CERN Summer Student Project 2008
Likelihood estimator using self-adapting phase-space binning (PDEFoam)

Alexander Voigt*
Technische Universität Dresden

Supervised by
Tancredi Carli and Andreas Höcker

September 25, 2008

Abstract

The PDEFoam method is an extension of the PDERS algorithm [2] used for multi-variate probability density estimation. It divides a multi-dimensional phase space in a finite number of hyper-rectangles (cells) of constant event density. This "foam" of cells is filled with averaged probability-density information sampled from a training event sample. For a given number of cells, the binning algorithm adjusts the size and position of the cells inside the multidimensional phase space based on a binary-split algorithm, minimizing the variance of the event density in the cell. The binned event density information of the final foam is stored in cells, organised in a binary tree, allowing for a fast and memory-efficient storage and retrieval of the event density information necessary for the classification of events.

Contents

1 Introduction

The implementation of PDEFoam within TMVA, a toolkit for multi-variate analysis (see ref. [4]), is based on the Monte-Carlo integra-

*Can be reached via Alexander.Voigt2@gmx.de

tion package TFoam [5] included in the analysis package ROOT [1].

The PDEFoam method (see ref. [3]) can be used for classification of signal and background events. In this case the algorithm makes bins of constant density of events of signal and background events or the ratio of signal over background. Furthermore, it can be used to reconstruct event variables. In this case the algorithm provides cells in which the event quantity to be reconstructed (target) is constant. In the following, we use the term density (ρ) for the event density in case of classification or for the target variable density in case of regression.

2 Description and implementation of the foam algorithm

2.1 Foam build-up algorithm

The general foam build-up from an arbitrary event sample works in the following way:

1. *Setup of binary search trees*: A binary search tree is created and filled with the D-dimensional observable vectors from the given event sample as it is done in the PDERS method (see ref. [2]).

2. *Initialisation phase*: A foam is created, which at first consists of one D-dimensional hyper-rectangle (base cell). The coordinate system of the foam is normalised such that the base cell extends from 0 to 1 in each dimension. The coordinates of the events in the corresponding training tree are linearly transformed into the coordinate system of the foam.

3. *Growing phase*: A binary splitting algorithm iteratively splits cells of the foam along hyperplanes until the maximum number of cells, set by the parameter `nCells`, is reached. The splitting algorithm minimizes the relative variance of the density $\sigma_\rho/\langle\rho\rangle$ across each cell (see ref. [5]). For each cell `nSampl` random points uniformly distributed over the cell volume are generated. For each of these points a small box centered around this point is defined. The box has a size of `VolFrac` times the size of the base cell in each dimension. The density is estimated as the number of events found in the binary search tree that are contained in this box divided by the volume of the box[1]. The obtained densities for all sampled points in the cell are projected on the D axes of the cell and the projected values are filled in histograms with `nBin` bins. The cell to be split next and the corresponding division edge (bin) for the split are selected as the ones that have the largest relative variance. The two new daughter cells are marked as 'active' cells and the old mother cell is marked as 'inactive'. A detailed description of the splitting algorithm can be found in ref. [5]. The geometries of the final foams reflect the distribution of the training samples: Phase-space regions where the density is constant are combined in large cells, while in regions with large gradients in density many small cells are created. In Fig. 1(a) a foam, based on training events distributed according to a 2-dimensional Gaussian distribution, is shown.

4. *Filling phase*: Each active cell is filled with values, which classify the event distribution within this cell and allow the later calculation of i.e. the discriminator (see classification and regression methods for more details). The total number of cells, `nCells`, as well as the normalisation constant $N_{\mathrm{sig}}/N_{\mathrm{bg}}$ (needed in case of signal and background samples of different size) calculated from the total number of

[1] or in case of regression the average target divided by the box volume

signal and background training events are also stored with the PDEFoam object.

5. *Evaluation phase*: The estimator for a given event is evaluated based on the information stored in the foam cells. The corresponding foam cell, in which the event variables (D-dimensional vector) of a given event are contained, are found using a binary search algorithm[2].

The initial trees, which contain the training events, needed to evaluate the densities for the foam build-up, are discarded after the training phase. The memory consumption for the foam is 160 bytes per foam cell plus an overhead of 1.4 kbytes for the PDEFoam object on a 64 architecture. Note that in the foam all cells created during the growing phase are stored within a binary tree structure. Cells which have been split are marked as inactive and remain empty. To reduce memory consumption, the geometry of a cell is not stored with the cell, but rather obtained recursively from the information about the division edge of the corresponding mother cell. This way only two short integer numbers per cell contain the information about the entire foam geometry: the division coordinate and the bin number of the division edge.

