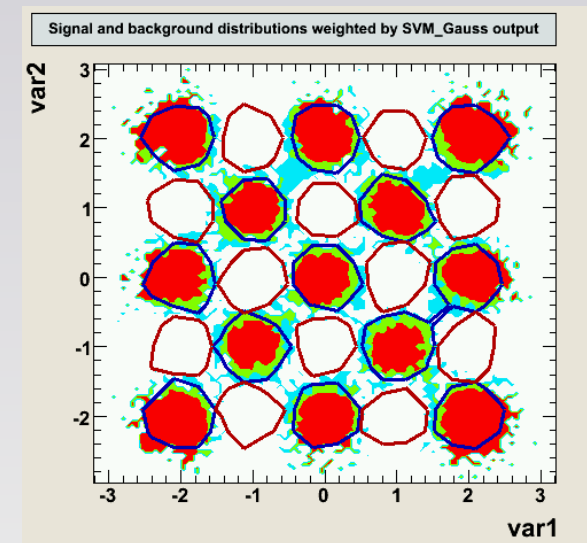


■ Kernel size (σ of the Gaussian) chosen too large: \rightarrow not enough “flexibility” in the underlying transformation

■ Kernel size (σ of the Gaussian) chosen properly for the given problem



colour code:
Red \rightarrow large signal
probability:



Overview

- **Multivariate classification/regression algorithms (MVA)**
 - what they are
 - how they work
- **Overview over some classifiers**
 - **Multidimensional Likelihood (kNN : k-Nearest Neighbour)**
 - **Projective Likelihood (naïve Bayes)**
 - **Linear Classifier**
 - **Non linear Classifiers**
 - Neural Networks
 - Boosted Decision Trees
 - Support Vector Machines
- **General comments about:**
 - **Overtraining**
 - **Systematic errors**

See some Classifiers at Work

- “generative models”:

- kNN/Likelihood \rightarrow PDF estimators $\rightarrow y(x) = \frac{PDF(x|Signal)}{PDF(x|Bkg)}$

- “discriminative models”

- (non-)linear model $y(x) = \sum w_i h(x)$: $h(x)$ – basis function of model

- Fisher Discriminant, Logistic Regression, Neural Networks

- Boosted Decision Trees: belong here: do you see what $h(x)$ is in that case?

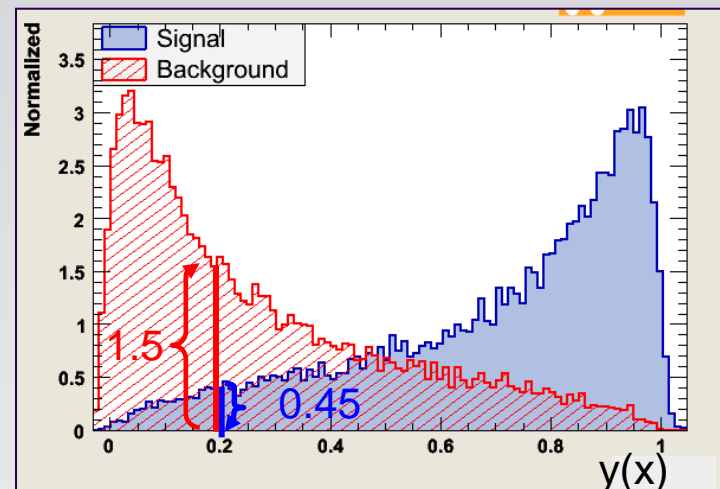
- So do SVM (linear model in transformed variable space \rightarrow it’s just expressed not directly as “sum over basis functions)

$\rightarrow y(x)$ for each event:

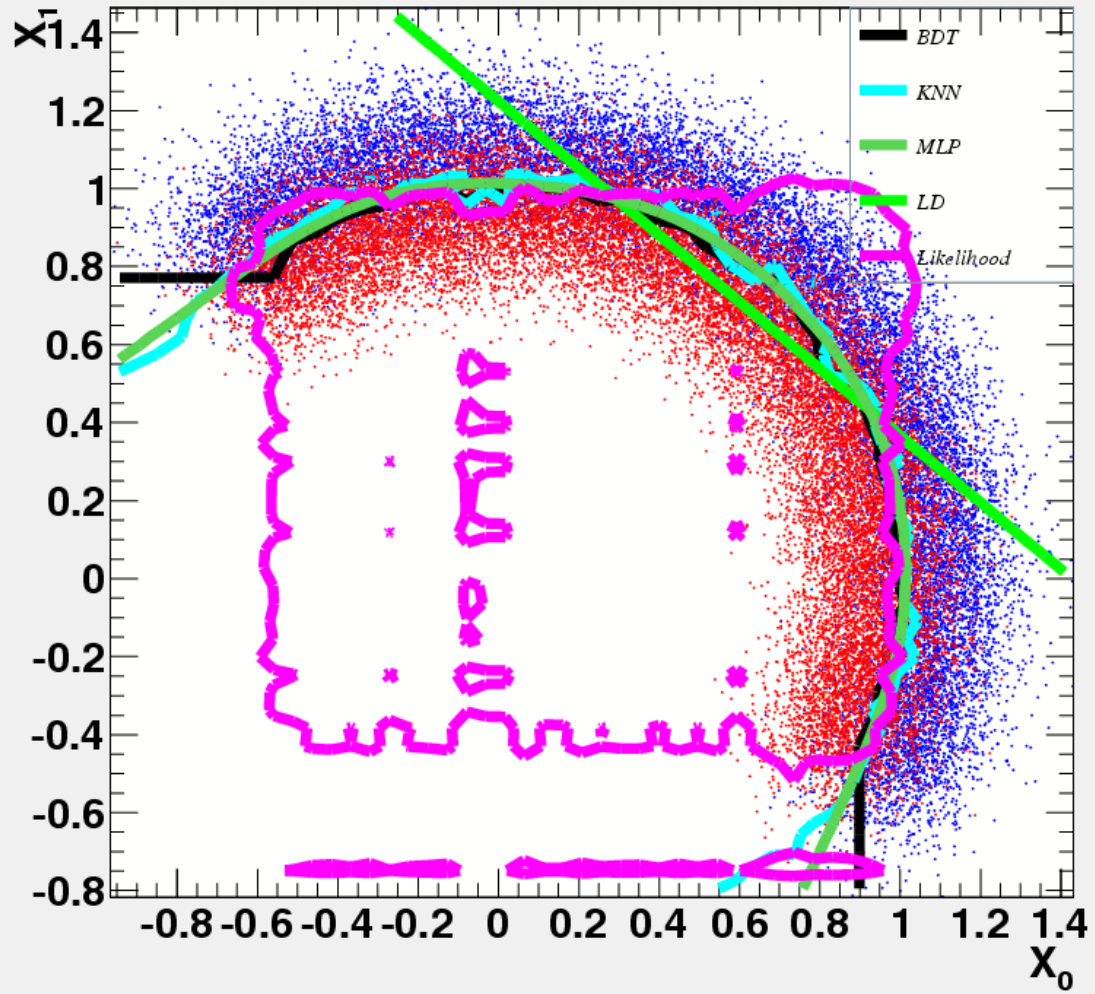
$\rightarrow P(S|x)$

\rightarrow pose cut on $P(S|x)$

\rightarrow weigh event with $P(S|x)$



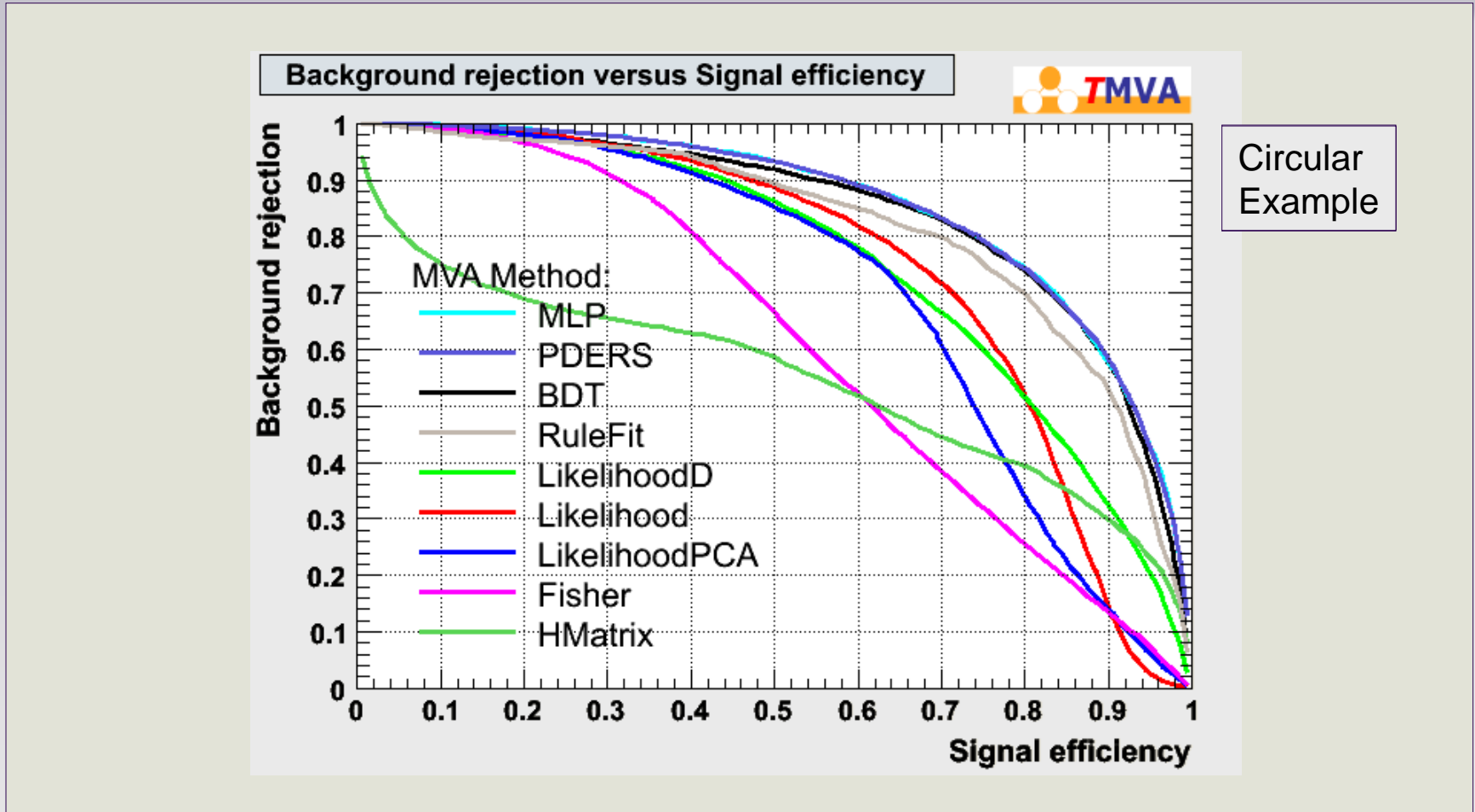
Decision Boundaries



- BDT**
- kNN**
- MLP**
- LD**
- Likelihood**

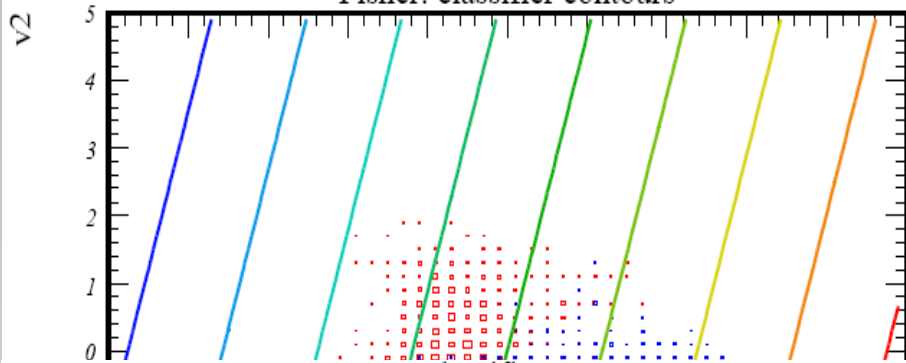
Final Classifier Performance

- Background rejection versus signal efficiency curve:

