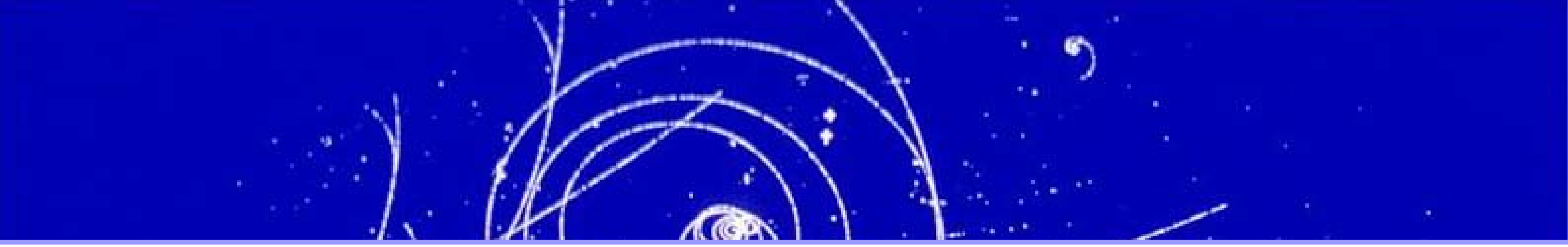
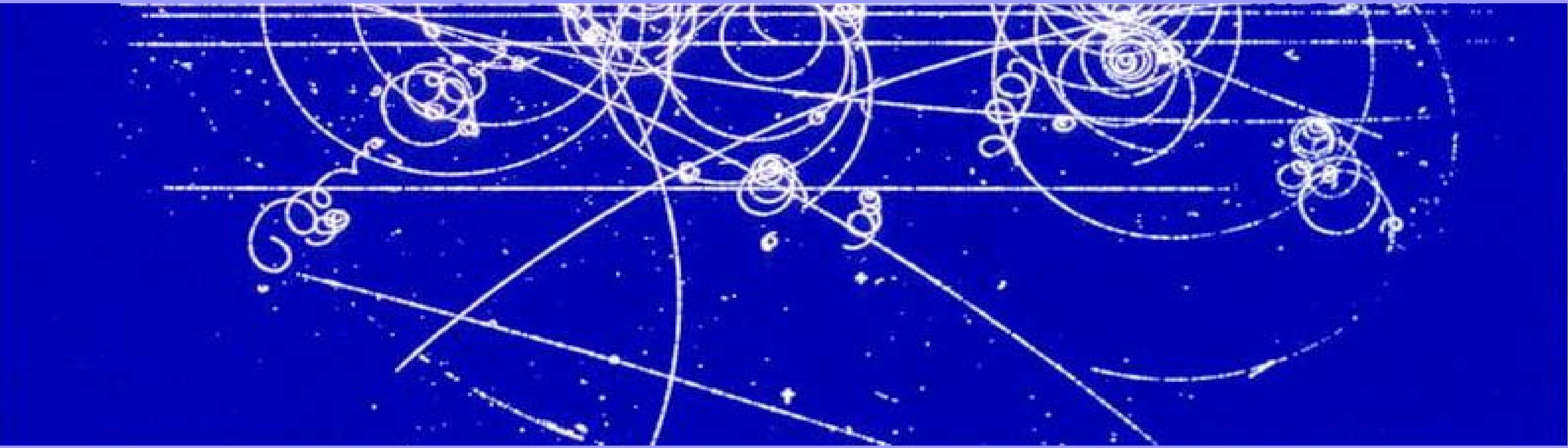


Cuts and Likelihood Classifiers in *T*MVA

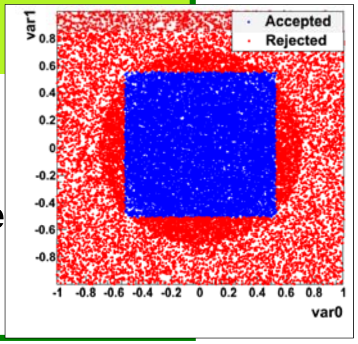
Jörg Stelzer – Michigan State University

TMVA Workshop 2011, CERN, Geneva, Switzerland, January 21st



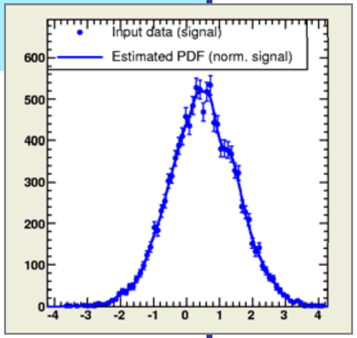
Cut and Likelihood Based Classifiers in TMVA