2.2 Summary

The PDEFoam algorithm has a few parameters to be optimised to the specific problem. However, the variation of the performance on the exact values of these parameters is not so large. The most important parameters to be optimised, are the minimal number of events in a cell and the total number of cells. The recommended strategy is to set the number of cells to a large value and the minimal number of events to values between $10-100$ depending on the size of the training sample. The other parameters can remain at their default values.

[2]For events outside the foam boundaries, the cells with the smallest cartesian distance to the event are chosen.

3 Classification

To classify an event in a D-dimensional phase space as being either of signal or of background type, a local estimator of the probability that this event belongs to either class can be obtained from the foam's hyper-rectangular cells. The foams are created and filled based on samples of signal and background training events. For classification two possibilities are implemented. One foam can be used to separate the S/B probability density or two separate foams are created, one for the signal events and one for background events.

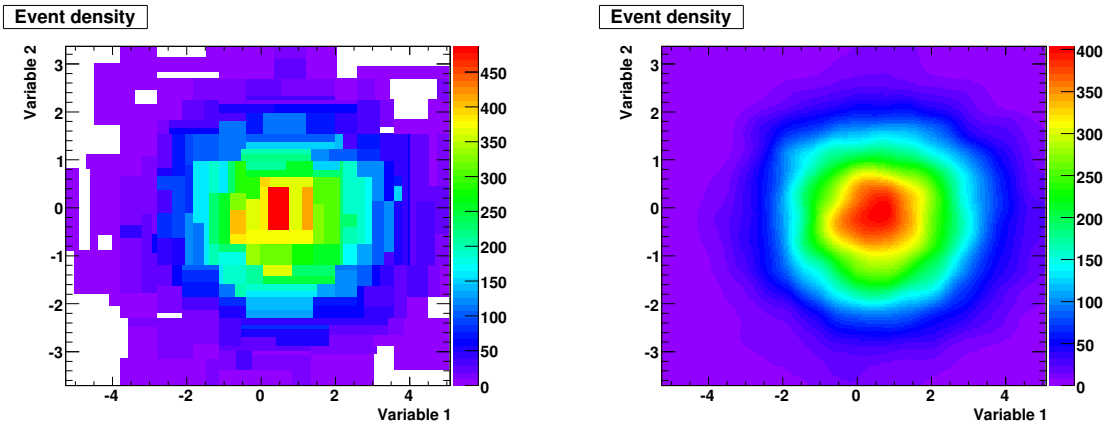
3.1 Separate Signal and Background Foams

If the option `SigBgSeparate = True` is set (default), the method PDEFoam treats the signal- and background distributions separately and performs the following steps to build the two foams and to calculate the classifier discriminator for a given event:

1. *Setup of training trees*: Two binary search trees are created and filled with the D-dimensional observable vectors of all signal and background training events, respectively.

2. *Initialisation phase*: Two independent foams for signal and background are created.

3. *Growing phase*: The growing is performed independently for the two foams. The density of events is estimated as the number of events found in the corresponding tree that are contained in the sampling



(a) foam projection without kernel

(b) foam projection with Gaussian kernel

Figure 1: Projections of a 2-dimensional foam with 500 cells for a Gaussian distribution on a 2-dimensional histogram. The foam was created with 5000 events from the input tree. (a) shows the reconstructed distribution without using kernel weighting and (b) shows the distribution with a Gaussian kernel. The colors indicate the event density of the drawn cell.

box divided by the volume of the box (see `VolFrac` option). The geometries of the final foams reflect the distribution of the training samples: Phase-space regions where the density of events is constant are combined in large cells, while in regions with large gradients in density many small cells are created.

4. *Filling phase*: Both for the signal and background foam each active cell is filled with the number of training events contained in the corresponding cell volume.

5. *Evaluation phase*: The estimator for a given event is evaluated based on the number of events stored in the foam cells. The two corresponding foam cells that contain the event are found. The number of events (n_{sig} and n_{bg}) is read from the cells. The estimator $y_{\mathrm{PDEFoam}}(i)$ is then given as

$$y_{\mathrm{PDEFoam}}(i) = \frac{n_{\mathrm{sig}}/V_{\mathrm{sig}}}{\frac{n_{\mathrm{bg}}}{V_{\mathrm{bg}}}\frac{N_{\mathrm{sig}}}{N_{\mathrm{bg}}} + \frac{n_{\mathrm{sig}}}{V_{\mathrm{sig}}}}\,, \qquad (1)$$

where V_{sig} and V_{bg} are the respective cell volumes and N_{sig}, N_{bg} are the total number of signal and background training events.