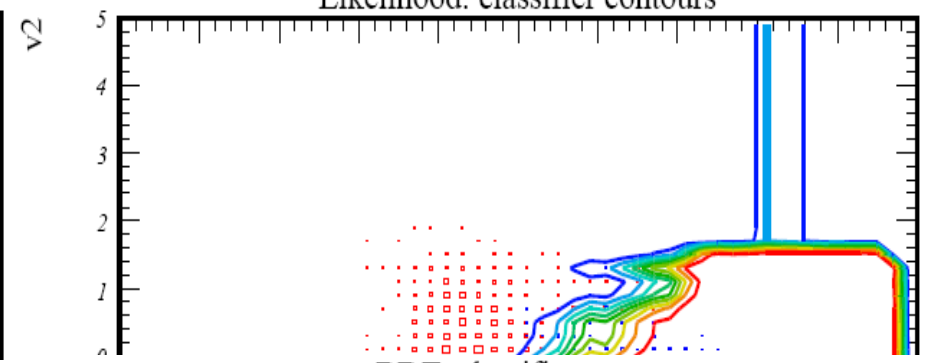


Visualisation of Decision Boundary

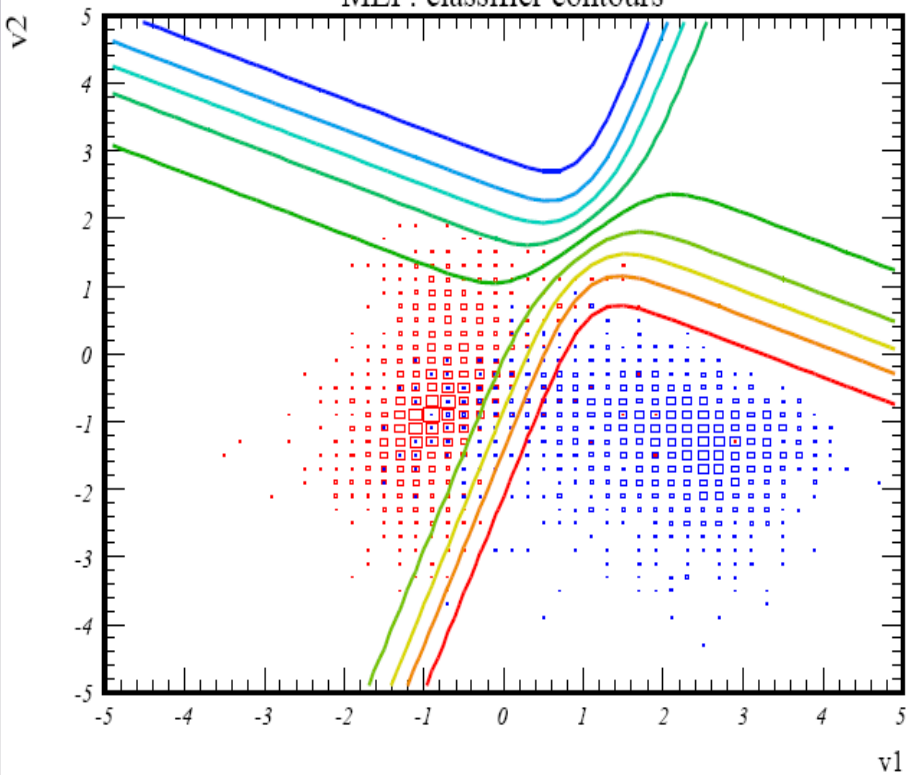
Fisher: classifier contours



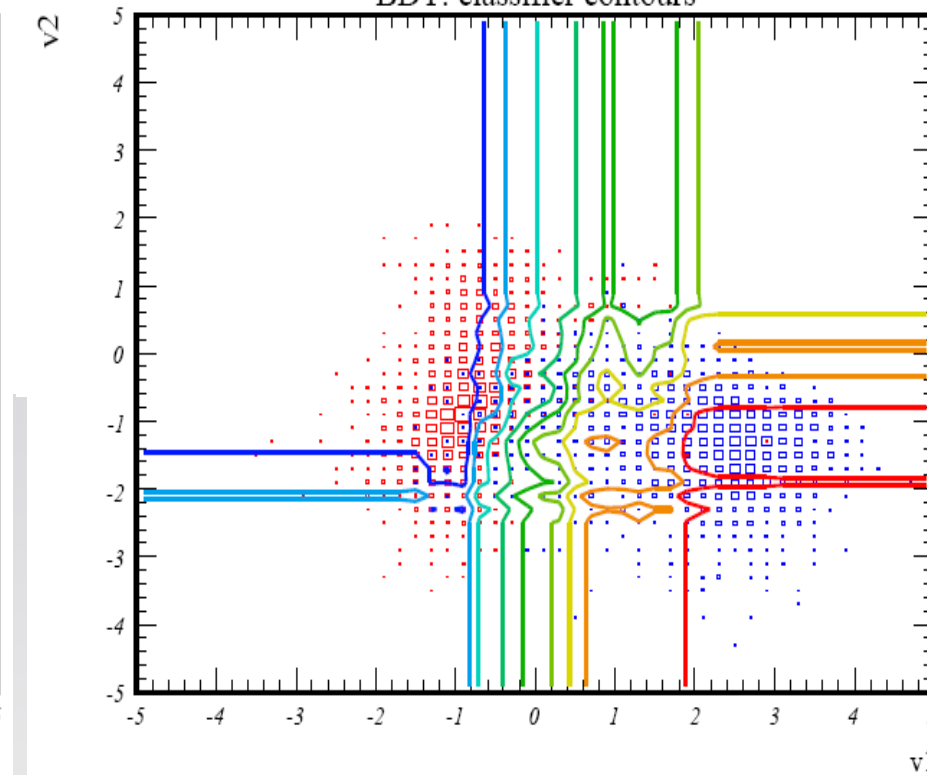
Likelihood: classifier contours



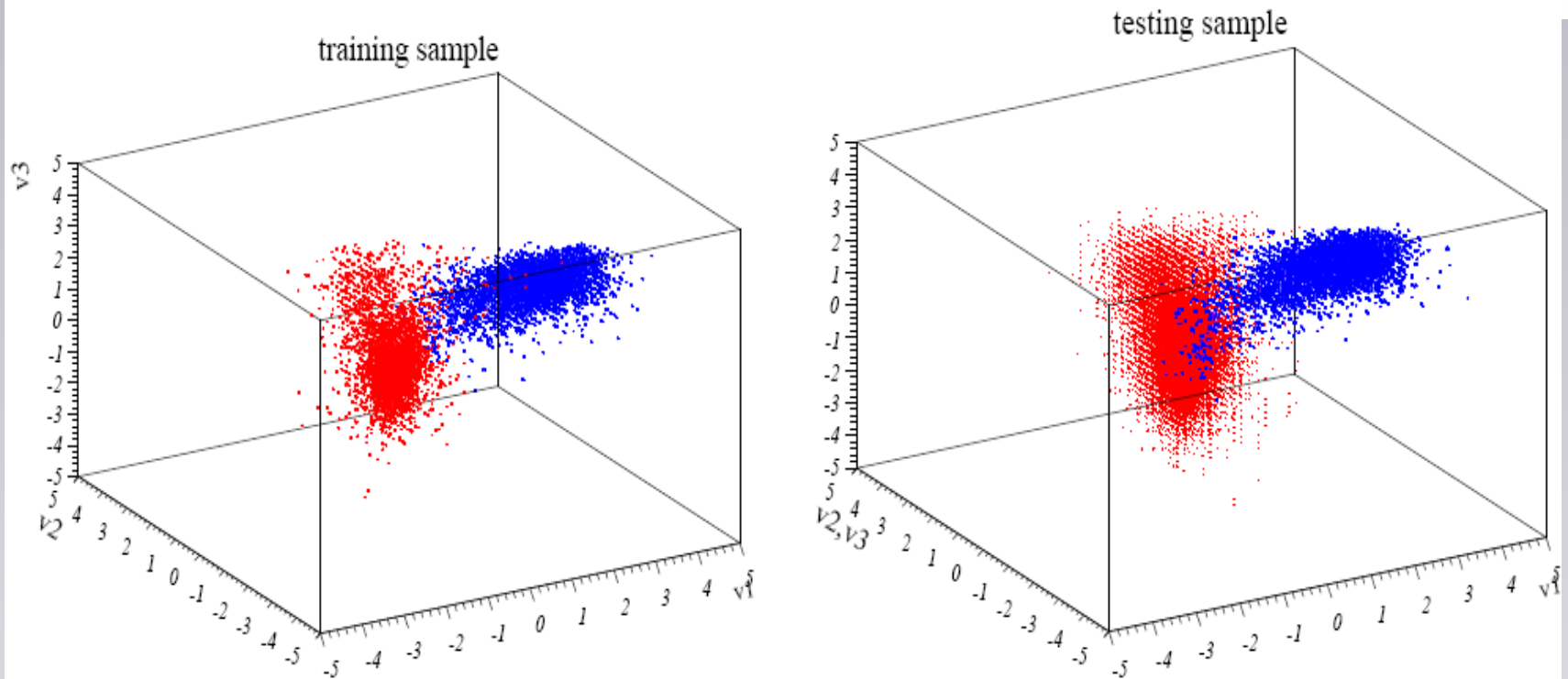
MLP: classifier contours



BDT: classifier contours



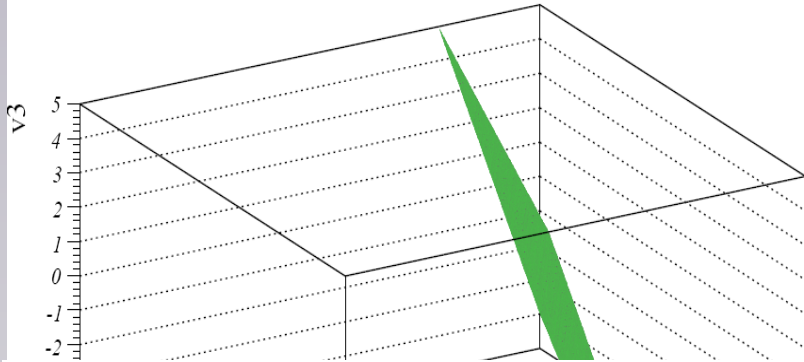
Visualisation in 3 Variables



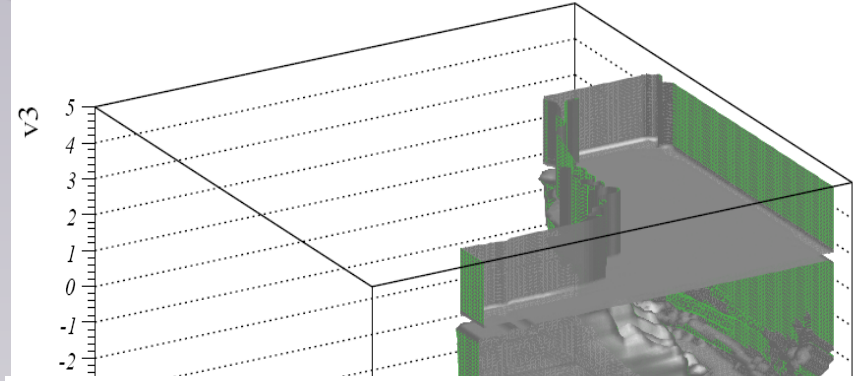
- some number of signal and background candidates in training
 - same statistical power to estimate PDFs
- imbalanced samples in testing
 - gauge performance under realistic conditions

Visualisation of Decision Boundary

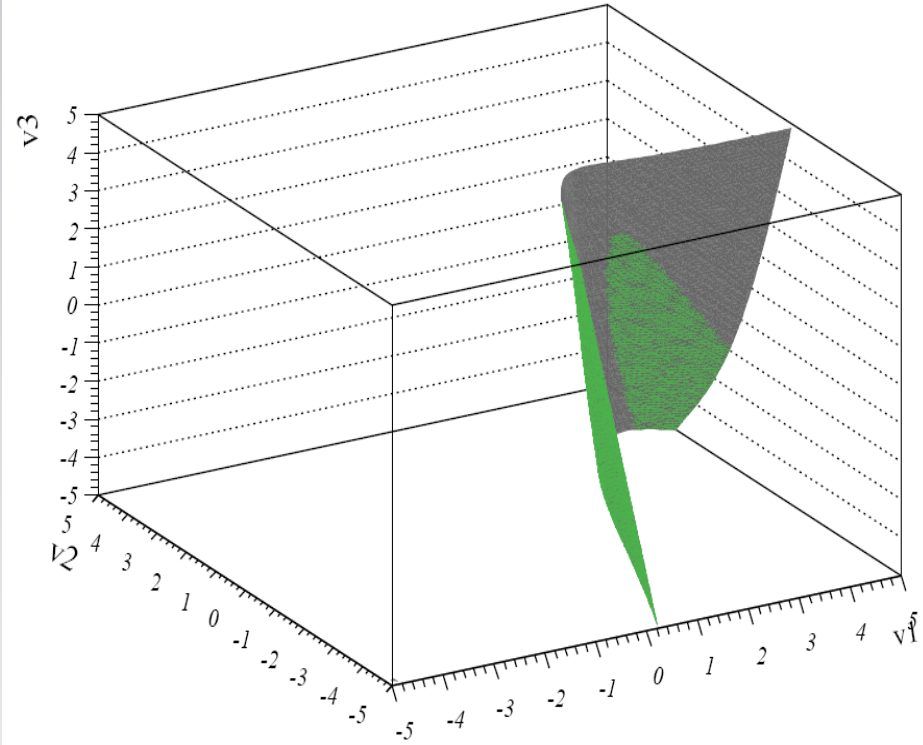
Fisher-3D: optimal cut contour



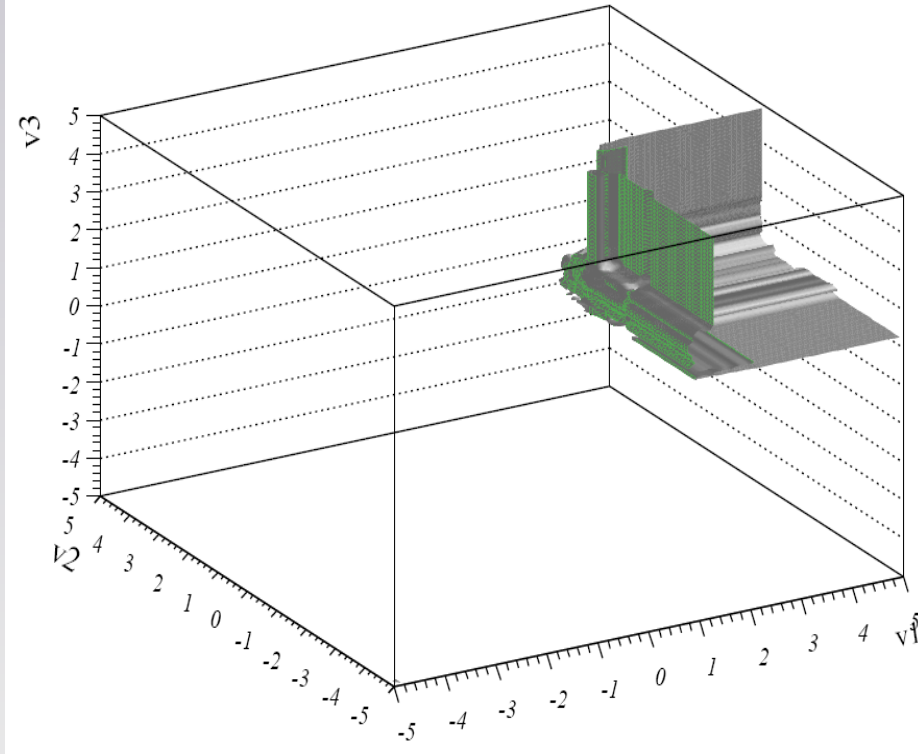
Likelihood-3D: optimal cut contour



MLP-3D: optimal cut contour



BDT-3D: optimal cut contour



General Advice for (MVA) Analyses

- There is no magic in MVA-Methods:
 - no need to be too afraid of “black boxes” → they are not sooo hard to understand
 - you typically still need to make careful tuning and do some “hard work”
 - no “artificial intelligence” ... just “fitting decision boundaries” in a given model
- The most important thing at the start is finding good observables
 - good separation power between S and B
 - little correlations amongst each other
 - no correlation with the parameters you try to measure in your signal sample!
- Think also about possible combination of variables
 - this may allow you to eliminate correlations
 - rem.: you are MUCH more intelligent than what the algorithm
- Apply pure preselection cuts and let the MVA only do the difficult part.
- “Sharp features should be avoided” → numerical problems, loss of information when binning is applied
 - simple variable transformations (i.e. $\log(\text{variable})$) can often smooth out these areas and allow signal and background differences to appear in a clearer way
- Treat regions in the detector that have different features “independent”
 - can introduce correlations where otherwise the variables would be uncorrelated!

“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 (→ **buhuhu... fails if $y(x)$ is NOT $P(S|x)$!!**)

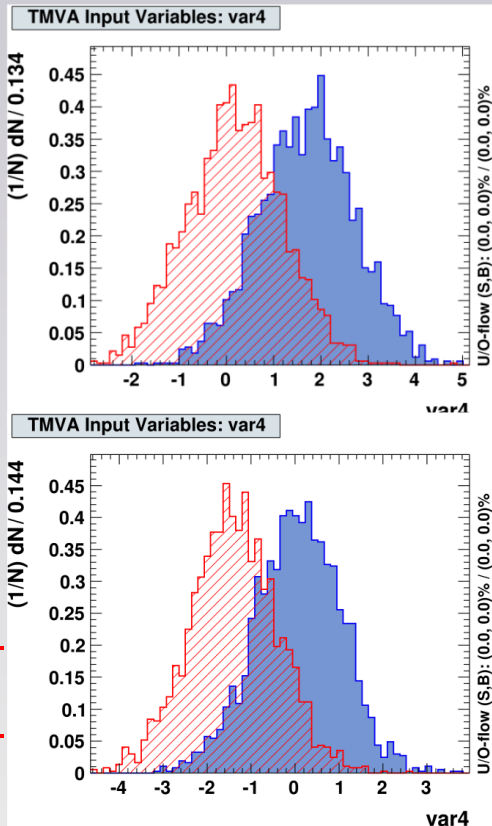
TMVA Categories

- TMVA Categories: one classifier per ‘region’
- ‘regions’ in the detector (data) with different features treated independent
 - improves performance
 - avoids additional correlations where otherwise the variables would be uncorrelated!

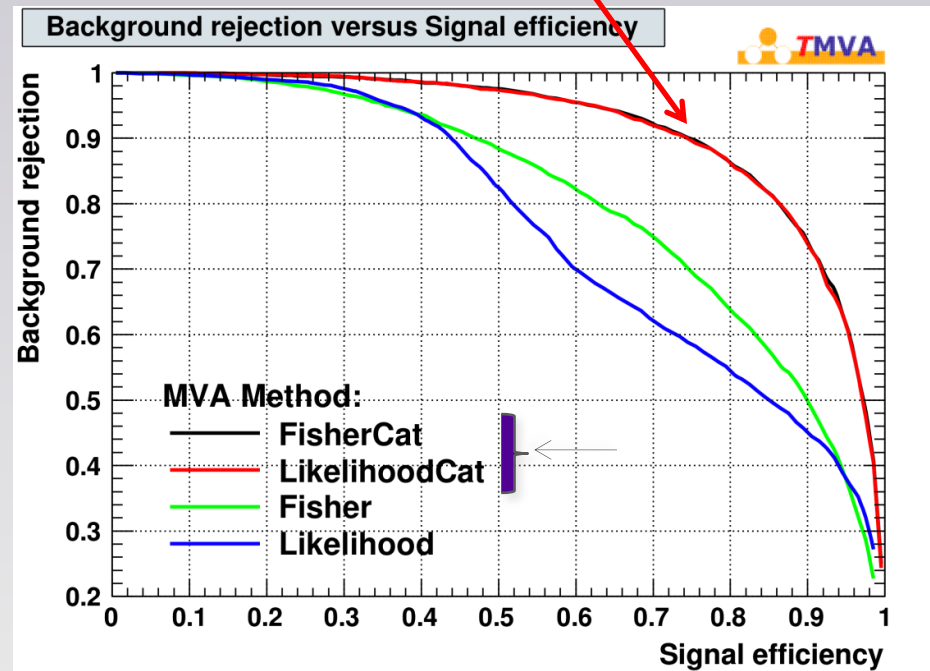
Example: var4 depends on some variable “eta”

$|\eta| > 1.3$

$|\eta| < 1.3$



Recover optimal performance after splitting into categories



Some Words about Systematic Errors

■ Typical worries are:

- What happens if the estimated “Probability Density” is wrong ?
- Can the Classifier, i.e. the discrimination function $y(x)$, introduce systematic uncertainties?
- What happens if the training data do not match “reality”

→ Any wrong PDF leads to imperfect discrimination function

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

→ Imperfect (calling it “wrong” isn’t “right”) $y(x)$ → loss of discrimination power

that’s all!

