Boosting PDE-Foam

Summer project at CERN Jul–Aug 2010

Alexander Voigt*
Technische Universität Dresden

November 13, 2010

PDE-Foam is a multivariate probability density estimator, which divides a multi-dimensional phase space into a finite number of hyper-rectangles (cells) of constant event or discriminant density. When used for event classification, a reasonable PDE-Foam ($\sim$ 500 cells) performs moderately compared to a neural network (NN) or boosted decision tree (BDT). As to improve this, we study the general idea of classifier boosting on the PDE-Foam method within the TMVA framework for multivariate analysis.

*Alexander.Voigt@physik.tu-dresden.de

Contents

1 Introduction

PDE-Foam [1] is a multivariate probability density estimator, which was developed as an advancement of PDE-RS [2] in face of reduction of the sensitivity on statistical fluctuations in the training sample as well as memory consumption and speed during the classification phase. It is based on the self-adapting binning algorithm TFoam [3, 4], which divides the multi-dimensional phase space in a finite number of hyper-rectangles (cells).

It was found that for most classification problems a standard PDE-Foam with $\sim$ 500 cells performs moderately compared to a neural network (NN) or boosted decision tree (BDT). A way of improving this is to "boost" the classifier [5]. In this process a set of weak classifiers (PDE-Foams) is created, which, in combination, form a more powerful classifier. The aim of this project was to

1. Study the behavior of PDE-Foam under the boosting algorithm AdaBoost [5].
2. Develop and study alternative event and classifier weightings for the boosting algorithm.
3. Technical comparison of PDE-Foam and decision tree (DT).
4. Show that, with certain modifications, a (boosted) PDE-Foam behaves like a (boosted) decision tree (BDT).
5. Write a framework for parameter scans of TMVA classifiers using a batch system.

For the following studies we use three different event samples:

Example 1: Two-dimensional Gaussian ring distribution (see Figure 1), as was used in [1]. This example is challenging, because both PDE-Foam and BDT use rectangular cells/nodes to approximate the ring distributions.

Example 2: Sine distribution in the first variable and exponential in the second variable (see Figure 2).

Example 3: Five moderately correlated observables constructed from Gaussian distributions, as was used in [1].

For every example we use $5 \cdot 10^5$ events for training and $5 \cdot 10^5$ events for testing to get numerical results with sufficient precision.

As a measure of the average estimator performance of the trained classifier we use the *Receiver operating characteristic* (ROC). A value of 0.5 is obtained for a random classification of signal and background. A value of 1.0 is obtained for a perfect discrimination between signal and background.

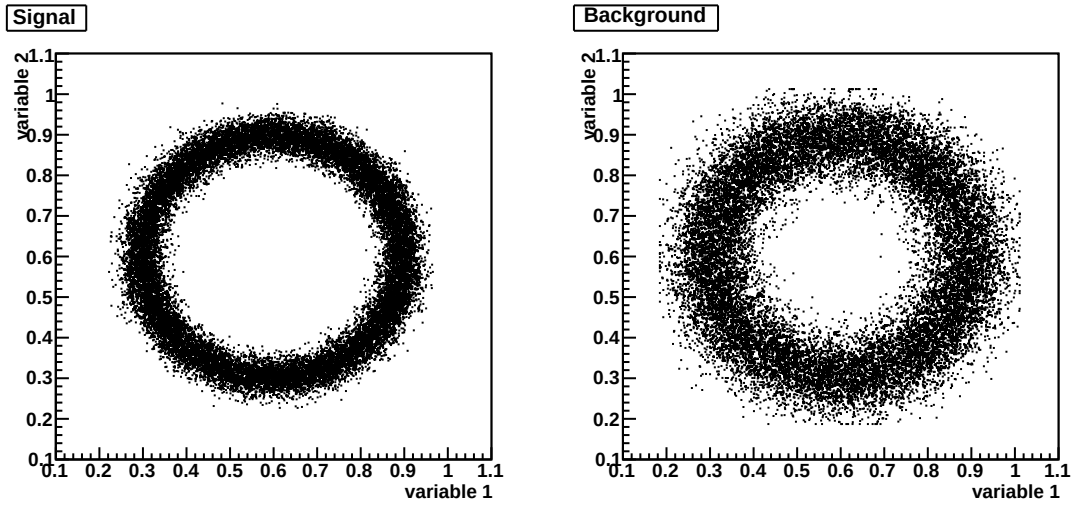


Figure 1: Signal and background distributions of example 1.