Steps 1-4 correspond to the training phase of the method. Step 5 is performed during the testing phase. In contrast to the PDERS method the memory consumption and computation time for the testing phase does not depend anymore on the number of training events, but only on the number of foam cells, `nCells`.

The described implementation with two separate foams for signal and background allows the foam algorithm to adapt the foam geometries to the individual shapes of the signal and background event distributions. It is therefore

well suited for cases where the shapes of the two distributions are very different.

3.2 Single Signal and Background foam

If the option `SigBgSeparate = False` is set, the PDEFoam method creates only one foam, which holds directly the estimator instead of the number of signal and background events. The differences with respect to the default method for are:

1. *Setup of training trees*: Fill only one binary search tree with both signal events and background events. This is possible, since the binary search tree has the information whether a event is of signal or background type.

2. *Initialisation phase*: Only one foam is created. The cells of this foam will contain the estimator $y_{\mathrm{PDEFoam}}(i)$ (see eq. (1)).

3. *Growing phase*: The splitting algorithm in this case minimizes the variance of the estimator density $\sigma_\rho/\langle\rho\rangle$ across each cell. The estimator density ρ is sampled as the number of signal events n_{sig} over the total number of events $n_{\mathrm{sig}} + n_{\mathrm{bg}}$ in a small box around the sampling points, normalised to the total number of signal and background events in the training sample:

$$\rho = \frac{n_{\mathrm{sig}}}{n_{\mathrm{sig}} + n_{\mathrm{bg}} \frac{N_{\mathrm{sig}}}{N_{\mathrm{bg}}}} \frac{1}{\texttt{VolFrac}} \qquad (2)$$

   In this case the geometries of the final foams reflect the distribution of the estimator density in the training sample: Phase-space regions where the signal to background ratio is constant are combined in large cells, while in regions where the signal-to-background ratio changes rapidly many small cells are created.

4. *Filling phase*: Each active cell is filled with the estimator given as the ratio of signal events contained in the cell to the total number of events in the cell and normalised to the total number of signal and background evens in the training tree:

$$y_{\mathrm{PDEFoam}}(i) = \frac{n_{\mathrm{sig}}}{n_{\mathrm{sig}} + n_{\mathrm{bg}} \frac{N_{\mathrm{sig}}}{N_{\mathrm{bg}}}}. \qquad (3)$$

   The statistical error of the estimator also is stored in the cell.

5. *Evaluation phase*: The estimator for a given event is directly obtained as the discriminator that is stored in the cell which contains the event. (For events outside the foam boundaries, the cells with the smallest cartesian distance to the event are chosen.)

Having only one foam instead of two reduces the total number of cells and therefore the memory consumption by about a factor of two. The single foam can give better results if the distributions for signal and background have similar shapes. Furthermore, it is less sensitive to steep gradients in absolute event numbers that affect both the signal and background distributions. However, for most studied examples, the default method with two foams resulted in a better estimator performance.

4 Regression

The foam can also be used to reconstruct event quantities (regression). Two different ways are possible: First, one uses the possibility of the foam to save the target value in every foam cell. Second, one saves the target values in further foam dimensions. Since the first method can only be used if one target is given, it is called 'Mono target regression'. In order to do regression with multiple targets one has to use

the second method, called 'Multi target regression'. Note, that the second method also can be used if only one target is given.

4.1 Mono target regression

In this method, the density, used for the foam build up, is given by the mean target density in a given box. Let us assume, that an input event has the form $\vec{x} = (x_1, \ldots, x_{n_{\text{var}}}, t)$, consisting of n_{var} variables x_j $(j = 1, \ldots, n_{\text{var}})$ and one target t.

In particular, the Mono target regression now works in the following way:

1. Fill one binary search tree with all training events.

2. Build up one n_{var}-dimensional foam: The density ρ, needed by the foam during its buildup, is given by the mean target value $\langle t \rangle$ within the sampling box, divided by the box volume (given by the `VolFrac` option):

$$\rho = \frac{\langle t \rangle}{\texttt{VolFrac}} \equiv \frac{\sum_{i=1}^{N_{\text{box}}} t^{(i)}}{N_{\text{box}} \times \texttt{VolFrac}}\,, \quad (4)$$

   whereas the sum goes over all events N_{box} within the sampling box and $t^{(i)}$ is the target value of the event $\vec{x}^{(i)}$ $(i = 1, \ldots, N_{\text{box}})$.

3. Fill every foam cell with the average target value $\langle t \rangle = \sum_{i=1}^{N_{\text{box}}} t^{(i)} / N_{\text{box}}$.

4. Estimate the target value, when an event $\vec{x} = (x_1, \ldots, x_{n_{\text{var}}})$ is given: Find the corresponding foam cell in which $\vec{x}$ is situated and read the average target value $\langle t \rangle$ from the cell.