→ classical cuts face exactly the same problem, **however:**

in addition to cutting on features that are not correct, now you can also “exploit” correlations that are in fact not correct

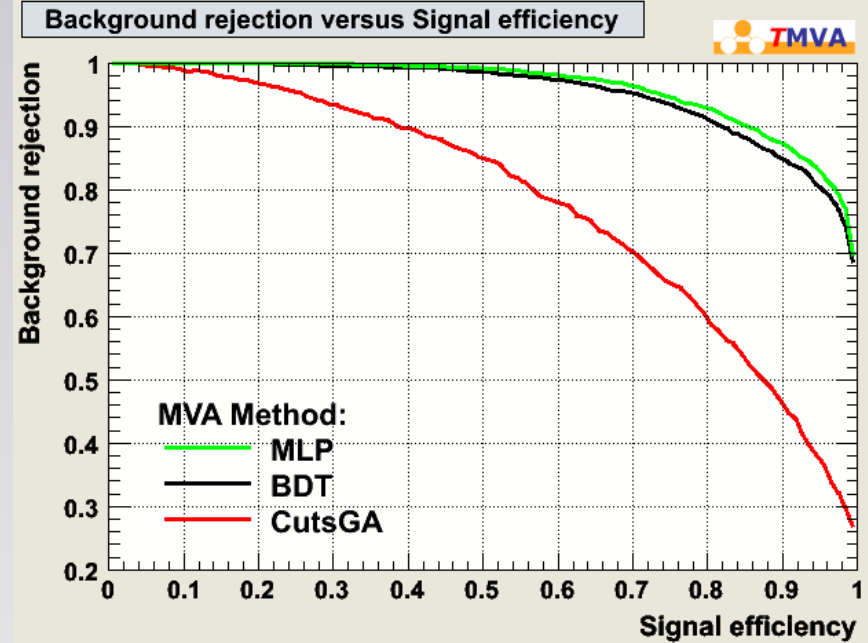
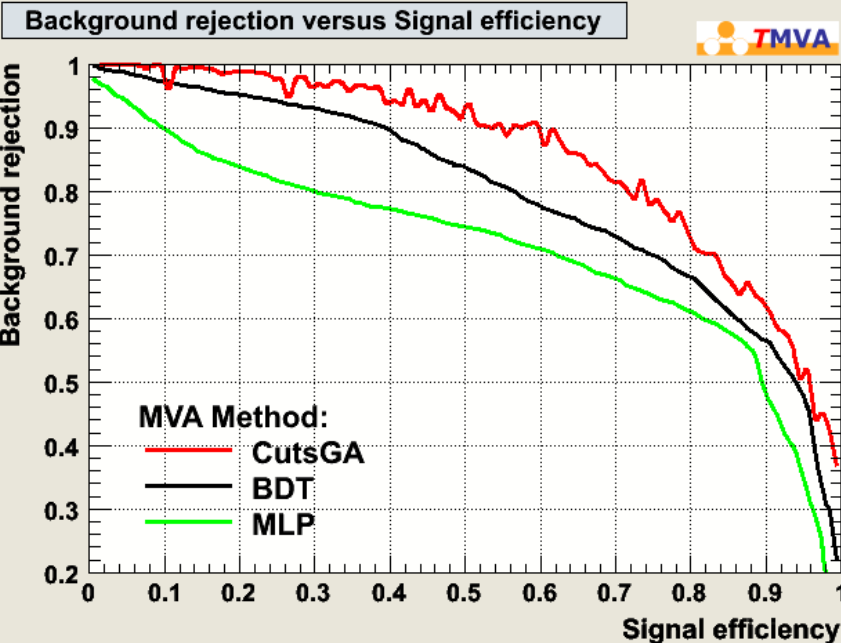
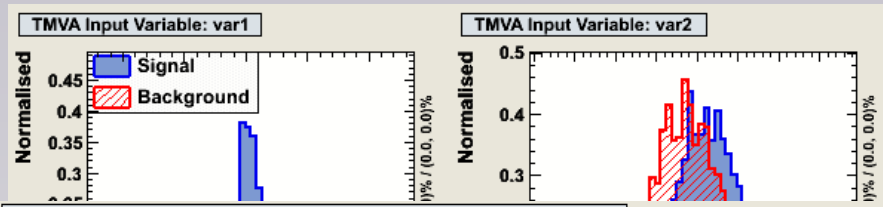
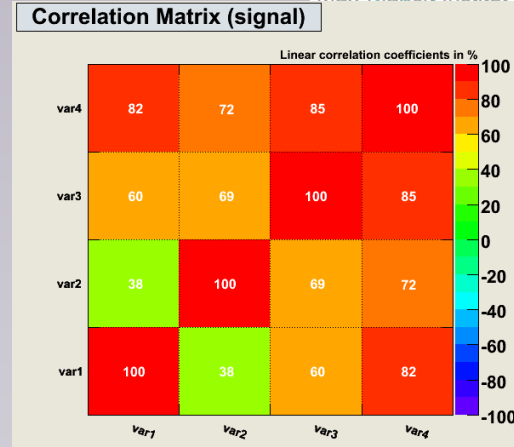
■ Systematic error are only introduced once “Monte Carlo events” with imperfect modeling are used for

- efficiency; purity
- #expected events
- same problem with classical “cut” analysis
- use control samples to test MVA-output distribution ($y(x)$)

■ Combined variable (MVA-output, $y(x)$) might “hide” problems in ONE individual variable more than if looked at alone → train classifier with few variables only and compare with data

Systematic “Error” in Correlations

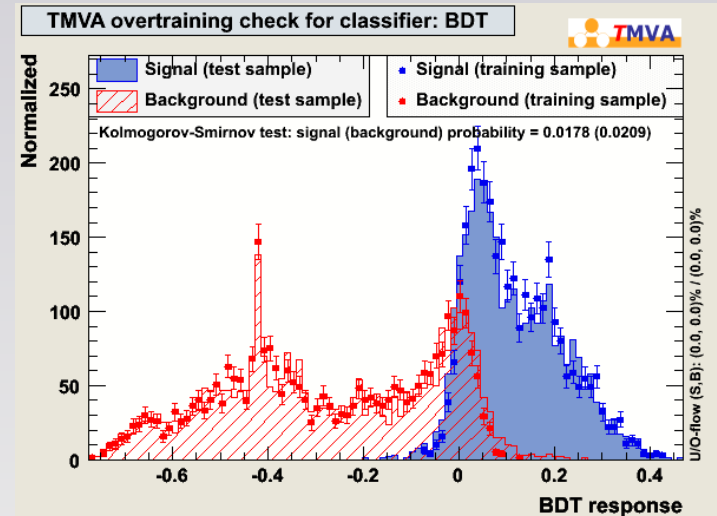
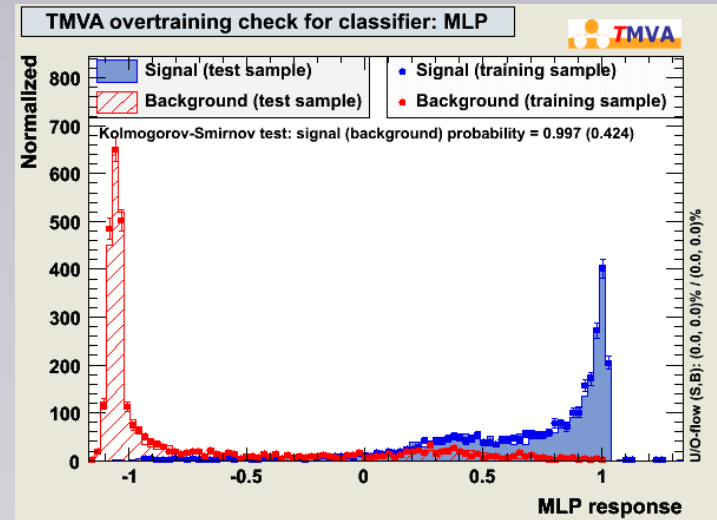
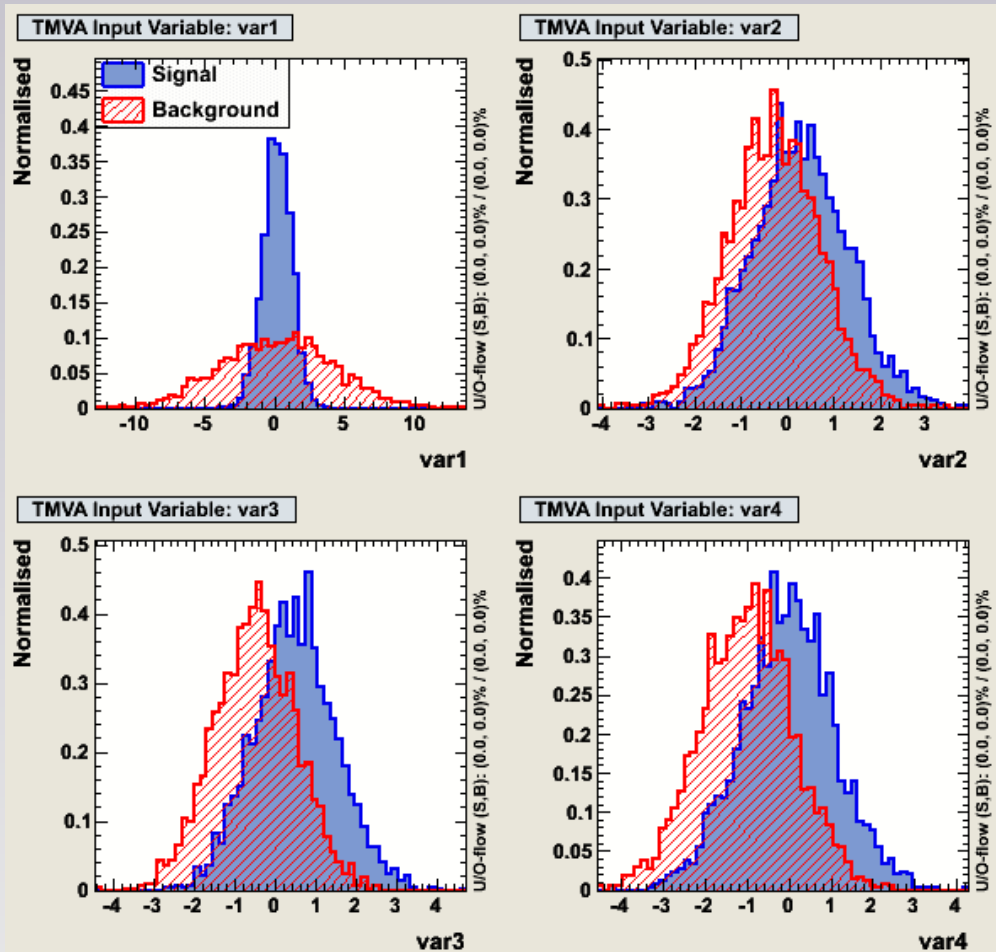
- Use as training sample events that have correlations
 - optimize CUTs
 - train an proper MVA (e.g. Likelihood, BDT)



- Assume in “real data” there are NO correlations → SEE what happens!!

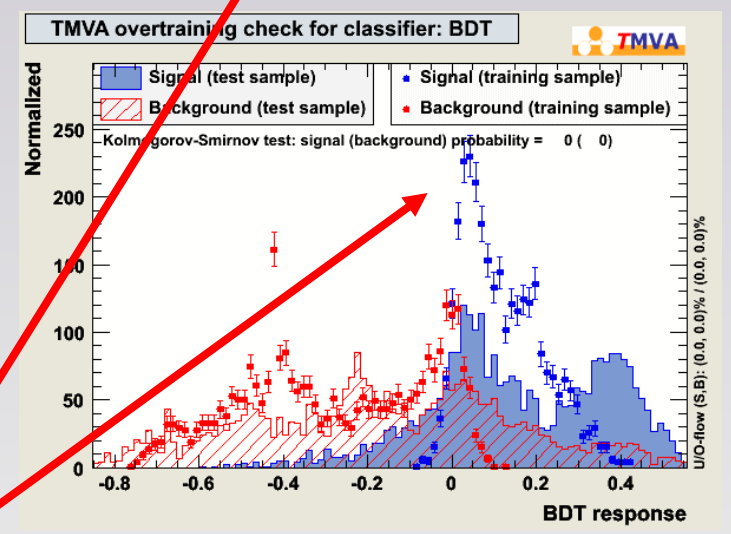
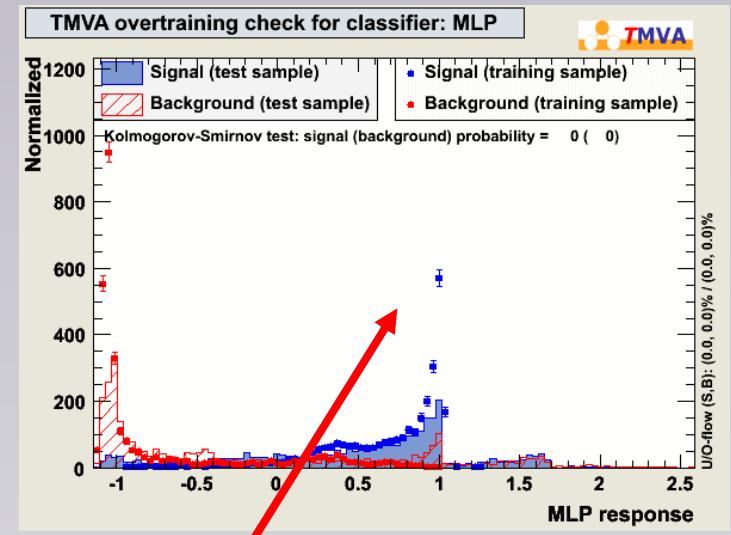
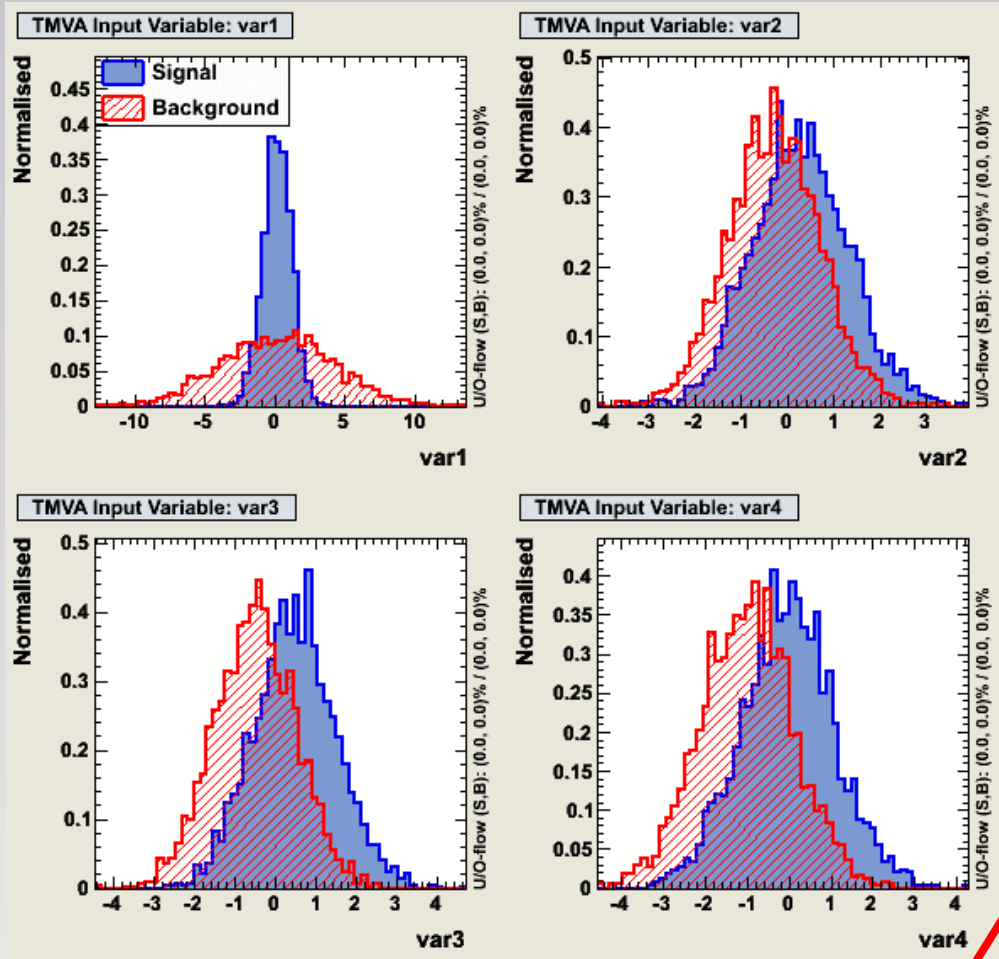
Systematic “Error” in Correlations

- Compare “Data” (TestSample) and Monte-Carlo (both taken from the same underlying distribution)



Systematic “Error” in Correlations

- Compare “Data” (TestSample) and Monte-Carlo (both taken from the same underlying distributions that **differ by the correlation!!!**)



Differences are ONLY visible in the MVA-output plots (and if you'd look at cut sequences....)

(*T*)MVA and Systematic Uncertainties

→ Don't be afraid of correlations!

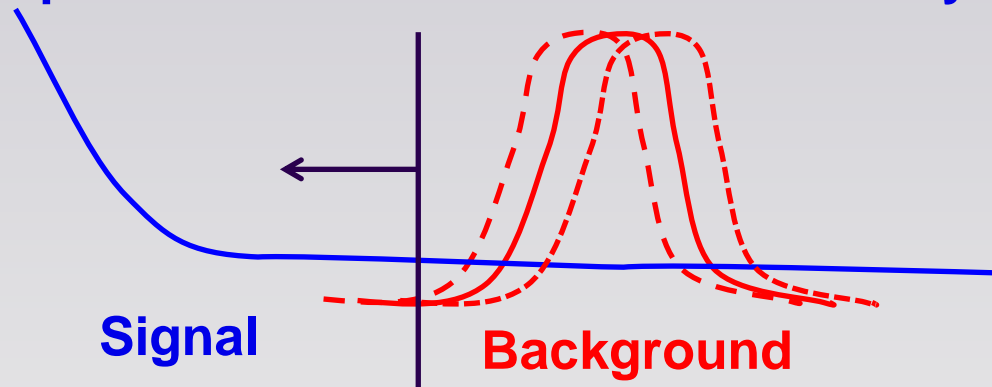
- typically “kinematically generated” → easily modeled correctly
- “classical cuts” are also affected by “wrongly modeled correlations”
- MVA method let's you spot mis-modeled correlations!
 - “projections” of input variables
 - + the combined MVA test statistic “ $y(x)$ ” !