2 Boosting PDE-Foam

The idea of boosting is to combine a set of weak classifiers to give a more powerful ensemble [5]. There are various boosting algorithms, e.g. GentleBoost [6], LogitBoost [6] and AdaBoost [5, 6]. The latter is popular in HEP applications and as of version 4.0.7 the only one implemented in TMVA [7].

2.1 AdaBoost

2.1.1 The algorithm

The AdaBoost was first proposed in 1995 by Y. Freund [5]. It is intended for classifiers, which give a discrete output $Y \in \{-1, 1\}$. However, since there exist many likelihood classifiers which give continuous results, extensions were developed [6], which consider real-valued classifier output, e.g. $Y \in [0, 1]$. Since PDE-Foam uses the discriminant

$$D = \frac{N_{\text{sig}}}{N_{\text{sig}} + N_{\text{bkg}}} \tag{1}$$

to classify an event x (N_{sig} is the number of signal, and N_{bkg} is the number of background events in the PDE-Foam cell corresponding to the event x), the real AdaBoost algorithm is most suited for boosting PDE-Foam. It is described in Algorithm 1 on page 6.

2.1.2 PDE-Foam boosting results

Here and in the following we use the standard PDE-Foam option string

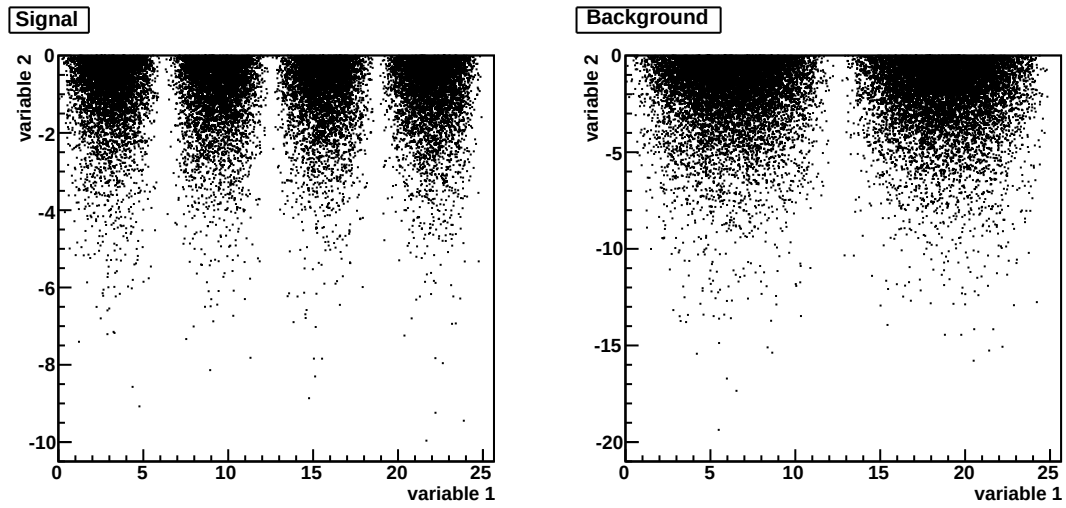


Figure 2: Signal and background distributions of example 2.

```
   SigBgSeparate=F:VolFrac=0.0333:TailCut=0.001:nBin=5:
↪   nSampl=2000:Nmin=100:Kernel=None:
↪   FillFoamWithOrigWeights=F:MaxDepth=0:UseYesNoCell=F:
↪   DTLogic=None:PeekMax=T
```

until stated otherwise. We use the single foam approach (`SigBgSeparate=F`), which holds directly the discriminant, instead of treating signal and background distributions separately. The root cell geometry is adjusted such that 0.1 % of the training data (outlier events) is not included. The probe volume for the density sampling is set to $1/30^d$ (`VolFrac=0.0333`), where d is the number of variables in the training event sample. The number of MC samplings per cell (`nSampl`) is set to 2000 and the number of bins in the cell edge histograms (`nBin`) to 5. For the cell splitting we always choose the cell with the highest separation gain (`PeekMax=T`), which contains more than 100 events (`Nmin=100`). The cell tree depth is not limited (`MaxDepth=0`). After the build-up the foam is filled with the full event weights (`FillFoamWithOrigWeights=F`). During the classification phase it returns the discriminant (`UseYesNoCell=F`), instead of discrete values for signal and background events. See also [7] and Table 3 for a more detailed description of the PDE-Foam options.