Rectangular Cut Optimization



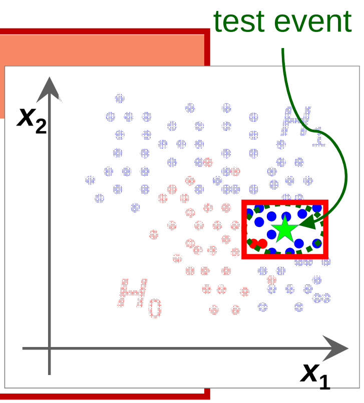
- Widely used because transparent
- Machine optimization is challenging:
  - MINUIT fails for large *n* due to sparse population of input parameter space
  - Alternatives are Monte Carlo Sampling, Genetic Algorithms, Simulated Annealing

Projective Likelihood Estimator



- Probability density estimators for each variable combined into one
- Much liked in HEP
  - Returns the likelihood of a sample belonging to a class
- Projection ignores correlation between variables
  - Significant performance loss for correlated variables

PDE Range-Search, k Nearest Neighbors, PDE Foam



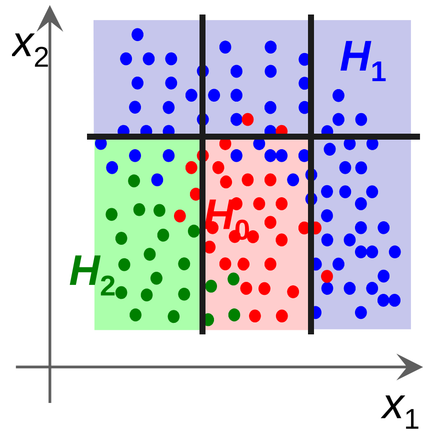
- n- dimensional signal and background PDF, probability obtained by counting number of signal and background events in vicinity of test event
  - Range Search: vicinity is predefined volume
  - k nearest neighbor: adaptive (k events in volume)

Rectangular Cut Optimization

- Classical method because simple and transparent

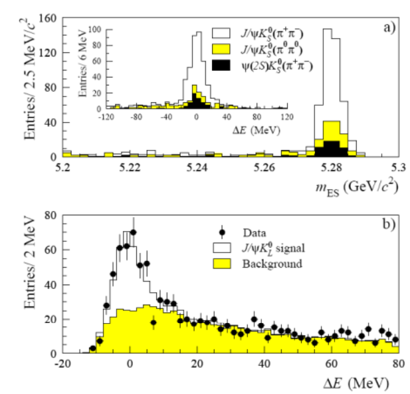
$$x_{\text{cut}}(i_{\text{event}}) \in \{0,1\} = \bigcap_{v \in \{\text{variables}\}} \left(x_v(i_{\text{event}}) \subset [x_{v,\min}, x_{v,\max}] \right)$$

  - Rectangular cuts best on independent variables



- Often the variables with separation power are not as independent as you wish.
  - →Transform variables before you try to cut on them
    - TMVA provides methods to linearly de-correlate or PCA transform input data (see Peters talk)
    - Apply some transformation that reflects the correlation in your data. E.g. at BABAR and Belle, two uncorrelated variables used to select candidates for B-mesons

$$m_{\text{ES}} = \sqrt{\left(E_{\text{beam}}^{\text{cm}}\right)^2 - \left(p_{\text{B}}^{\text{cm}}\right)^2} \quad \text{and} \quad \Delta E = E_{\text{B}}^{\text{cm}} - E_{\text{beam}}^{\text{cm}}$$



- How to find optimal cuts?
  - Human: look at the variables in one and two dimensions, sequentially in order of separation power.

How TMVA Finds the Optimal Cuts

- Three implemented methods to optimize the cut parameters
  - Monte Carlo sampling (MC)
    - Test the possible cuts in the variable phase space (random points)
  - Genetic algorithm (GA)
    - Biology-inspired optimization algorithm. **Preferred algorithm.**
  - Simulated annealing (SA)
    - slow “cooling” of system to avoid “freezing” in local solution

  - (MINUIT) standard minimizer in HEP, but …
    - Poor performance to find global

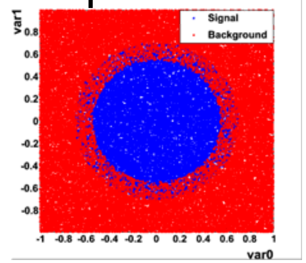
- All methods are basically trial and error.
  - Sample set of cuts across the phase space to find the best one
    - GA and SA have build-in sophistication about the trials they do.
  - Make use of computers data grinding power
  - Since they probe out the full phase space, they suffer with increasing number of dimensions
    - TMVA sorts the training events in a binary search tree, which reduces the training time substantially.