MVA and Systematic Uncertainties

- Multivariate Classifiers THEMSELVES don't have systematic uncertainties
→ even if trained on a “phantasy Monte Carlo sample”
 - there are only “bad” and “good” performing classifiers !
 - **OVERTRAINING is NOT a systematic uncertainty !!**
 - **difference between two classifiers resulting from two different training runs DO NOT CAUSE SYSTEMATIC ERRORS**
 - same as with “well” and “badly” tuned classical cuts
 - MVA classifiers: → only select regions in observable space
- Efficiency estimate (Monte Carlo) → statistical/systematic uncertainty
 - involves “estimating” (uncertainties in) distribution of $PDF_{y_{S(B)}}$
 - statistical “fluctuations” → re-sampling (Bootstrap)
 - “smear/shift/change” input distributions and determine $PDF_{y_{S(B)}}$
 - estimate systematic error/uncertainty on efficiencies
- Only involves “test” sample..
 - systematic uncertainties have nothing to do with the training !!

(T)MVA and Systematic Uncertainties

- minimize “systematic” uncertainties (robustness)
- “classical cuts” : do not cut near steep edges, or in regions of large sys. uncertainty
- hard to translate to MVAs:
 - artificially degrade discriminative power (shifting/smearing) of systematically “uncertain” observables IN THE TRAINING
 - remove/smooth the ‘edges’ → MVA does not try to exploit them
 - **First attempts to automatize this are on the way**



- **Note: if I KNEW about the error, I'd correct for it. I'm talking about 'unknown' systematics**

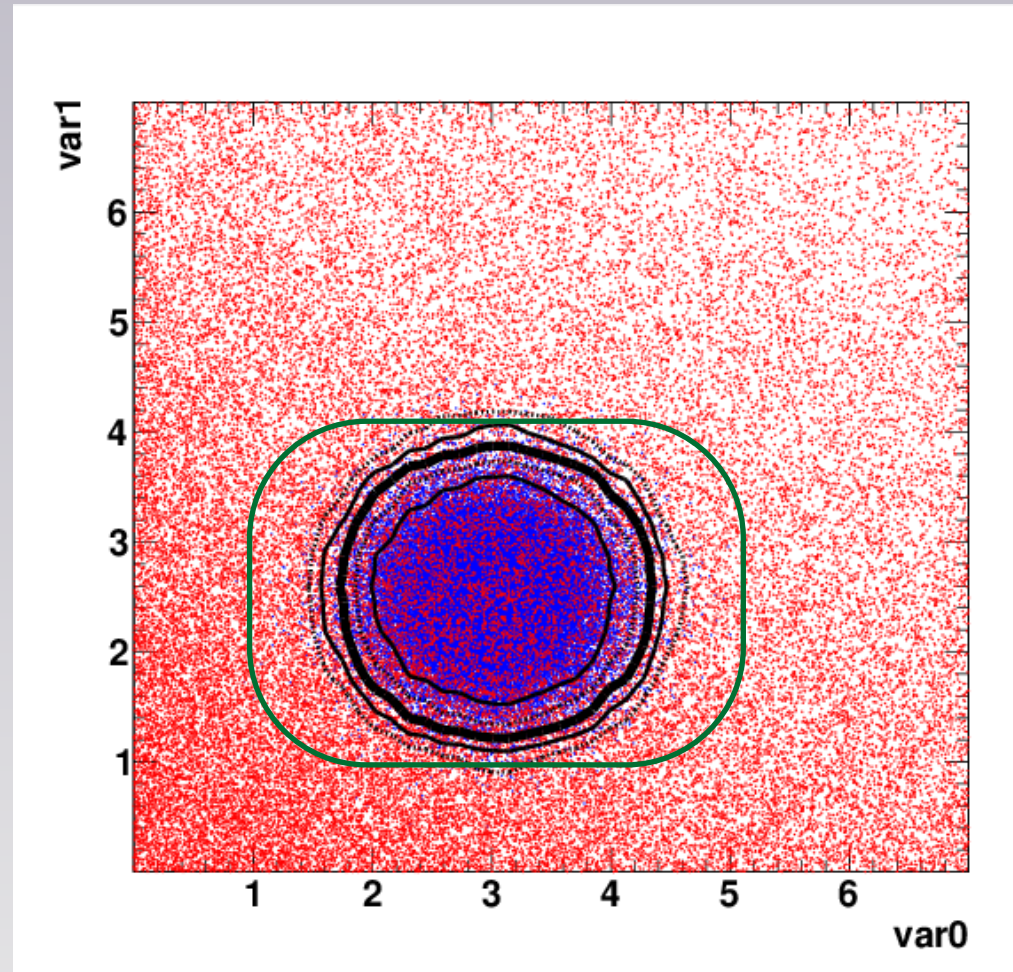
How does this look in 2D?

MVA-decision boundaries

- Looser MVA-cut \rightarrow wider boundaries in BOTH variables

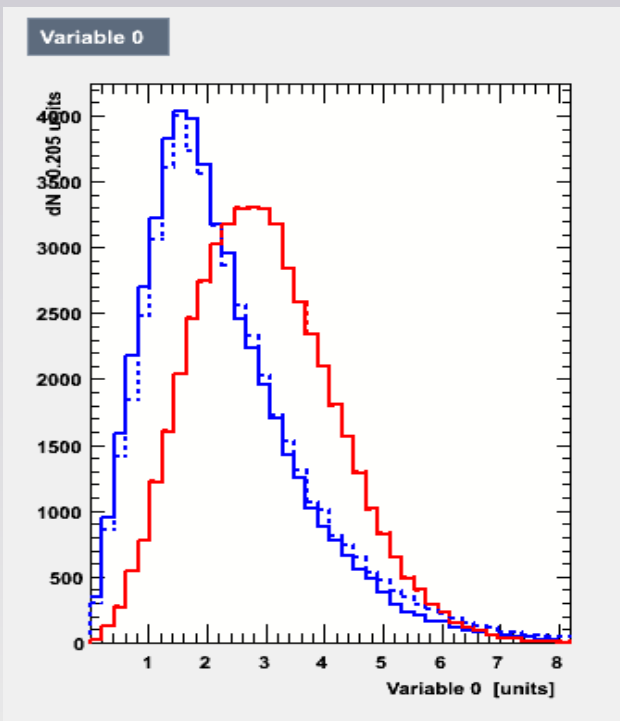
What if you are sure about the peak's position in var1, but less sure about var0 ?

- You actually want a boundary like **THIS**
 - Tight boundaries in var1
 - Loose boundaries in var0



Reduce information content

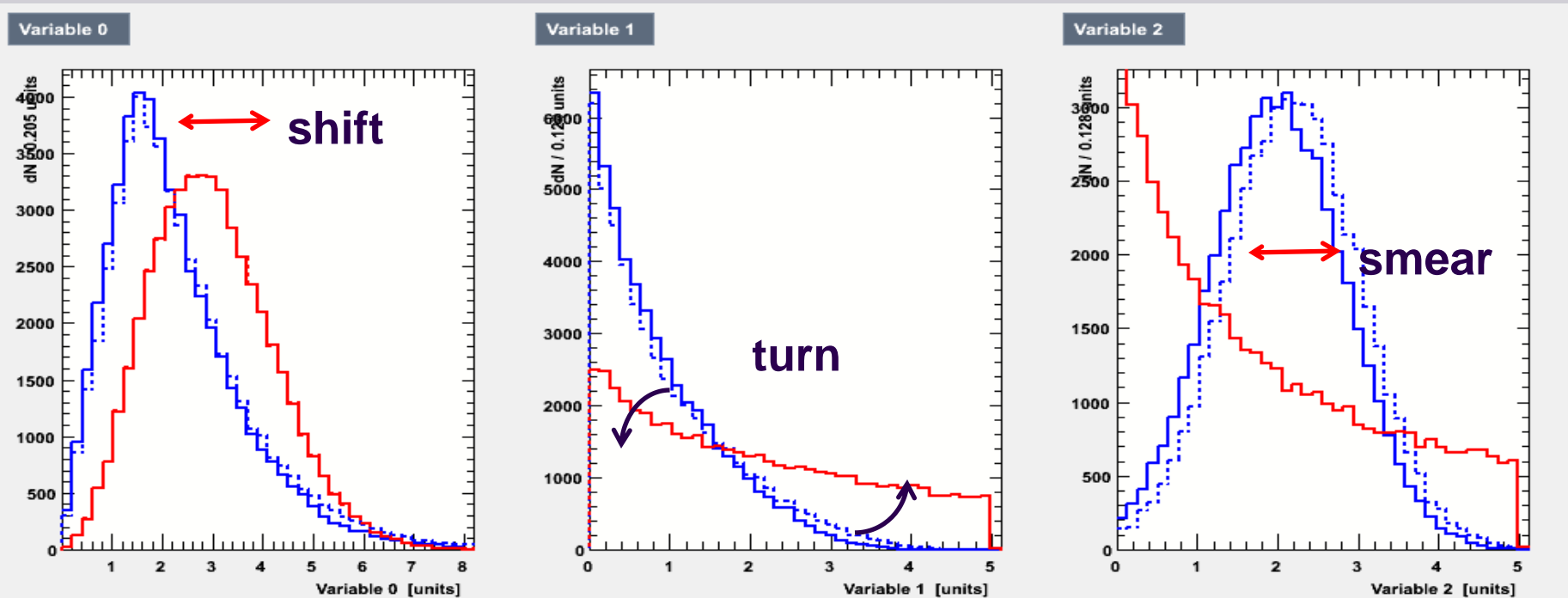
- Looking for a general tool to ‘force’ any MVA algorithm, not to rely too much on exact feature:
 - Similar: early stopping techniques in Neural networks to avoid overtraining
- reduce difference between “signal” and “background”
- or reduce information content in each, “signal” and “background”



- Here: one would for example “shift” such that signal and backgr. are less separated
- However, that’s not “universal”

Reduce information content

- Looking for a general tool to ‘force’ any MVA algorithm, not to rely too much on exact feature:
 - Similar: early stopping techniques in Neural networks to avoid overtraining
- reduce difference between “signal” and “background”
- or reduce information content in each, “signal” and “background”

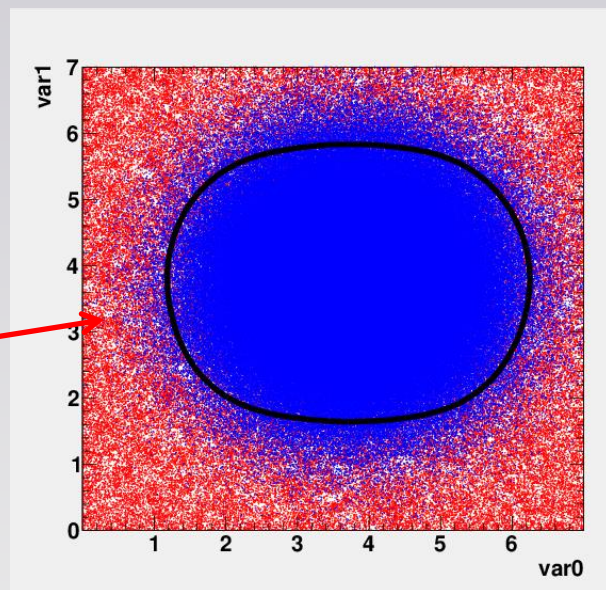
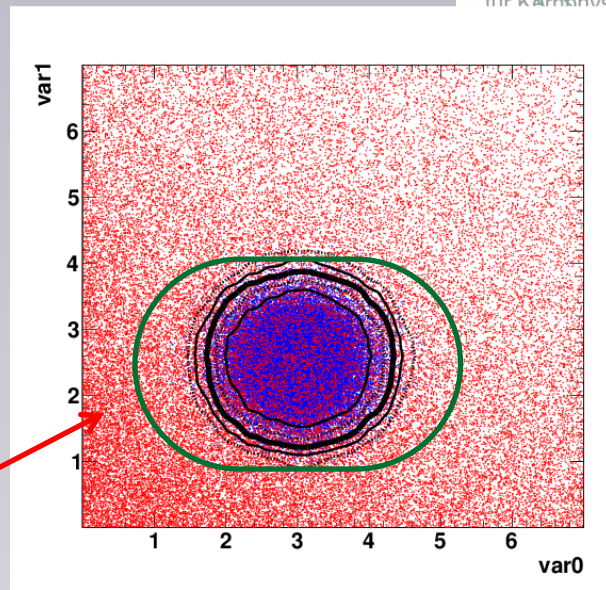


Reminder:

MVA-decision boundaries

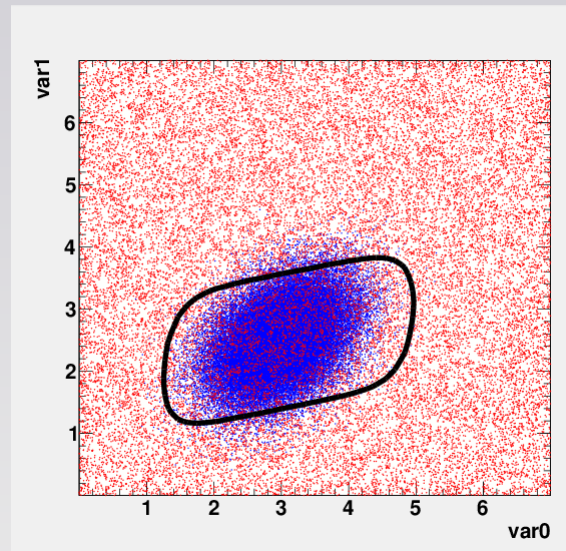
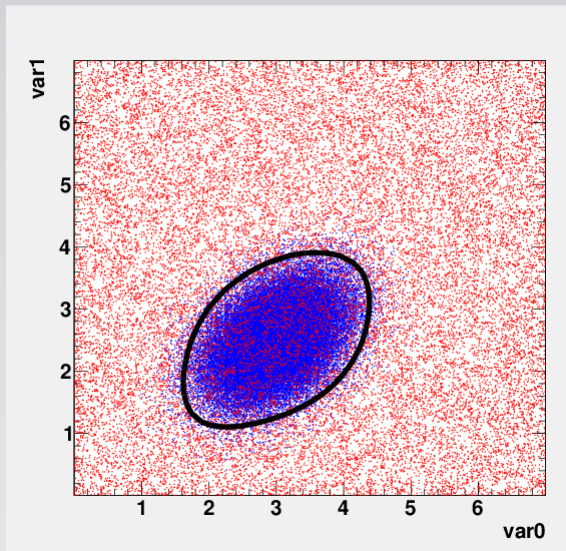
- Looser MVA-cut \rightarrow wider boundaries in BOTH variables
- You actually want a boundary like **THIS**
 - Tight boundaries in var1
 - Loose boundaries in var0

■ **YES it works !**



Another example..

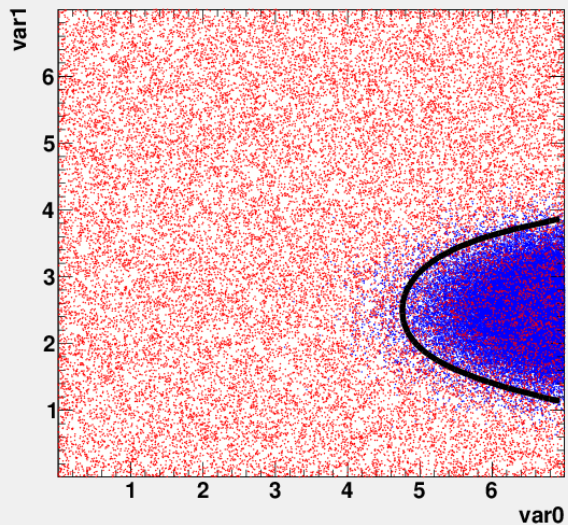
- Hmm... also here, I'd still say it does exactly what I want it to do
 - The difficulty is to 'evaluate' or 'estimate' the advantage (reduction in systematic \leftrightarrow loss in performance)