The behavior of a standard PDE-Foam under AdaBoost for example 3 with different number of active cells is shown in Figure 3. One finds a sufficient convergence of the boosting algorithm after 20 boosts and a performance gain up to 15 %. Concerning the number of active cells, in general foams with more cells perform better than the ones with fewer. Note also that foams with less than six cells tend to break boosting at a very early stage, because the misclassification rate of the last trained classifier exceeds 0.5, i.e. it performs worse than random guessing. The maximum

Algorithm 1 real AdaBoost, as implemented in TMVA

Assume that a classifier $G(x)$ shall be boosted M times. The training sample $\vec{x} = (x_1, \ldots, x_N)$ consists of N events x_i with known classification result y_i and weight w_i for every x_i $(i = 1, \ldots, N)$.

1. For $m = 1$ to M do
   a) Train a classifier $G_m(x)$ on the training sample.
   b) Compute misclassification rate

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i},$$

   where $I(\text{true}) = 1$ and $I(\text{false}) = 0$. If $\mathrm{err}_m \geq 0.5$ set $M \leftarrow m$, i.e. stop the loop after this iteration.
   c) Compute classifier weight $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   d) Reweight training sample

$$w_i \leftarrow w_i \exp\left[\alpha_m I(y_i \neq G_m(x_i))\right]$$

   for $i = 1, \ldots, N$.
2. Output $G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$

ROC integrals for PDE-Foams with 500 active cells for the different event samples are listed in Table 1. The table also shows the ROC integrals obtained with a standard boosted decision tree (BDT) with the option string

```
   NTrees=400:nEventsMin=400:MaxDepth=3:BoostType=AdaBoost
↪   :SeparationType=GiniIndex:nCuts=20:PruneMethod=
↪   NoPruning
```

In example 1 the standard BDT performs significantly worse than PDE-Foam, which is due to the small number of BDT leaf nodes ($2^3 = 8$). Increasing the BDT tree depth to 5 (equates to 32 leaf nodes) would increase the ROC integral to 0.700, which is approximately the PDE-Foam result. The same argument holds for example 2. Increasing the BDT tree depth to 5 here yields a ROC integral of 0.811, which is equal to the PDE-Foam result. Concerning example 3 one finds that the BDT performs nearly optimal for a small number of leaf nodes. Even with 500 active cells the standard PDE-Foam is not able to reach the BDT result. In Section 3.3 a modification of PDE-Foam is presented, which yields ROC integrals similar to the ones from BDT.