- Box search: ~ $(N_{events})^{Nvar}$
- BT search: ~ $N_{events} \cdot N_{var} \ln_2(N_{events})$

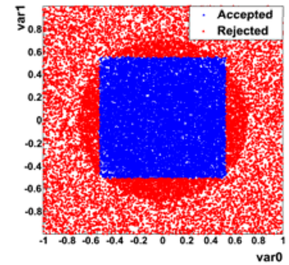
How MethodCuts Works

- MethodCuts finds a single signal box, ie a lower and upper limit for each variable.
  - Example of a 2-D Gaussian signal above a uniform background
  - It does not work on a checker board pattern. (There are not many variables in HEP with such a distribution though)

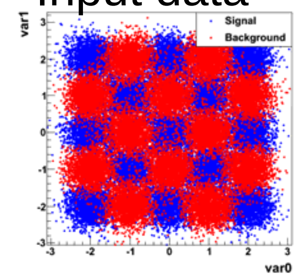
Input data



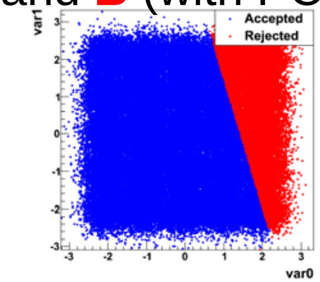
S and B(with SA)



Input data



S and B (with PCA)



- Unlike all other classifiers, which have one response function to be applied to an event, MethodCuts provides a different signal box definition for different efficiencies, the response is 0 or 1.

```
y_mva = reader->EvaluateMVA( vec<float>, "PDERS method" ); // usually [0,1]

passed = reader->EvaluateMVA( vec<float>, "CutsGA method", effS=0.7 ); // {0,1}
```

Weight file shows you which cuts are applied for a certain efficiency

```
<Bin ibin="76" effS="7.5e-01" effB="2.242e-02">
   <Cuts cutMin_0="-4.57e-01" cutMax_0="5.19e-01" cutMin_1="-5.26e-01" cutMax_1="5.56e-01" />
</Bin>
```

Details about the TMVA Minimizers

- 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
    - Option "**SampleSize=200000**", depends on dimensionality of the problem
  - Good global minimum finder, but poor accuracy

- Default solution in HEP: (T)Minuit/Migrad
  - Gradient-driven search
  - Poor global minimum finder, gets quickly stuck in presence of local minima

- Genetic Algorithm:
  - Inspired by biological principal of producing slight modifications of successful cuts. Most important parameter
    - Option "**PopSize=300**", could be increase to ~1000

- Simulated Annealing:
  - Avoids local minima by continuously trying to jump out of the these
    - "**InitialTemp=1e06**" and "**TempScale=1**" can be adjusted to increase performance

Likelihood based Classifiers in TMVA

- Basic feature of all LH based classifiers
  - Signal likelihood ratio as response function

$$y(\vec{x}) = \frac{P(X = \vec{x} \mid C = S)}{P(X = \vec{x} \mid C = S) + P(X = \vec{x} \mid C = B)} = \frac{1}{1 + f_B(x)/f_S(x)}$$

  - Training means building a data model for each class

- Two basic types
  - Projective Likelihood Estimator (Naïve Bayes)
    - Flavors of how to build the variable densities (PDFs)
  - Multidimensional Probability Density Estimators (PDEs)
    - Various ways to parcel the input variable space and weight the event contributions within each cell
    - Search trees are used to provide fast access to cells

Probability Density

Prior probability P(C)
Relative abundance of "class C" in the data

Likelihood PDF P(x|C)
Probability density distribution of **x** in "class C"

Posterior probability P(C|x)
probability that the observed event is of class C, given the measured observables **x**={x_1,….,x_D}

$$P(C\,|\,\vec{x}) = \frac{P(C)\times P(\vec{x}\,|\,C)}{P(\vec{x})}$$

Evidence P(x)
probability density to observe the actual measurement y(**x**)

- For signal classification:

$$P(C=S\,|\,X=\vec{x}) = \frac{N_S f_S(\vec{x})}{N_S f_S(\vec{x}) + N_B f_B(\vec{x})}$$

  - We can't answer *P(C=S|X=x)*, since we don't know the true numbers N_S and N_B of signal and background events in the data.
  - Confidence of classification only depends on $f_S(x)/f_B(x)$! Remember that the ROC curve also does not include knowledge about class sizes.