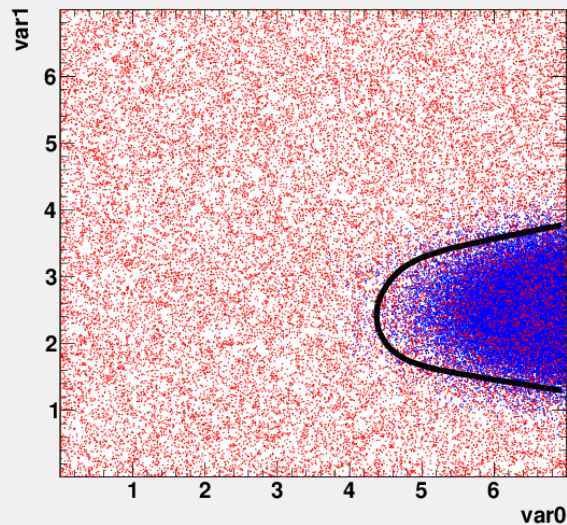
peak on the edge

AGAIN: assume uncertainty in position of peak in var0

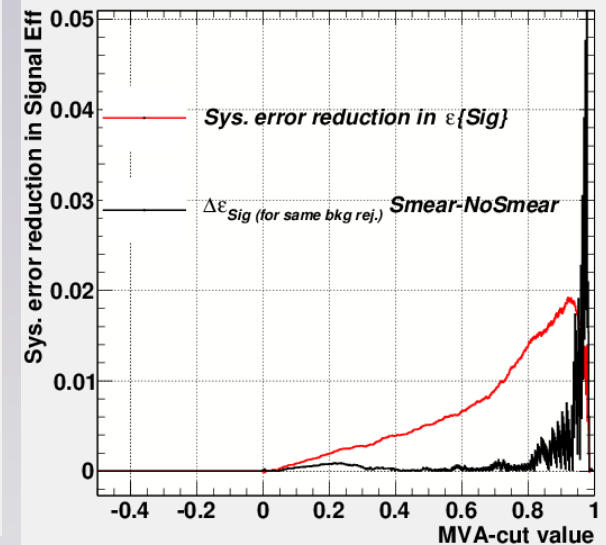
bad



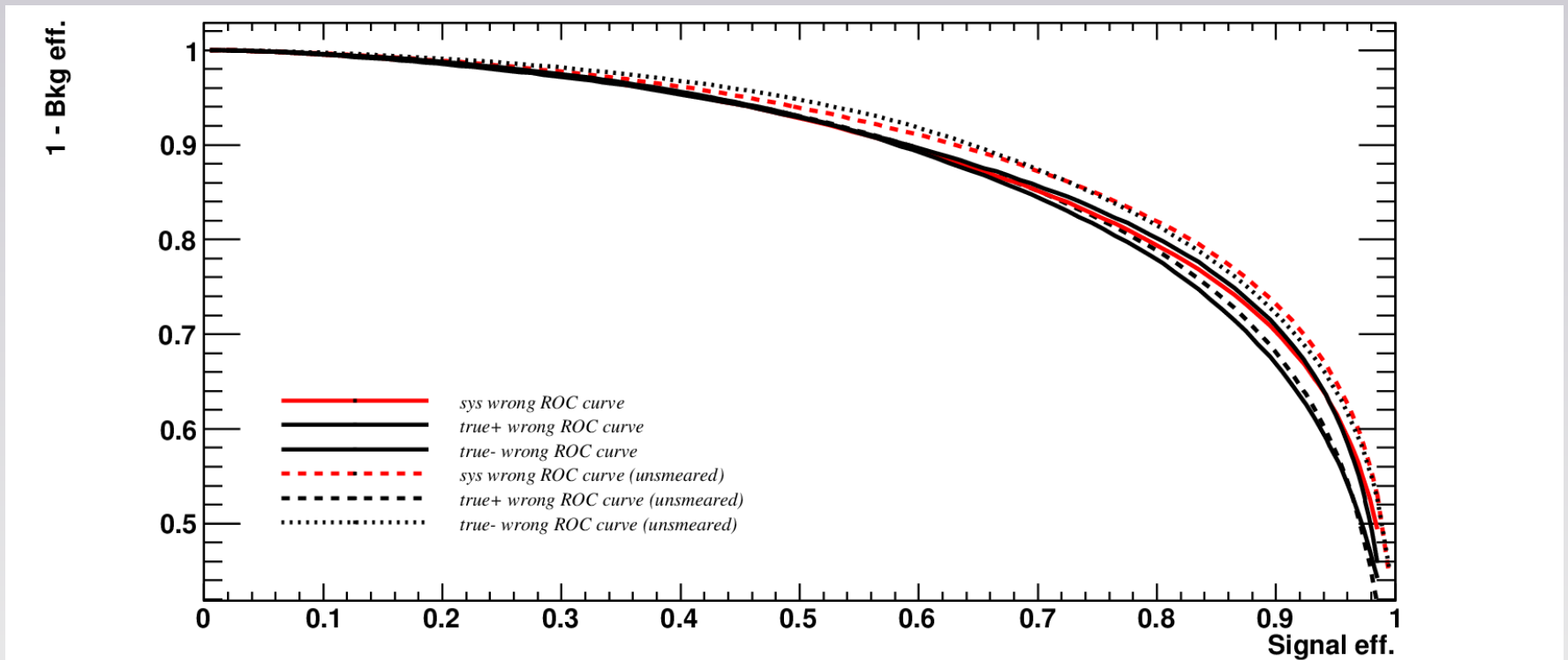
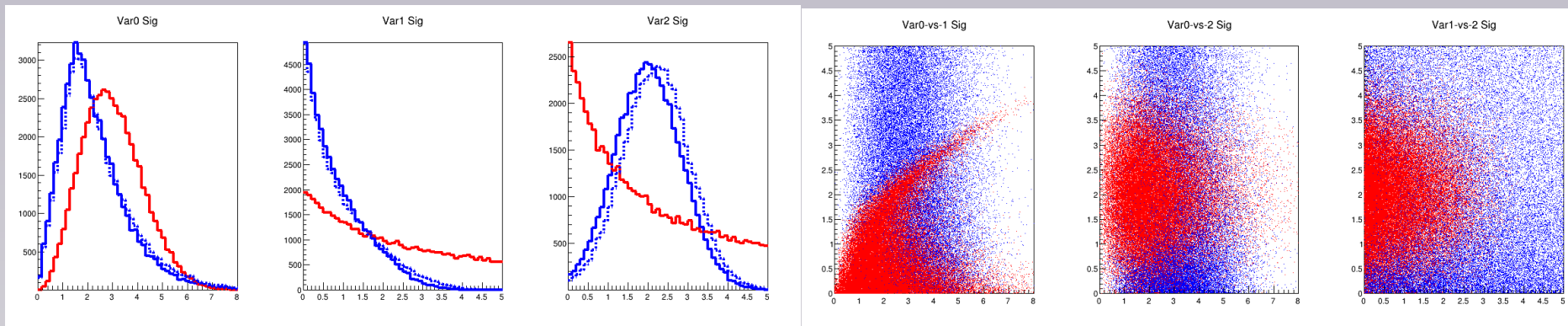
better



but by how much??

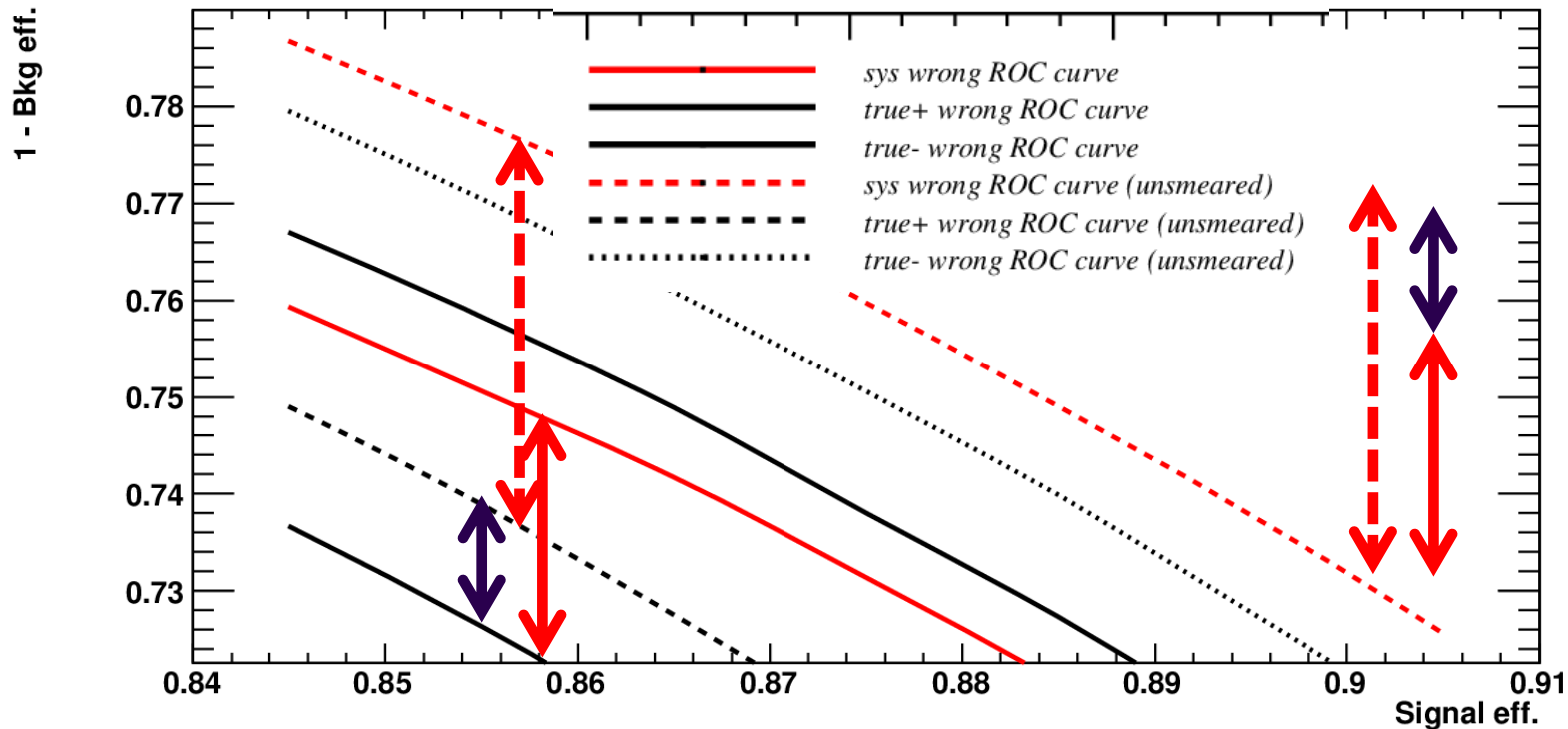


Back to my "complicated" 3D example



ROC Curve - Zoom

- compare: difference between **red (what you think you have)** and **black (what your algorithm applied to nature might actually provide)**
- do this for **solid (smeared)** and **dashed (unsmeared)** classifiers

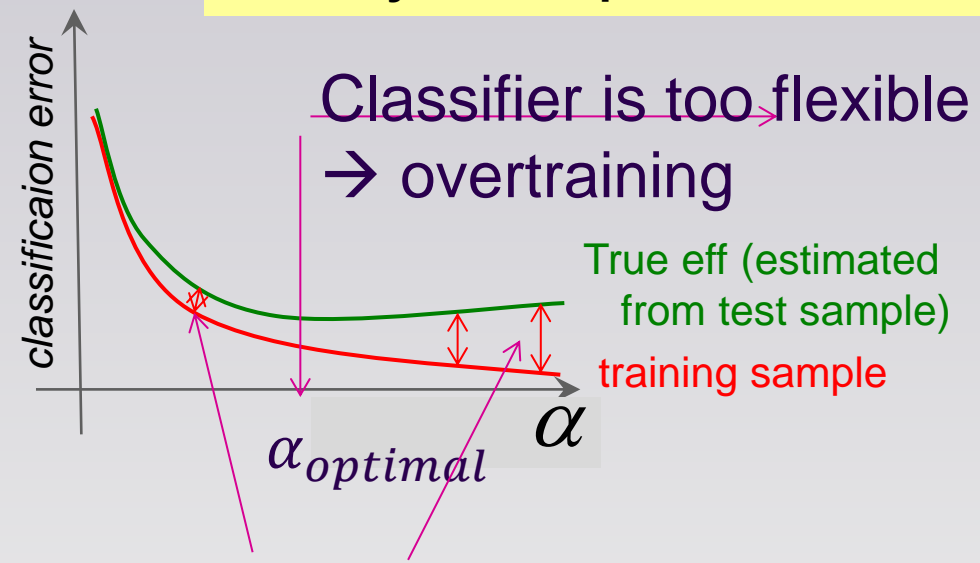
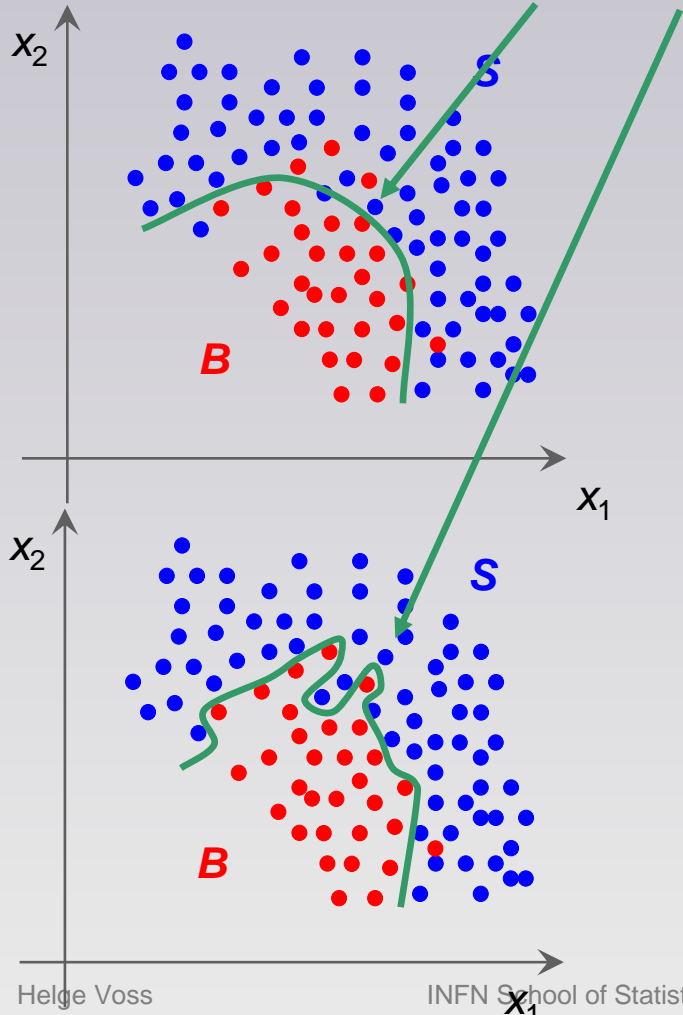


Overtraining

- how to choose 'meta' parameters " α " that determine the classifier 'flexibility' e.g. Number of Nodes in Neural network, number of training cycles, "size of neighbourhood (k)" in nearest Neighbour algorithms etc...
- it seems intuitive that this boundary will give better results in another statistically independent data set than that one

→ possible overtraining is concern for every "tunable parameter" α of classifiers: Smoothing parameter, n-nodes...

→ verify on independent "test" sample

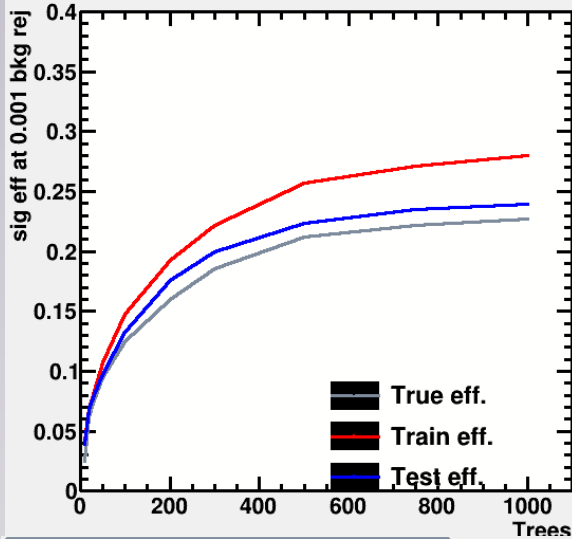


Bias if 'efficiency/performance' is estimated from the training sample

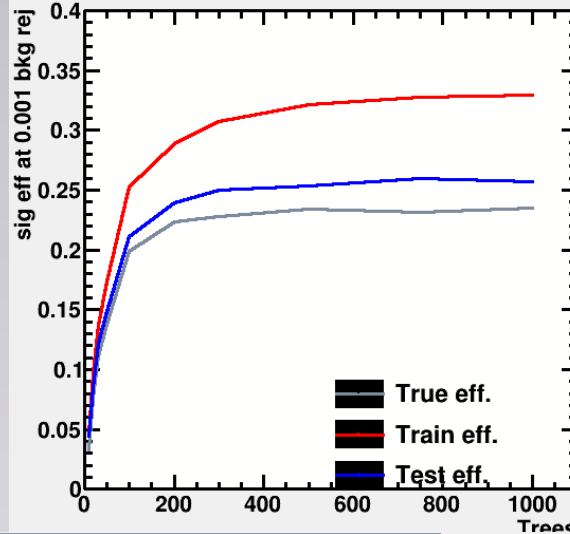
BDT performance vs #trees

For background rejection of 99.9% (bkg. Eff. 0.1%)

average True efficiency (maxDepth=1)



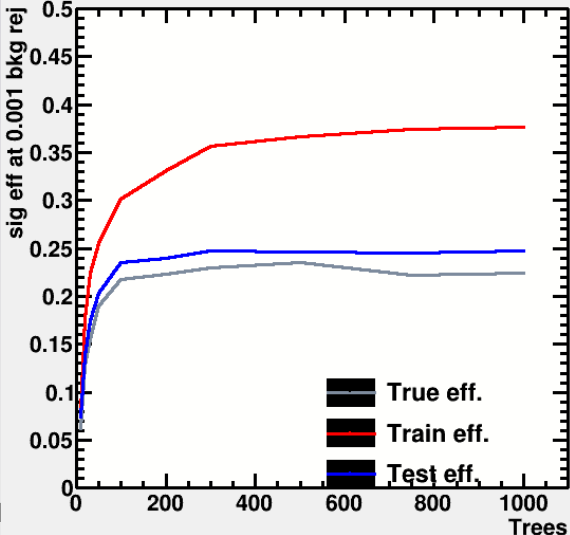
average True efficiency (maxDepth=2)



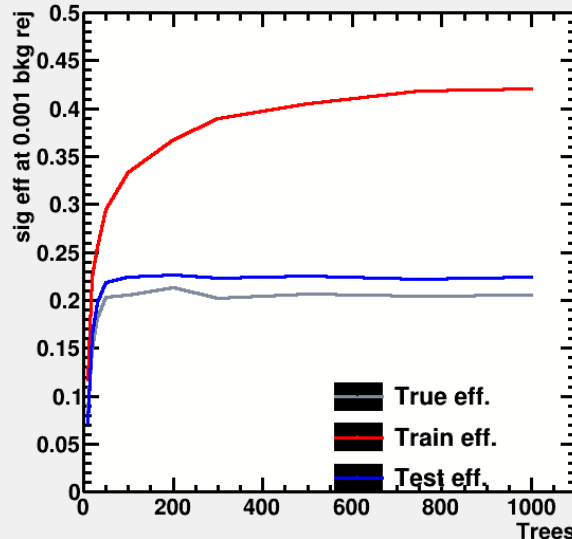
Notice: Best performance is at MaxDepth = 2 and Ntrees = 1000

→ That is NOT where the difference between true, test and training efficiency is minimal !!!

