PDE-Foam

How does this fit work?

PDE-Foam divides the multi-dimensional phase space in a finite number of cells of constant event density. (“Foam”)

“Foam” of cells are filled with averaged probability density information from training data

The binning algorithm adjusts the size and position of the cells inside the phase space to minimize the variance of the event density in the cells

Description of Foam Algorithm

Setup of binary search trees

- A binary search tree is created and filled with the d-dimensional event tuples from the training sample

Initialization phase

- A foam is formed

Growing phase

- A binary splitting algorithm splits cells of the foam until the maximum number of active cells is reached.

Filling phase

- Each active cell is filled with values that classify the event distribution

Evaluation phase

- The estimator for a given event is evaluated based on the information stored in the foam cells.

Tools Available

TargetSelection

- Target selection method (Default: mean)

TailCut

- Tailors training ensemble to not include outliers of the distributions

UseYesNoCell

- If false, discriminant is returned as classification output
- If true, -1 is returned for background, +1 is returned for signal

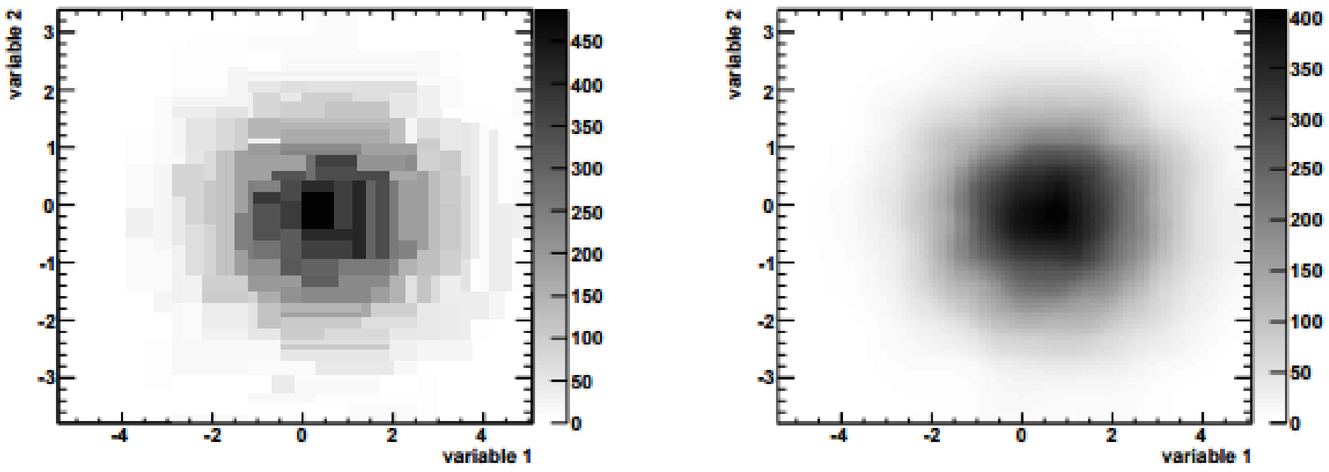
MaxDepth

- The cell tree depth can be limited by using this option.

Additional Tools

Kernel

- Smoothing Process applied in the evaluation phase. In most cases it results in an improved separation power between signal and background.



(a) foam projection without kernel

(b) foam projection with Gaussian kernel

Comprehensive Chart for Available Tools

Option	Classification		Regression	
	Separated foams	Single foam	Mono target	Multi target
`SigBgSeparate`	True	False	–	–
`MultiTargetRegression`	–	–	False	True
`Kernel`	•	•	•	•
`TargetSelection`	∘	∘	∘	•
`TailCut`	•	•	•	•
`UseYesNoCell`	•	•	∘	∘
`MaxDepth`	•	•	•	•

Table 5: Availability of options for the two classification and two regression modes of PDE-Foam. Supported options are marked by a '•', while disregarded ones are marked by a '∘'.

What scenarios is it best at describing?

This method is a powerful classification tool for problems with highly non-linearly correlated observables

K-Nearest Neighbor

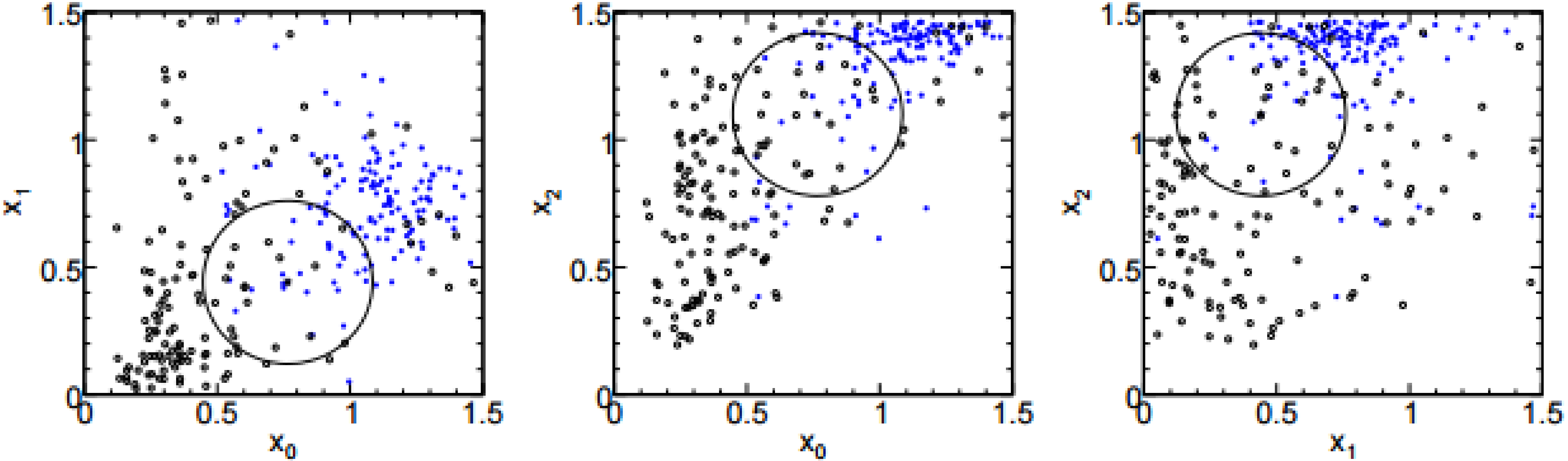
How does this fit work?

The k-NN algorithm searches for k events that are closest to the test event

It then searches for a fixed number of adjacent events, which define a volume

The output of the k-nearest neighbor algorithm can be interpreted as a probability that an event is of signal type, but only if the number of signal and background events are equal in the training sample

k-nearest neighbour algorithm in a three-dimensional space



Blue dots: Signal
Black dots: Background
Star (center of circle): query point to find k-nearest neighbors around

Configuration Tools

Trim

- Training events of the overabundant type are randomly removed until parity is achieved.

nkNN

- Number of k-nearest neighbors (Default 20)
- (The value of k determines the size of the neighborhood for which a probability density function is evaluated.)

What scenarios is it best at describing?

The k-NN classifier has best performance when the boundary that separates signal and background events has irregular features that cannot be easily approximated by parametric learning methods

The TMVA implementation of the k-NN method is reasonably fast to allow classification of large data sets. In particular, it is faster than the adaptive PDE-RS method

A large training set allows the algorithm to probe small-scale features that distinguish signal and background events.