Projective Likelihood Estimator (Naïve Bayes)

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

Likelihood ratio for event i_{event}

PDFs

discriminating variables

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

Species: signal, background types

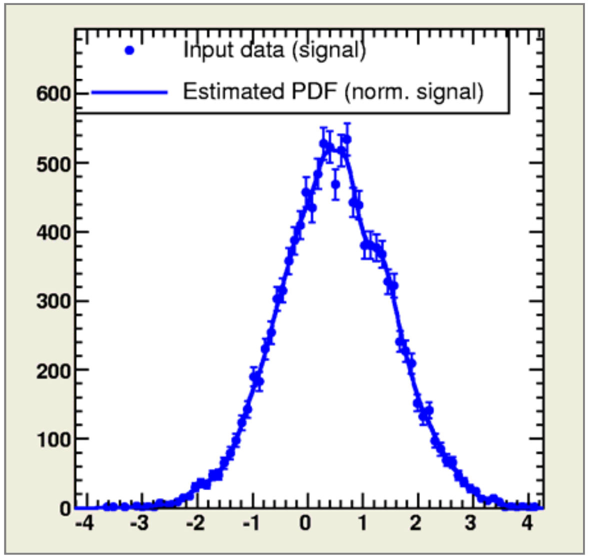
- Naïve assumption about independence of all input variables
  - Optimal approach if correlations are zero (or linear → decorrelation)
  - Otherwise: significant performance loss

- Advantages:
  - independently estimating the parameter distribution alleviates the problems from the "*curse of dimensionality*"
  - Simple and robust, especially in low-D problems

Building the PDF

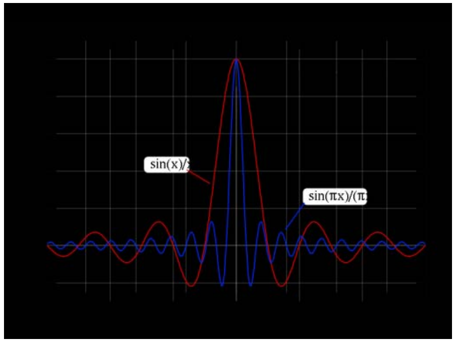
- Technical challenge: estimating the PDF of the input variables. Three ways:
  - Parametric fitting: excellent if the variable distribution function is known (in this case use RooFit package). Cannot be generalized to a-priori unknown problems.
  - Non-parametric fitting: easy to automate, but can create artifacts (edge effects, outliers) or hide information (smoothing) and hence might need tuning.
  - Event counting: unbiased PDF (histogram), automatic. Sub-optimal since it exhibits details of the training sample.

- TMVA uses nonparametric fitting
  - Binned shape interpolation using spline functions or adaptive smoothing
    - Option “PDFInterpol[2]=KDE” or “=Spline3”
  - Unbinned adaptive kernel density estimation (KDE) with Gaussian smearing
  - TMVA performs automatic validation of goodness-of-fit
    - Option “CheckHistSig[2]=1”

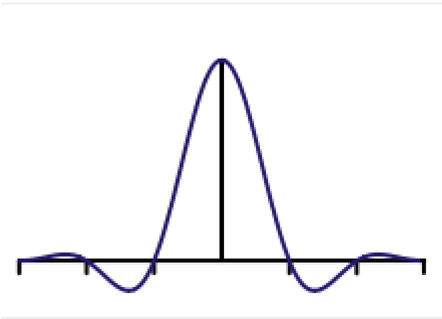


Multi-Dimensional PDE (Range-Search)

- Use a single, *n*-dimensional PDF per event class (S, B), $n = N_{var}$.
- PDE Range-Search:
  - Count number of signal and background events in "vicinity" of test event → preset or adaptive rectangular volume defines "vicinity"
  - Improve y_{PDERS} estimate within volume by using various N_{var}-D kernel estimators



sinc(x)= sin(x)/x



LanczosX(x)= sinc(x)/sinc(x/X)