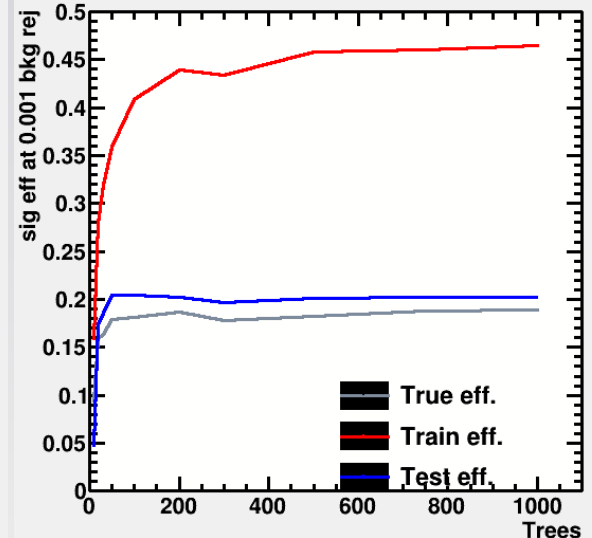
average True efficiency (maxDepth=3)



average True efficiency (maxDepth=4)



average True efficiency (maxDepth=6)



BDT:

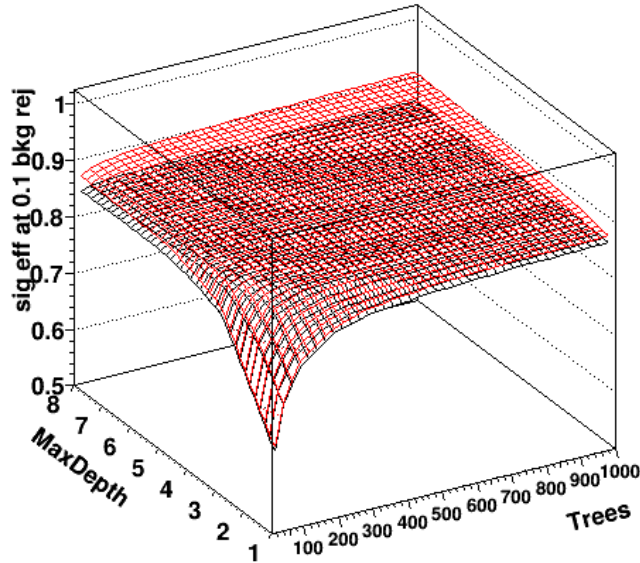
**Training
Efficiency**

**Testing
Efficiency**

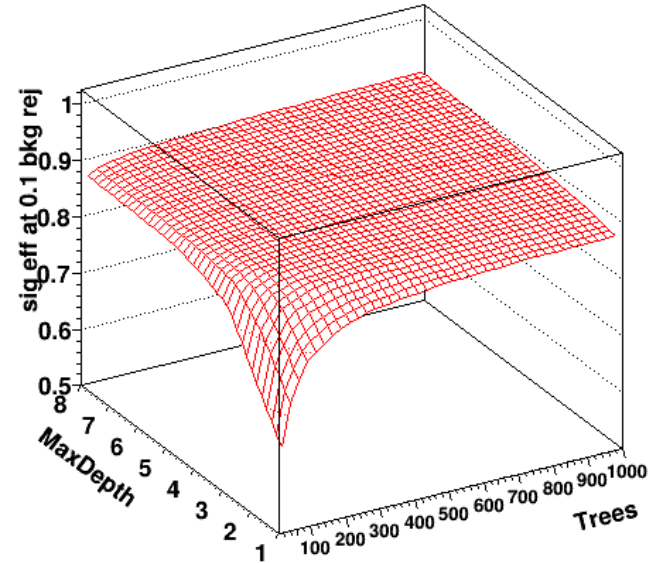
**True
Efficiency**

**for nominal
background
eff 10%**

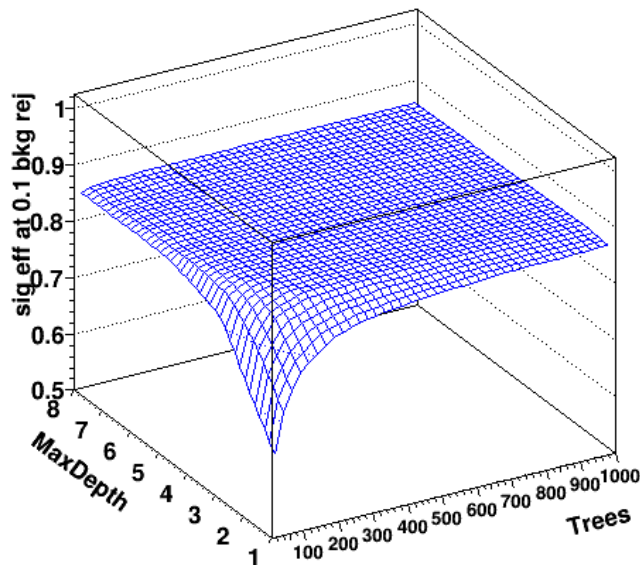
average True efficiency at 0.1 bkg



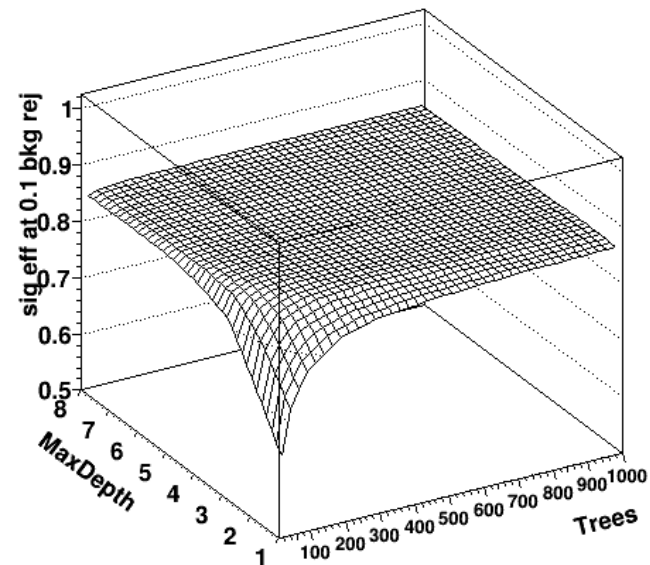
average Training efficiency at 0.1 bkg



average Testing efficiency at 0.1 bkg

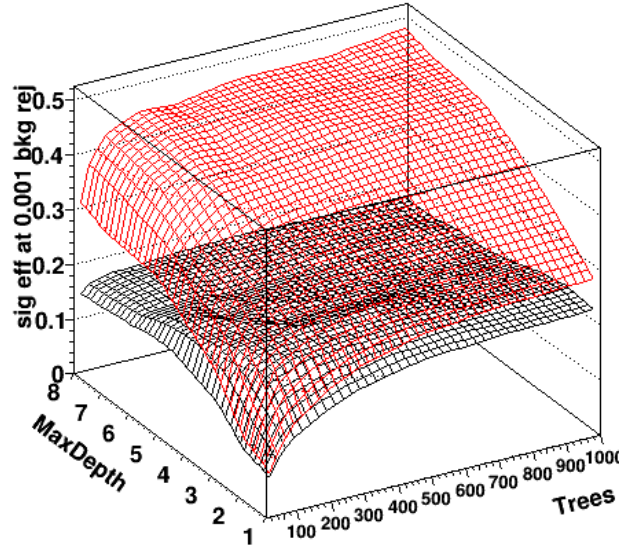


average True efficiency at 0.1 bkg

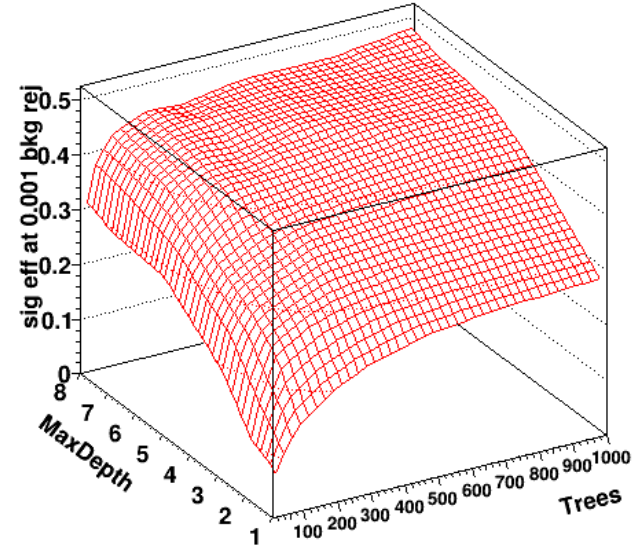


BDT:
Training Efficiency
Testing Efficiency
True Efficiency
for nominal background eff 0.1%

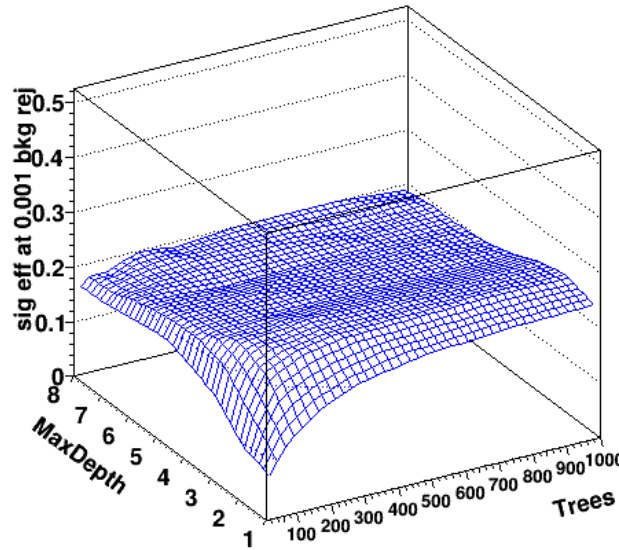
average True efficiency at 0.001 bkg



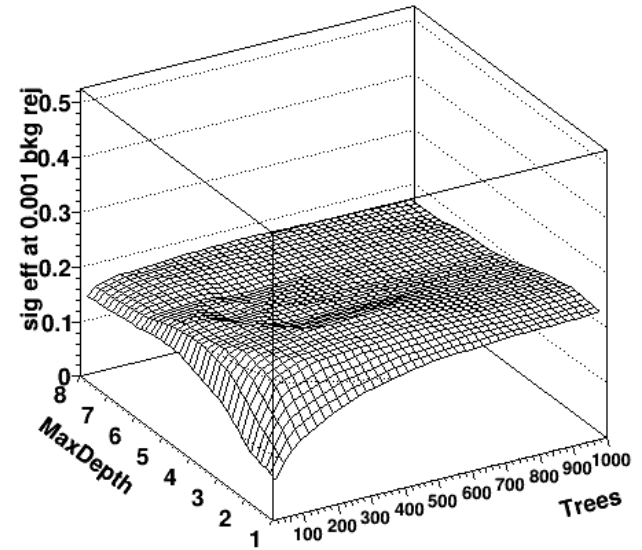
average Training efficiency at 0.001 bkg



average Testing efficiency at 0.001 bkg



average True efficiency at 0.001 bkg



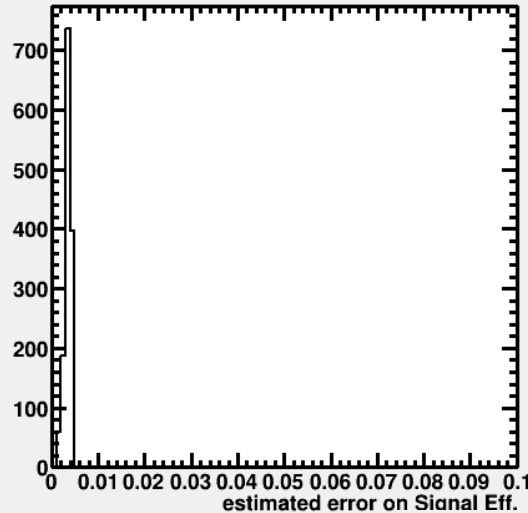
And for Cuts ?

Bias is fairly small, but for 99.9% bkg. Rejection still statistically significant

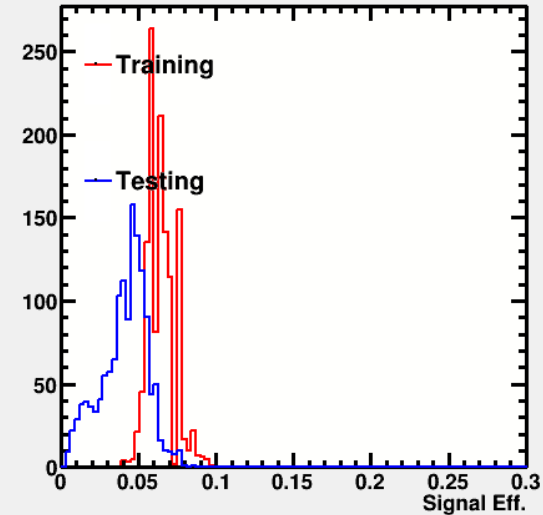
Training: effS = 6.5%
Testing : = 4.2%
True: = 4.1

Difference: 2%+- 0.06%

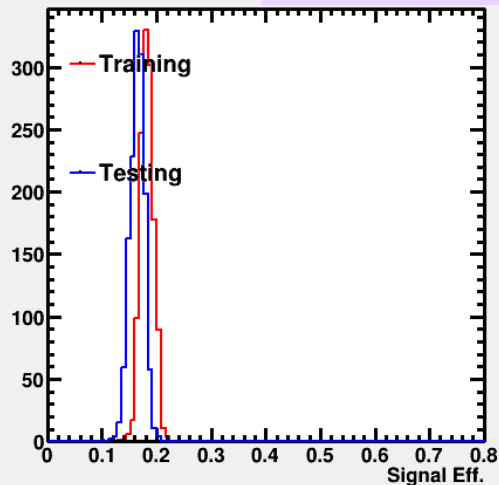
Testing Delta EffS (at 0.001 %bkg)



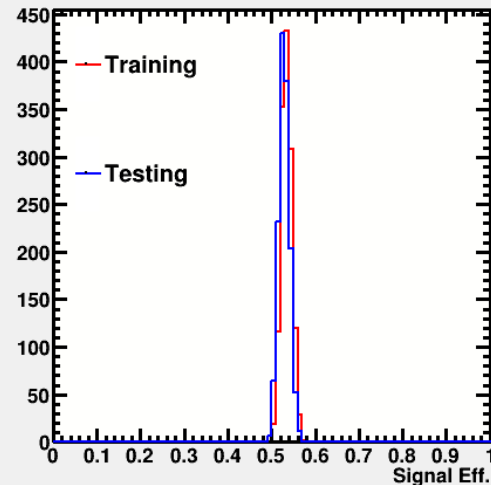
Training EffS (at 0.001 %bkg)



Training EffS (at 0.01 %bkg)



Training EffS (at 0.1 %bkg)



Summary

- **Multivariate Classifiers (Regressors)**
 - (fit) decision boundary (target function)
 - **Generative Classifiers:**
 - direct exploitation of Neyman-Pearson's Lemma: (best test statistic is the Likelihood ratio) → direct pdf estimate in
 - **Discriminative Classifiers:**
 - multi-dimensional (and projective) Likelihood
 - classifiers that “fit” a decision boundary for given “model”
 - Linear: Linear Classifier (e.g. Fisher Discriminant)
 - Non-Linear: ANN, BDT, SVM
- **TMVA lets ou train/test understand/judge/improve your classifier**
 - carefully study the control plots
 - compare different MVAs !
 - find working point on ROC curve
- **MVAs are not magic ... just fitting**
 - systematic uncertainties don't lie in the training !!
 - estimate them similar as you'd do in classical cuts

Backup and Left overs...