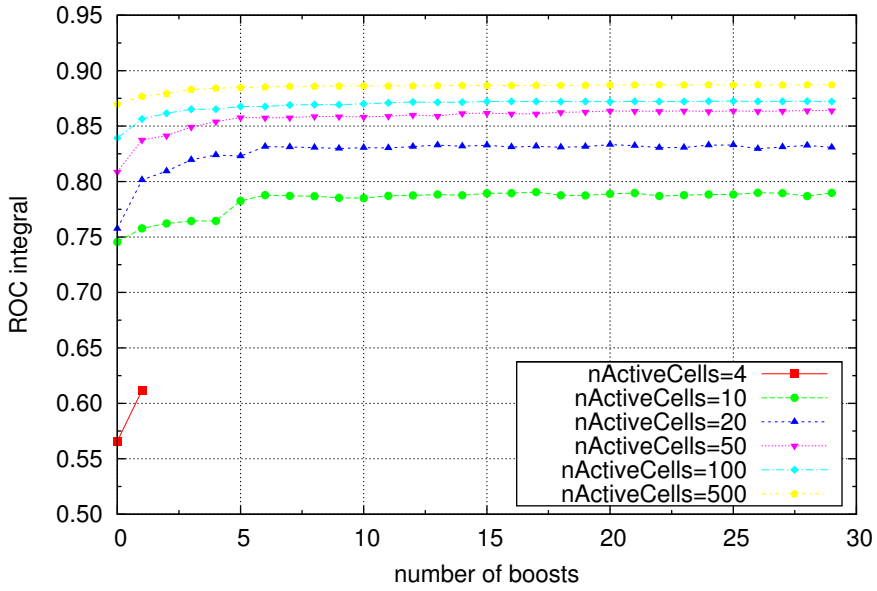


Figure 3: Standard PDE-Foam boosted with real AdaBoost for different number of active cells for example 3. The used option string is listed in Section 2.1.2.

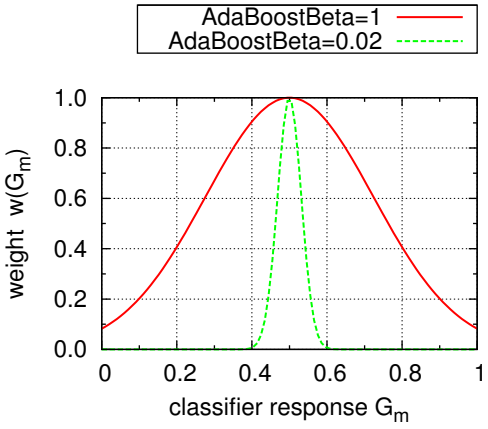
A limiting factor of this study is the CPU time needed for a single boost. For the used event samples and a standard PDE-Foam with 500 active cells it takes $\sim 5 \ldots 10$ min on 2.0 GHz 64-bit processor. However, in practice one could speed up the boosting by not creating all boosting control plots.

2.2 MVA value dependent event reweighing

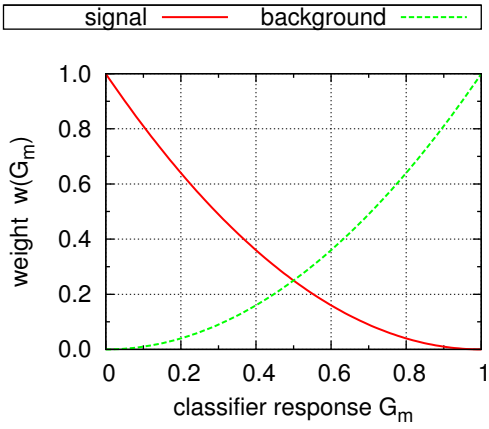
In the AdaBoost algorithm all weights of misclassified events are scaled by the same factor. In other words, there is no distinction between events which are classified "very wrong" and those which are classified "slightly wrong". In the following other

Table 1: Maximum ROC integrals for a standard PDE-Foam with 500 active cells boosted with real AdaBoost for examples 1, 2 and 3. The third column shows the number of boosts needed for PDE-Foam to reach the maximum ROC integral. In the last column the ROC integrals for the standard BDT (see option string above) are listed.

Example	max. ROC (PDE-Foam)	number of boosts	max. ROC (BDT)
1	0.694	11	0.619
2	0.811	22	0.808
3	0.887	20	0.905



(a) `HighEdgeGauss` weight function for $C_m = 0.5$ and different values of `AdaBoostBeta`



(b) `HighEdgeCoPara` weight function for signal and background, `AdaBoostBeta=2`

Figure 4: Event weight functions (a) `HighEdgeGauss` and (b) `HighEdgeCoPara` (see Eq. (2)–(3))

boosting algorithms are studied, which adjust the individual boost weights based on the corresponding discriminant values.

2.2.1 HighEdgeGauss

The `HighEdgeGauss` method strongly weights all events, which yield discriminant values close to the signal/background cut value C_m of the classifier G_m. The events are reweighted by the Gaussian function

$$w_i \leftarrow \exp\left[-\frac{(G_m(x_i) - C_m)^2}{0.1 \cdot \texttt{AdaBoostBeta}}\right], \tag{2}$$

which peeks at C_m. The parameter `AdaBoostBeta` can be set by the user to adjust the "strength" of the boosting (see Figure 4a). The boosting results for a PDE-Foam with different number of active cells and different values of `AdaBoostBeta` for example 3 are shown in Figure 5. It was found that the value `AdaBoostBeta=0.02` gives the best classification results for this event sample. For a PDE-Foam with 500 active cells the `HighEdgeGauss` event reweighting yields a maximum ROC integral of 0.885 in case of `AdaBoostBeta=1`, and 0.900 in case of the optimal value `AdaBoostBeta=0.02`. The maximum ROC integrals for the examples 1, 2 and 3 are also listed in Table 2.