4.2 Multi target regression

In order to do regression with more than one target, we put the information about all targets into further dimensions of the foam. Assume an event $\vec{x} = (x_1, \ldots, x_{n_{\text{var}}}, t_1, \ldots, t_{n_{\text{tar}}})$ consists of n_{var} variables and n_{tar} targets. Then we can build an $(n_{\text{var}} + n_{\text{tar}})$-dimensional foam, which cells contain numbers of events.

In order to finally calculate the targets, we need the coordinates of the cell center in every foam dimension. Thus, may $\text{CC}(i, k)$ be the coordinate of the cell center of cell i in the kth dimension. Note, that the event variables occupy the dimensions $k = 1, \ldots, n_{\text{var}}$, and the targets occupy the dimensions $k = n_{\text{var}} + 1, \ldots, n_{\text{var}} + n_{\text{tar}}$.

To build the foam and read out the targets, the following steps are done:

1. Fill one binary search tree with all training events.

2. Build up one $(n_{\text{var}} + n_{\text{tar}})$-dimensional foam: The event density ρ, needed by the foam during its buildup, is estimated by the number of events N_{box} within a predefined box of the dimension $(n_{\text{var}} + n_{\text{tar}})$, divided by the box volume, whereas the box volume is given by the `VolFrac` option

$$\rho = \frac{N_{\text{box}}}{\texttt{VolFrac}}\,. \quad (5)$$

3. Fill every foam cell with the number of corresponding training events.

4. Estimate the target value, when an event $\vec{x} = (x_1, \ldots, x_{n_{\text{var}}})$ is given: Find all corresponding foam cells N_{cells} in which the coordinates $(x_1, \ldots, x_{n_{\text{var}}})$ of the event vector are situated. Depending on the `TargetSelection` option, the target value t_k $(k = 1, \ldots, n_{\text{tar}})$ is

   (a) the coordinate of the cell center in direction of the target dimension $n_{\text{var}} + k$ of the cell j $(j = 1, \ldots, N_{\text{cells}})$,

which has the maximum event density

$$t_k = \mathrm{CC}(j, n_{\mathrm{var}} + k)\,, \qquad (6)$$

if `TargetSelection = Mpv` is set.

(b) the mean cell center in direction of the target dimension $n_{\mathrm{var}} + k$ weighted by the event densities $d_{\mathrm{ev}}(j)$ $(j = 1, \ldots, N_{\mathrm{cells}})$ of the cells

$$t_k = \frac{\sum_{j=1}^{N_{\mathrm{cells}}} \mathrm{CC}(j, n_{\mathrm{var}} + k) \times d_{\mathrm{ev}}(j)}{\sum_{j=1}^{N_{\mathrm{cells}}} d_{\mathrm{ev}}(j)} \qquad (7)$$

if `TargetSelection = Mean` is set.

5 Performance

Like PDERS (see ref. [2]), this method is a powerful classification tool for problems with highly non-linearly correlated observables. Furthermore PDEFoam is a fast responding classifier, because of its limited number of cells, independent of the size of the training samples.

An exception is the Multi target regression with Gauss kernel because the time scales with the number of cells squared. Also the training can be slow, depending on the number of training events and number of cells one wishes to create.

6 Acknowledgements

At first I would like to thank both of my supervisors, Tancredi Carli and Andreas Höcker and Dominik Dannheim for helping me to understand the TFoam algorithm and for lots of discussions about the improvement ideas of the algorithm. I also would like to thank Peter Speckmayer for his patience to answer all my technical questions concerning the implementation of the PDEFoam algorithm into the TMVA package and C++ in general. And last but not least, I would like to thank the CERN administration and the summer student team for organising such a fantastic summer student program.

References

[1] R. Brun and F. Rademakers, "*ROOT – An Object Oriented Data Analysis Framework*", Nucl. Inst. Meth. in Phys. Res. A 389, 81 (1997).

[2] T. Carli and B. Koblitz, Nucl. Instrum. Meth. **A501**, 576 (2003) [hep-ex/0211019].

[3] T. Carli, D. Dannheim, A. Voigt, P. Speckmayer, "*PDE-FOAM – a probability-density estimation method based on self-adapting phase-space binning*", in preparation

[4] A. Höcker, P. Speckmayer, J. Stelzer, et. al. "*TMVA Toolkit for Multivariate Data Analysis with ROOT*", CERN-OPEN-2007-007, physics/0703039

[5] S. Jaddach, "*Foam: A General-Purpose Cellular Monte Carlo Event Generator*", CERN-TH/2002-059, physics/0203033