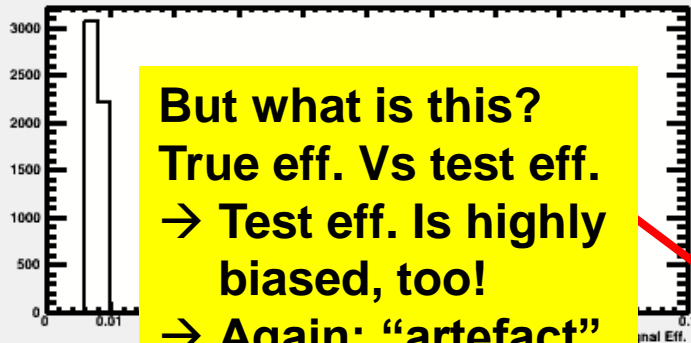
Example distributions

BDT:
MaxDepth = 2
Ntrees = 1000

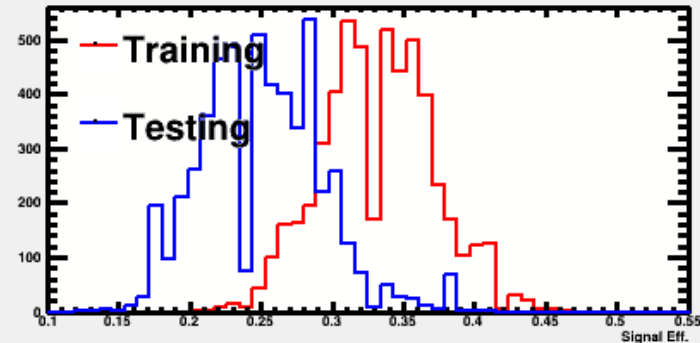
Background Rejection **99.9%**

0.1% bkg events of 6000 bkg total in test and training sample = 6 events only → train and test sample are highly “anti-correlated”

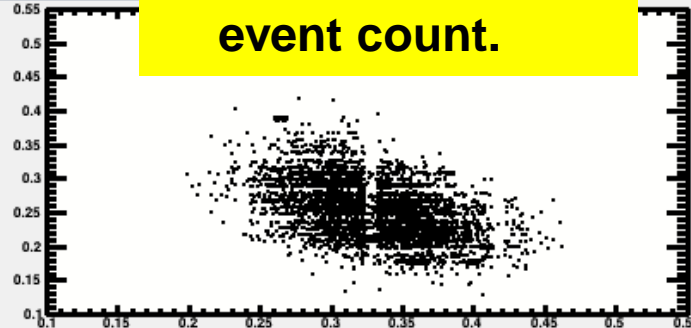
Binom err estimate EffS (at 0.001 bkg rejection)



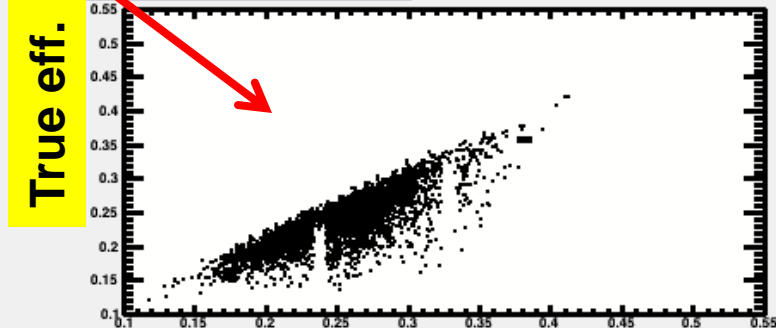
Training EffS (at 0.001 bkg rejection)



Test Vs Training Eff



True vs Test EffS (at 0.001 bkg rejection)



Test eff.

Example distributions

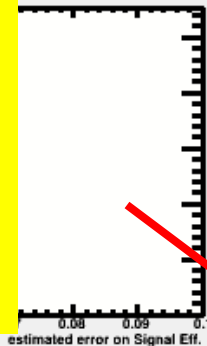
BDT:
MaxDepth = 2
Ntrees = 1000

Background Rejection **99%**

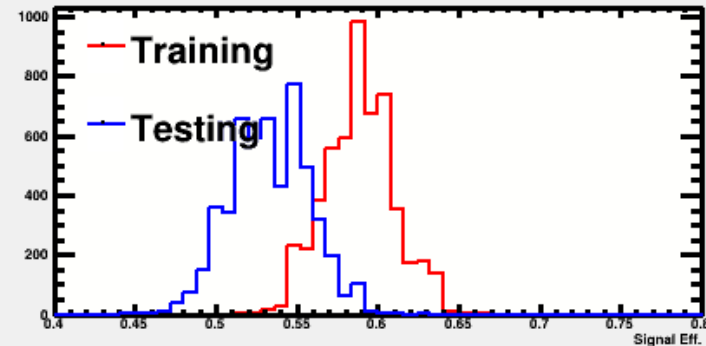
The correlation between test and training efficiency is still there (1% bkg events of 6000 bkg total in test and training sample = 60 events !!)

Binor

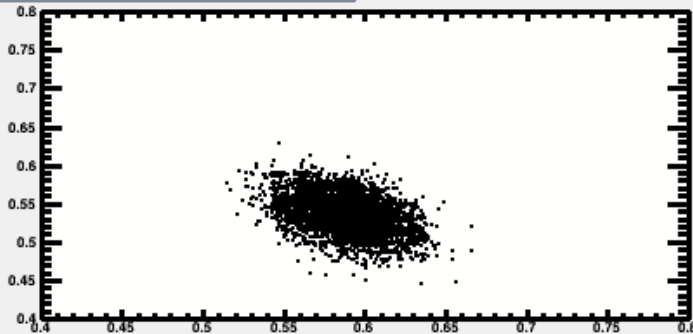
Bias between true and test eff. disappears with larger event counts, despite train-test correlation still visible



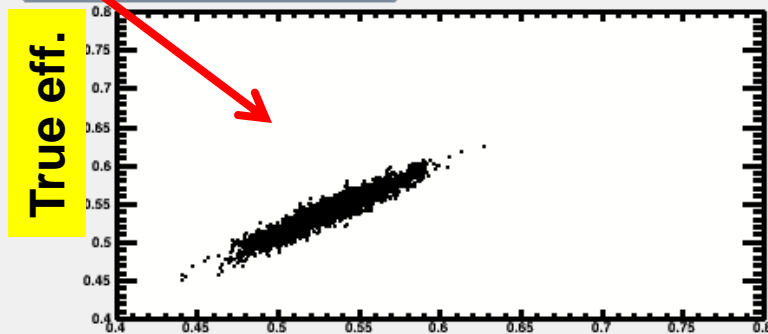
Training EffS (at 0.01 bkg rejection)



Test Vs Training EffS (at 0.01 bkg rejection)



True vs Test EffS (at 0.01 bkg rejection)



True eff.

Test eff.

Example distributions

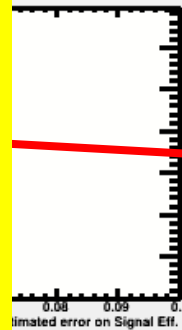
BDT:
MaxDepth = 2
Ntrees = 1000

Background Rejection **90%**

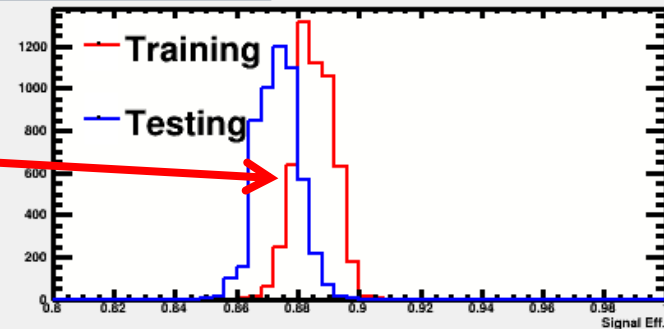
The correlation between test efficiency and training efficiency is almost gone, as the samples left and cut away are both big enough

Bir

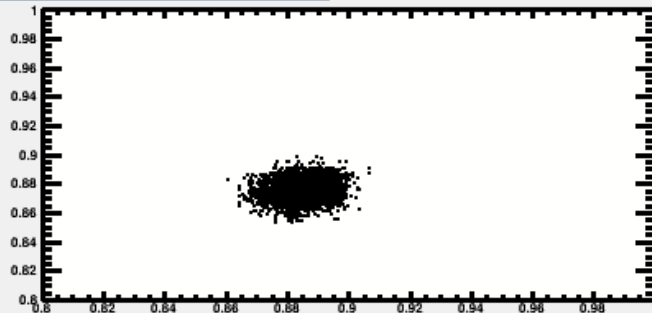
All correlations seem gone, we are only left with an ever smaller bias between true (test) and training eff.



Training EffS (at 0.1 bkg rejection)

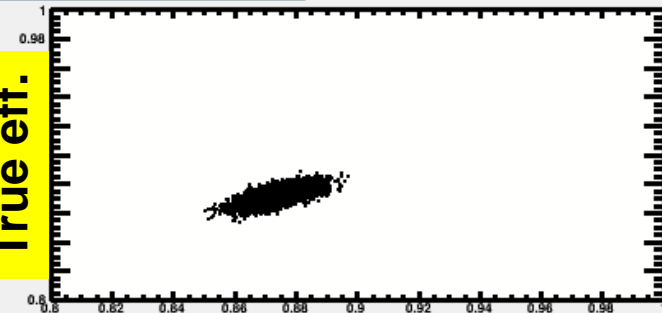


Test Vs Training EffS (at 0.1 bkg rejection)



True Vs Test EffS (at 0.1 bkg rejection)

True eff.



Test eff.

2006 ?

- **Endless discussions about possible ‘problems’ with MVA (systematic etc) show:**
- **We (physicists) are smart and try to really understand what we are doing**
- **But it also shows:**
 - **Sophisticated techniques are of course more difficult to grasp than ‘cuts’**
 - **To tap into the extremely valuable resource of pattern recognition techniques, the ‘physicist way’ of everyone re-inventing the wheel does not seem promising to me here.**

MVA-Literature /Software Packages... a 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

- attempts to provide “all inclusive” packages
 - TMVA: Höcker, Speckmayer, Stelzer, Therhaag, von Toerne, Voss, *arXiv: physics/0703039*
<http://tmva.sf.net> or every ROOT distribution (development moved from SourceForge to ROOT git repository)
 - WEKA: <http://www.cs.waikato.ac.nz/ml/weka/>
 - “R”: a huge data analysis library: <http://www.r-project.org/>
- “new Boosted Decision Tree library” <https://github.com/dmlc/xgboost> Rie Johnson and Tong Zhang. Learning nonlinear functions using regularized greedy forest, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(5):942-954, May 2014. [arXiv: 1109.0887v7.pdf](https://arxiv.org/abs/1109.0887v7)
- SciKit Learn <http://scikit-learn.org/stable/> Machine learning in Python
 - Theano: Popular for implementing deep learning neural networks
- CAFFE, Torch, You name them

Summary

- MVA's are great
- MVA's are widely used in HEP
- MVA's are even “widelier” used outside HEP
- MVA's are complicated to understand and to code !
- MVA's and work thereon still is not ‘funded’ buy HEP like
“Detector/Accelerator development” for example is:

- note: before TMVA in ROOT, the majority of the HEP community only used/knew simple cuts which often perform much worse
 - significant improvement in physics reach (imagine how much a 20% better accelerator/detectors would cost?)
 - provide state of the art analysis tools for state of the art accelerator/detectors

→ And I think we should have a much larger concentrated effort to put HEP to ‘state of the art’ in pattern recognition, then this one paid TMVA position I was unsuccessfully asking for!

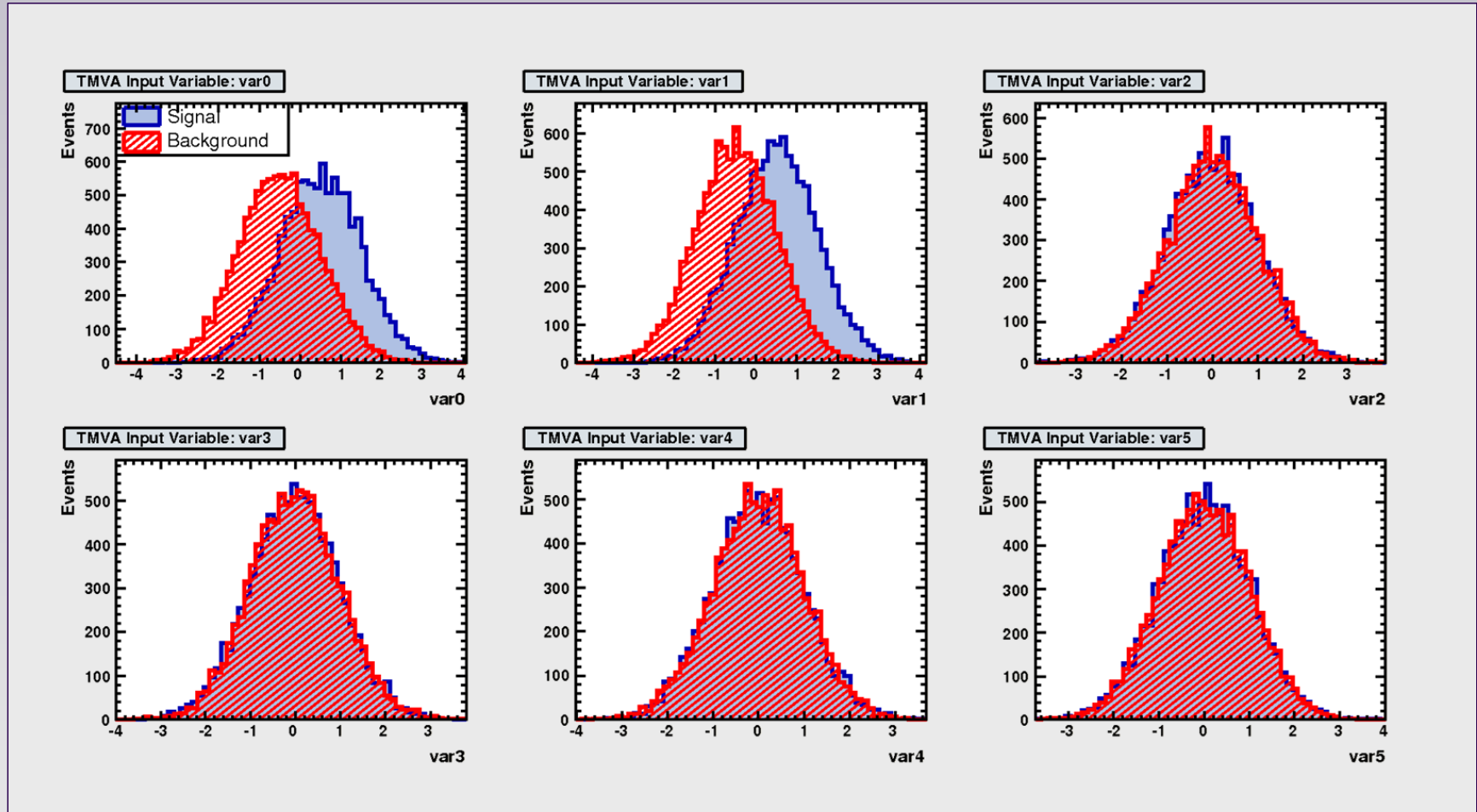
Summary

Linear classifiers : $y(x)$ = 'linear combination of observables "x" '
→ decision boundary ($y(x) = \text{const}$) is a linear hyperplane

Non linear classifiers: $y(x)$ = 'non-linear combination of observables "x" '
→ decision boundary ($y(x) = \text{const}$) is 'any kind of hypersurface'
→ parameterised e.g. "pieces of sigmoid functions"
→ Neural Network

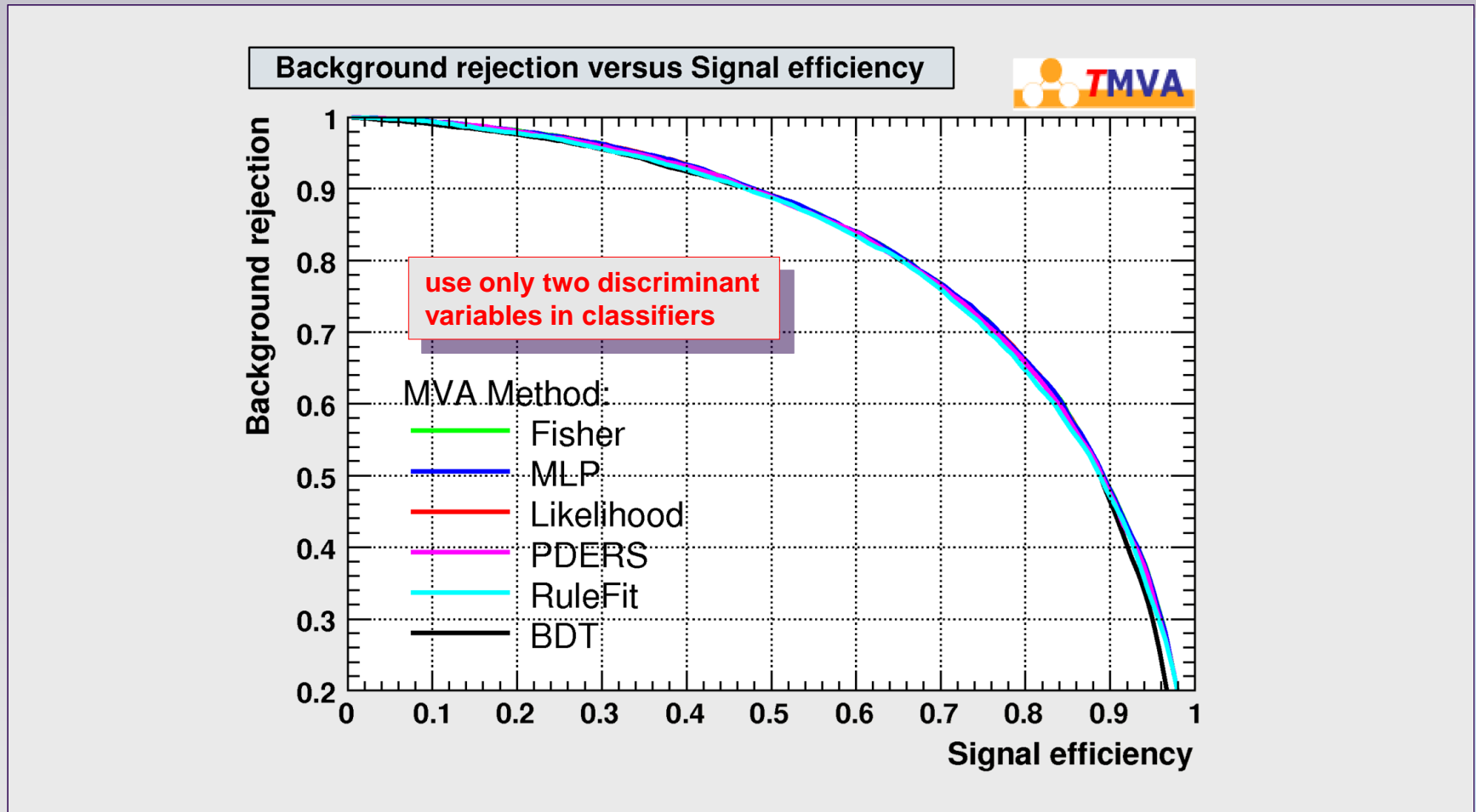
Stability with Respect to Irrelevant Variables