2.2.2 HighEdgeCoPara

The `HighEdgeCoPara` boost type reweights all those events strongly, which have discriminant values close to 1 in case of background, and close to 0 in case of signal events. The weighting function reads

$$w_i \leftarrow \begin{cases} (1.0 - G_m(x_i))^{\texttt{AdaBoostBeta}} & \text{if } x_i \text{ is a signal event,} \\ G_m(x_i)^{\texttt{AdaBoostBeta}} & \text{if } x_i \text{ is a background event.} \end{cases} \tag{3}$$

Again the parameter `AdaBoostBeta` can be set by the user to influence the strength of the boosting (see Figure 4b). The boosting results of a standard PDE-Foam with `HighEdgeCoPara` for example 3 is shown in Figure 6. It was found that the value `AdaBoostBeta=1` gives the best classification results for this event sample. The maximum ROC integral of 0.897 is reached for a PDE-Foam with 500 active cells. For a lower number of active cells this boosting algorithm performs worse than the optimized `HighEdgeGauss`, except for a foam with 4 active cells. Though this effect can be sample dependent. The maximum ROC integrals for all event samples are shown in Table 2.

2.3 Alternative classifier weighting

The AdaBoost algorithm weights a single classifier G_m according to the misclassification rate err_m of G_m on the reweighted training sample. Classifiers with values $\text{err}_m \approx 0$ get a high weight and the ones with $\text{err}_m \lessapprox 0.5$ get a weight close to zero. We studied the following alternative classifier weightings.

2.3.1 ROC integral

In case one is interested in maximising the ROC integral of the classifier $G(x)$ it makes sense to weight a single classifier G_m proportional to the ROC integral on the original training sample, i.e.

$$\alpha_m = \text{ROC}(\text{orig. train.}). \tag{4}$$

This option has been made available in TMVA by the name `ByROC`.

2.3.2 Inverse overlap integral

Since one is interested in separating the signal and background MVA distributions $\text{MVA}_{\text{sig}}(D)$, $\text{MVA}_{\text{bkg}}(D)$, it makes sense to choose the classifier weight as the inverse

overlap integral of these two distributions, i.e.

$$\alpha_m = \left[\int_0^1 \mathrm{d}D \; \min\Big[\mathrm{MVA}_{\mathrm{sig}}(D), \mathrm{MVA}_{\mathrm{bkg}}(D)\Big] \times \Theta\Big(\mathrm{MVA}_{\mathrm{sig}}(D)\,\mathrm{MVA}_{\mathrm{bkg}}(D)\Big) \right]^{-1}, \tag{5}$$

where

$$\Theta(x) = \begin{cases} 1 & \text{for } x \geq 0, \\ 0 & \text{else.} \end{cases} \tag{6}$$

This option is called `ByOverlap`.

The ROC integrals of a standard PDE-Foam boosted `ByROC` and `ByOverlap` for example 3 are shown in Figures 7–8 for different event weightings and different number of active cells. The performance curves of `ByROC` and `ByOverlap` nearly coincide for all event weighting types. Furthermore, they are very similar in shape to the curves in Figure 3, 5 and 6, where the original AdaBoost classifier weighting (`ByError`) was used. Table 2 shows the maximal ROC integrals for a PDE-Foam with 500 active cells for different event and classifier weightings. One finds that the differences in the classifier weighting methods for a fixed event weighting are smaller than 0.3 %.

Table 2: Maximum ROC integrals for a standard PDE-Foam with 500 active cells for different event and classifier weightings. The combination AdaBoost–ByError corresponds to Algorithm 1. For `HighEdgeGauss` and `HighEdgeCoPara` the optimal values of `AdaBoostBeta` found in Sec. 2.2 were used.