VolumeRangeMode	Adaptive	Method to determine volume size [Unscaled, MinMax, RMS, Adaptive, kNN]
KernelEstimator	Box	Kernel estimation [Box, Sphere, Teepee, Gauss, Sinc, LanczosX, Trim]
Controls for the size and complexity of the volumes …		

Configuration parameters

Multi-Dimensional PDE (kNN)

- 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 *k*d-tree event sorting (training)
    - *k*d-tree is a binary search tree that sorts objects in space by their coordinates

For evaluation = event building **…**		
nkNN	20	Number of k-nearest neighbors
UseKernel	False	Use kernel
Kernel	Gaus	Use polynomial (=Poln) or Gaussian (=Gaus) kernel
UseWeight	True	Use weight to count kNN events
SigmaFact	1	Scale factor for sigma in Gaussian kernel
For training = kd-tree building **…**		
ScaleFrac	0.8	Fraction of events used to compute variable width
Trim	False	Use equal number of signal and background events
BalanceDepth	6	Binary tree balance depth

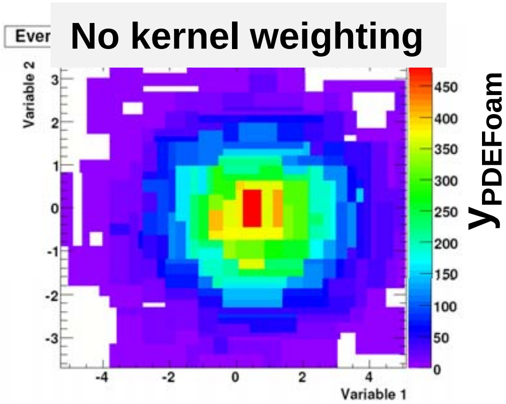
Configuration parameters

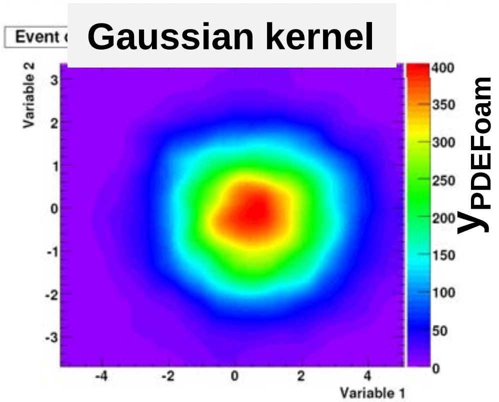
Multi-Dimensional PDE (Foam)

- Parcel phase space into cells of varying sizes, each cell represents the average of the neighborhood.

$$y_{\rm PDEFoam}(i) = \frac{n_{\rm sig}/V_{\rm sig}}{\frac{n_{\rm bg}}{V_{\rm bg}}\frac{N_{\rm sig}}{N_{\rm bg}} + \frac{n_{\rm sig}}{V_{\rm sig}}}$$

  - Evaluation can use kernels to determine response
  - Advantage over PDERS is the limited number of cells, independent of number of training events
  - Different parceling for signal and background possible, in case S and B distributions are very different.
  - Regression with multiple targets possible





SigBgSeparate	False	Separate foams for signal and background
Kernel	None	Kernel type used for calculating cell densities [None, Gauss, LinNeighbors]
DTLogic	None	Use decision tree algorithm to split cells [None, GiniIndex, MisClassificationError, CrossEntropy]

Controls for the size and complexity of the foam **…**
Weight treatment **…**
Regression **…**

Configuration parameters

Concluding Remarks on Cuts and Likelihoods

Criteria		Classifiers		
		Cuts	Likeli- hood	PDERS / k-NN
Perfor- mance	no / linear correlations	😐	☺	☺
	nonlinear correlations	😐	☹	☺
Speed	Training	☹	☺	☺
	Response	☺	☺	☹/😐
Robust -ness	Overtraining	☺	😐	😐
	Weak input variables	☺	☺	☹
Curse of dimensionality		☹	☺	☹
Clarity		☺	☺	😐

- Cuts and Likelihood are transparent, so if they perform (not often the case) use them (think about transforming variables first)

- In presence of correlations other, multi-dimensional, classifiers are better
  - Correlations are difficult to visualize and understand at any rate, no need to hang on to the transparency of Cuts and 1D LH

- Multivariate classifiers are no black boxes, we just need to understand the underlying principle