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



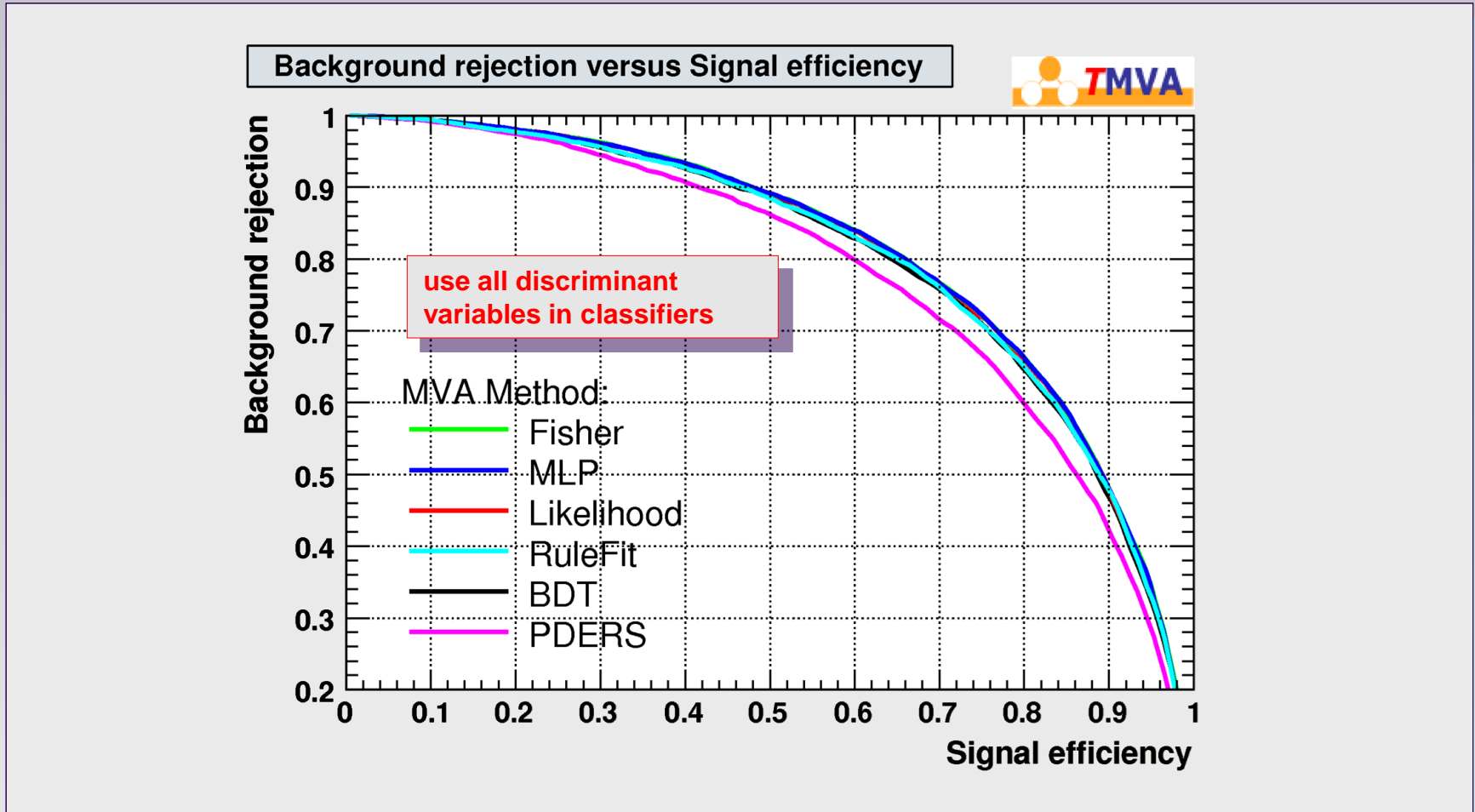
Stability with Respect to Irrelevant Variables

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



Stability with Respect to Irrelevant Variables

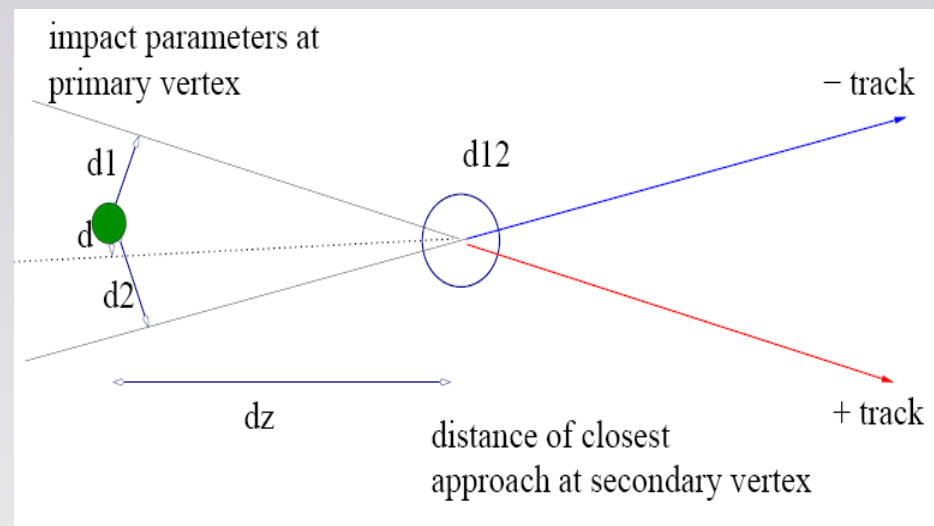
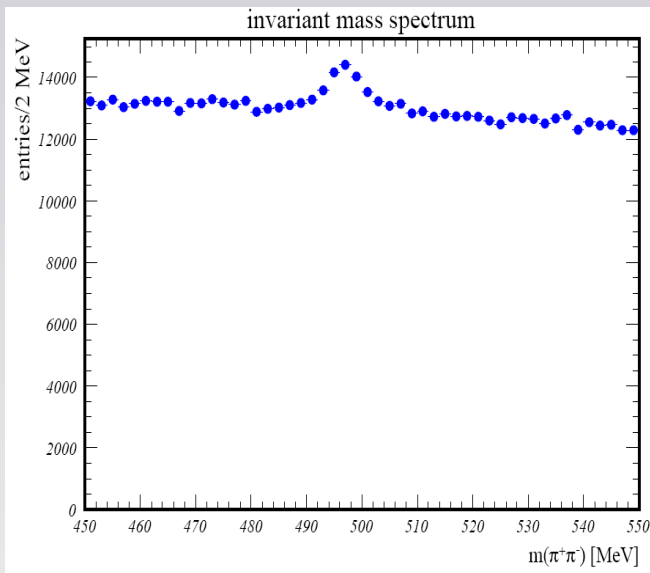
- Toy example with 2 discriminating and 4 non-discriminating variables ?



(following example taken from M. Schmelling)

A simple K-short selection:

- consider all opposite tracks with $450\text{GeV} < M_{\pi^-\pi^+} < 550\text{GeV}$
- reconstruct secondary vertex for those tracks
 - impact parameters d_i ($d_{\pi^+}, d_{\pi^-}, d = d_K$)
 - distance of closes approach between 2 tracks d_{ij}
 - distance : primary – secondary vertex d_z
- require the secondary vertex to lie “downstream” $Z_{\text{svtx}} - Z_{\text{pvtx}} > 0$



(following example taken from M. Schmelling)

→ criteria

- **informative**: good separation between signal and background
- **well behaved**: no “singularities” in the PDFs
 - avoid finetuning in placement of cuts
- **independent**: every variable contributes new information

❖ variables used in the following

A measure for impact parameter reduction

$$v_1 = \log_{10} \frac{d_1 d_2}{d}$$

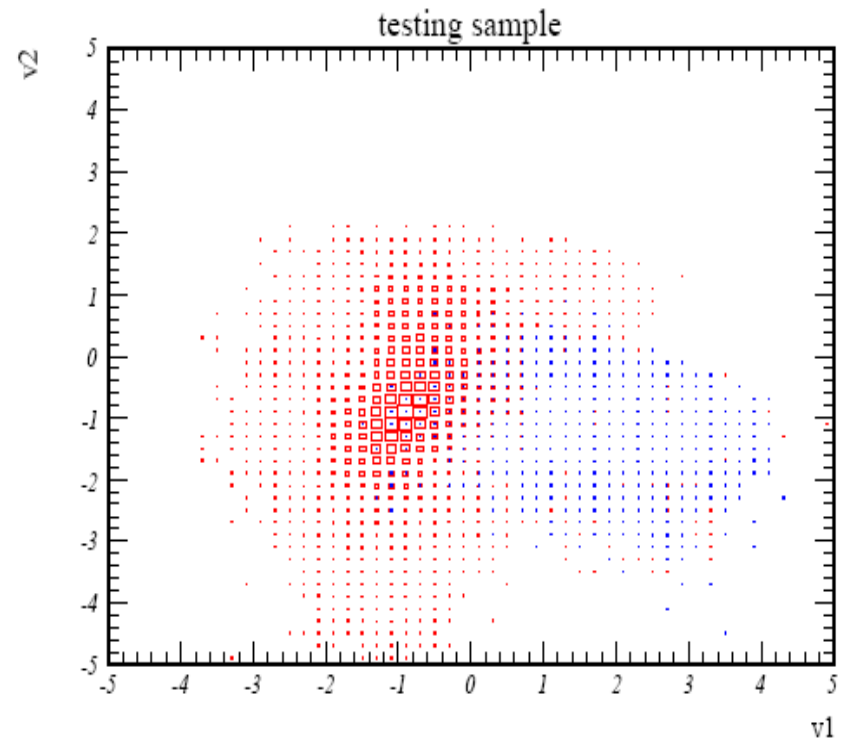
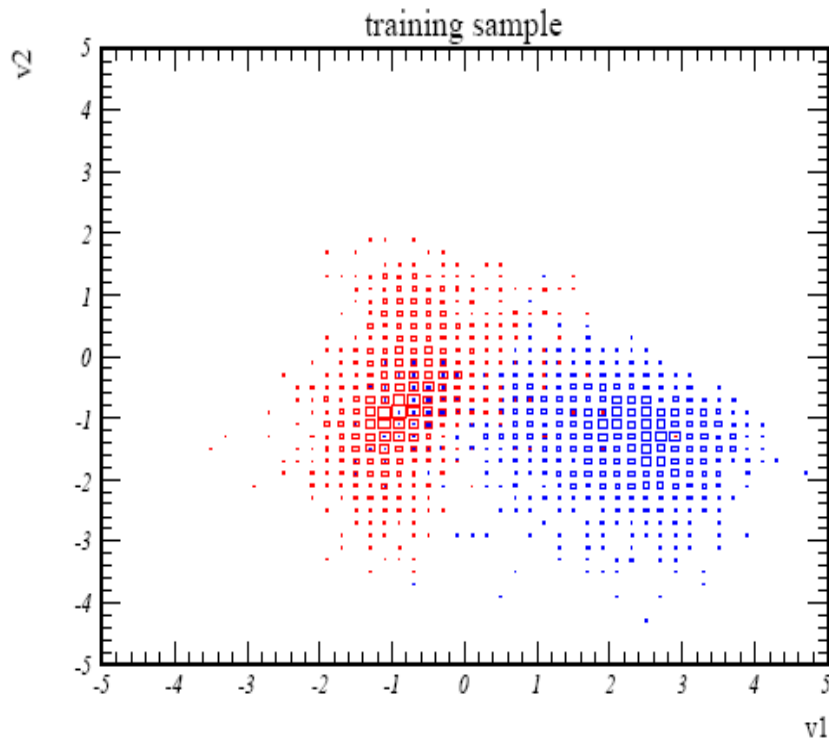
Quality of the secondary vertex

$$v_2 = \log_{10}(d_{12})$$

Lifetime of the particle (correlated with d_1 and d_2 , hence taken 3rd)

$$v_3 = \log_{10}(dz)$$

(following example taken from



- some number of signal and background candidates in training
 - same statistical power to estimate PDFs
- imbalanced samples in testing
 - gauge performance under realistic conditions

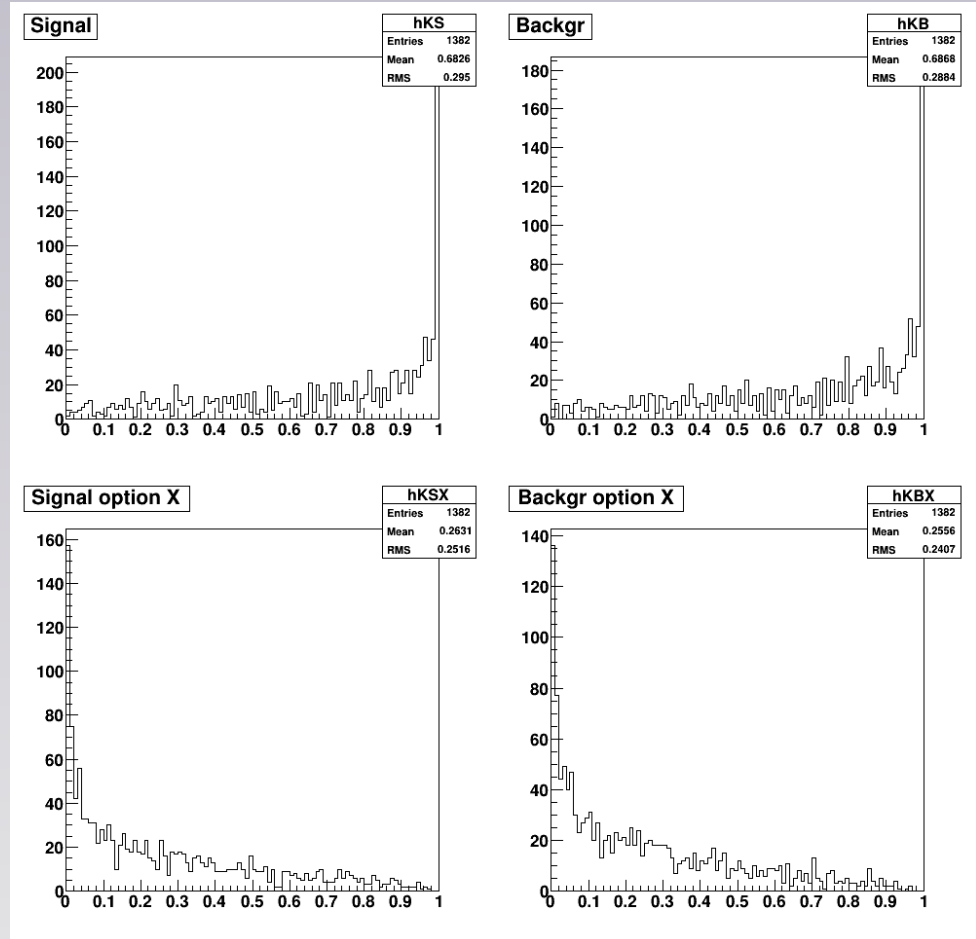
Kolmogorov Smirnov Test

Tests if two sample distributions are compatible with coming from the same parent distribution

- Statistical test: if they are indeed random samples of the same parent distribution, then the KS-test gives an uniformly distributed value between 0 and 1 !!
- Note: that means an average value of 0.5 !!
- How does that look like for TMVA, where many people use the KS test to check for “overtraining”

Kolmogorov Smirnov Test

Default (ROOT)



Setting option “X” (default in TMVA since 4.2)

→ Uses random sample generated from the two distributions that need to be compared

Distributions are not really “flat” → effect of using binned histograms (to be checked...)