		AdaBoost	HighEdgeGauss	HighEdgeCoPara
Example 1	ByError	0.694	0.701	0.697
	ByOverlap	0.693	0.701	0.700
Example 2	ByError	0.811	0.813	0.810
	ByOverlap	0.811	0.812	0.812
Example 3	ByError	0.887	0.900	0.897
	ByROC	0.888	0.900	0.898
	ByOverlap	0.890	0.900	0.898

3 Comparison PDE-Foam – decision tree

3.1 Difference between PDE-Foam and decision tree

The PDE-Foam structure is formally equivalent to a decision tree (DT). However, the foam build-up and the classification differs from the DT in three points.

Cell split algorithm

The first difference is in the way PDE-Foam and DT split a single cell/node.

PDE-Foam does a range searching in the cell (`nSampl` times) with a box with volume `VolFrac`. For the set events found within the box the discriminant Eq. (1) is calculated and projected onto the cell edges. After this the foam uses a two-pointer algorithm [3, 4] to determine the best cell division point, which maximal reduces the variance of the discriminant distribution (`SigBgSeparate=F`).

The decision tree projects all events in a node onto the node edges and uses a one-pointer algorithm to determine the best node division point, which maximal reduces a specified separation quantity (`GiniIndex`, `MisClassificationError`, `CrossEntropy`) [7].

It is obvious that these two approaches are not identical. First of all the gain measure is different. While PDE-Foam uses the variance of the discriminant distribution, the DT uses the purity of a node concerning signal or background. Second the edge distribution of the foam is smeared because of the MC sampling. Finally, the one-pointer algorithm might not find the same division point as the two-pointer one.

Choosing the next cell to split

Per default PDE-Foam chooses the cell with the maximum separation gain for the next split. Thus an arbitrarily shaped cell tree can be generated. In contrast, the DT splits all cells until the maximum tree depth (set by the user) is reached. Thus a completely balanced tree is generated.

Classification value

Per default the DT returns 1 for an event which is classified as signal, and -1 for background. On the contrary, PDE-Foam returns the discriminant $D \in [0, 1]$ for a given event. If D exceeds a certain cut value, the event is classified as signal, otherwise background.

3.2 New PDE-Foam options to make PDE-Foam behave like a decision tree

In order to study the difference between PDE-Foam and a DT in more detail additional PDE-Foam options were build into PDE-Foam, which, in a certain combination, can simulate the behavior of a decision tree. They are listed in Table 3.

The `UseYesNoCell` option can be set to `true` in order to return -1 for events with discriminant $D < 0.5$ (background-like) and 1 for events with $D \geq 0.5$ (signal-like), instead of D.

Since the PDE-Foam cells are filled with events after the splitting, it is in principle possible to use different event weights for the filling than for the foam build-up. The option `FillFoamWithOrigWeights` was created to choose either the original or the full event weight (including the boost weight) to be filled into the foam cells after the build-up. This option is only relevant for boosting, because for non-boosted classifiers the boost weights are always 1. When setting `FillFoamWithOrigWeights=T`, one would only boost the foam geometry, instead of the cell content. This would slow down the boosting process, because the boost weights are ignored in the classification. In most cases studied `FillFoamWithOrigWeights=T` leads to worse classification results than `FillFoamWithOrigWeights=F`. However, when using stronger boosting by choosing `AdaBoostBeta` accordingly, filling the original weights into the foam can improve the performance.

In analogy to the decision tree, the option `MaxDepth` was added. When given an integer value greater than zero, the cell tree will not be deeper than `MaxDepth`. By convention the root node has depth 1, which implies that a foam with 2 active cells (3 cells in total) has depth 2. If `nActiveCells` $\geq 2^{\texttt{MaxDepth}-1}$ the resulting cell tree will be completely balanced, as in the case of a decision tree. When `MaxDepth` is set to 0, the cell tree depth is not limited.

In order to emulate a decision tree-like cell splitting algorithm, the option `DTLogic` was introduced. When set to `GiniIndex`, `MisClassificationError` or `CrossEntropy`, the algorithm projects all events in a cell onto the cell edges and probes `nBin-1` division points such that the separation gain

$$\text{gain(parent cell)} - \text{gain(daughter cell 1)} - \text{gain(daughter cell 2)} \tag{7}$$

is maximal. For a given separation type and a given cell the gain is defined as

$$\texttt{GiniIndex :} \quad \text{gain(cell)} = p(1-p), \tag{8}$$

$$\texttt{MisClassificationError :} \quad \text{gain(cell)} = 1 - \max(p, 1-p), \tag{9}$$

$$\texttt{CrossEntropy :} \quad \text{gain(cell)} = -p\log p - (1-p)\log(1-p), \tag{10}$$

where $p = N_{\text{sig}}/(N_{\text{sig}}+N_{\text{bkg}})$ in the considered cell. It was found that `nBin=20` yields sufficient classification results. When `DTLogic` is set to `None`, the original PDE-

Foam cell splitting algorithm is used, which reduces the variance of the sampled discriminator density.

In order to emulate the decision tree behavior even more, it is now possible to choose the last created cell for the next split, instead of the one with maximum separation gain. The corresponding option is `PeekMax=F`. Note that adequate classification results are only achieved, if simultaneously the maximum tree depth is less than infinity, i.e. `MaxDepth` > 0.

Certain DT options were found to have an equivalent in PDE-Foam. These are listed in Table 4. In order to completely emulate a decision tree one would have to adjust the PDE-Foam options to their DT analogon. An adequate PDE-Foam option string is for example

```
   SigBgSeparate=F:TailCut=0:nActiveCells=100000:nBin=20:
↪   Nmin=400:Kernel=None:FillFoamWithOrigWeights=F:
↪   MaxDepth=4:UseYesNoCell=T:DTLogic=GiniIndex:PeekMax=F
```

3.3 Boosting a decision tree-like PDE-Foam

Figure 9–11 and Table 5 show a direct comparison of a BDT and a boosted decision tree-like PDE-Foam for example 3. For the BDT we use the option string

```
   nEventsMin=400:NNodesMax=100000:BoostType=AdaBoost:
↪   nCuts=19:PruneMethod=NoPruning:UseYesNoLeaf=T
```

and for PDE-Foam the equivalent

```
   SigBgSeparate=F:TailCut=0:nActiveCells=100000:nBin=20:
↪   Nmin=400:Kernel=None:FillFoamWithOrigWeights=F:
↪   UseYesNoCell=T:PeekMax=F:Boost_Type=AdaBoost
```

For all implemented separation types one finds an agreement between the BDT and PDE-Foam ROC integrals within 0.6 % precision after 100 boost steps. This remaining deviation mainly results from the different handling of events, which are close to the cell/node edges. Due to the strong weighting of wrong classified events, these edge events might affect the cell splitting point, which can lead to different ROC integrals. This effect then propagates through the boosting iterations. The highest ROC integral value of about 0.906 for both classifiers is obtained for the `MisClassificationError` separation type and a tree depth of 3 (BDT) and 4 (PDE-Foam). This corresponds to a BDT/PDE-Foam with only 8 leaf nodes/active cells. One also finds that greater and smaller tree depth yield worse classification results for this event sample, independent of the separation type.

4 Conclusions

It was shown that boosting a standard PDE-Foam can lead to an improvement of the average classification results (ROC integral) up to 15 %, depending on the number of active foam cells. Alternative event weighting methods (`HighEdgeGauss`, `HighEdgeCoPara`) were studied and, after optimization, found to improve the classification even further. The studied alternative classifier weightings `ByROC` and `ByOverlap` did not lead to significant improvements compared to the standard AdaBoost (`ByError`) for the here studied event samples.

The analogy between BDT and PDE-Foam was examined in detail and additional PDE-Foam options were created to emulate the behavior of a decision tree. The boosting results of a BDT and a analog PDE-Foam were found to coincide up to 0.6 %. The so obtained maximum ROC integrals exceed the ones obtained with the standard PDE-Foam algorithm up to 2 %.

5 Acknowledgments

I am heartily thankful to my supervisors Dominik Dannheim and Tancredi Carli as well as Lorenzo Moneta and Pere Mato Vila for making this project possible. For the many helpful suggestions and countless hours of discussion and I am grateful to Dominik Dannheim, Tancredi Carli, Helge Voss, Jan Therhaag and the TMVA developers team.

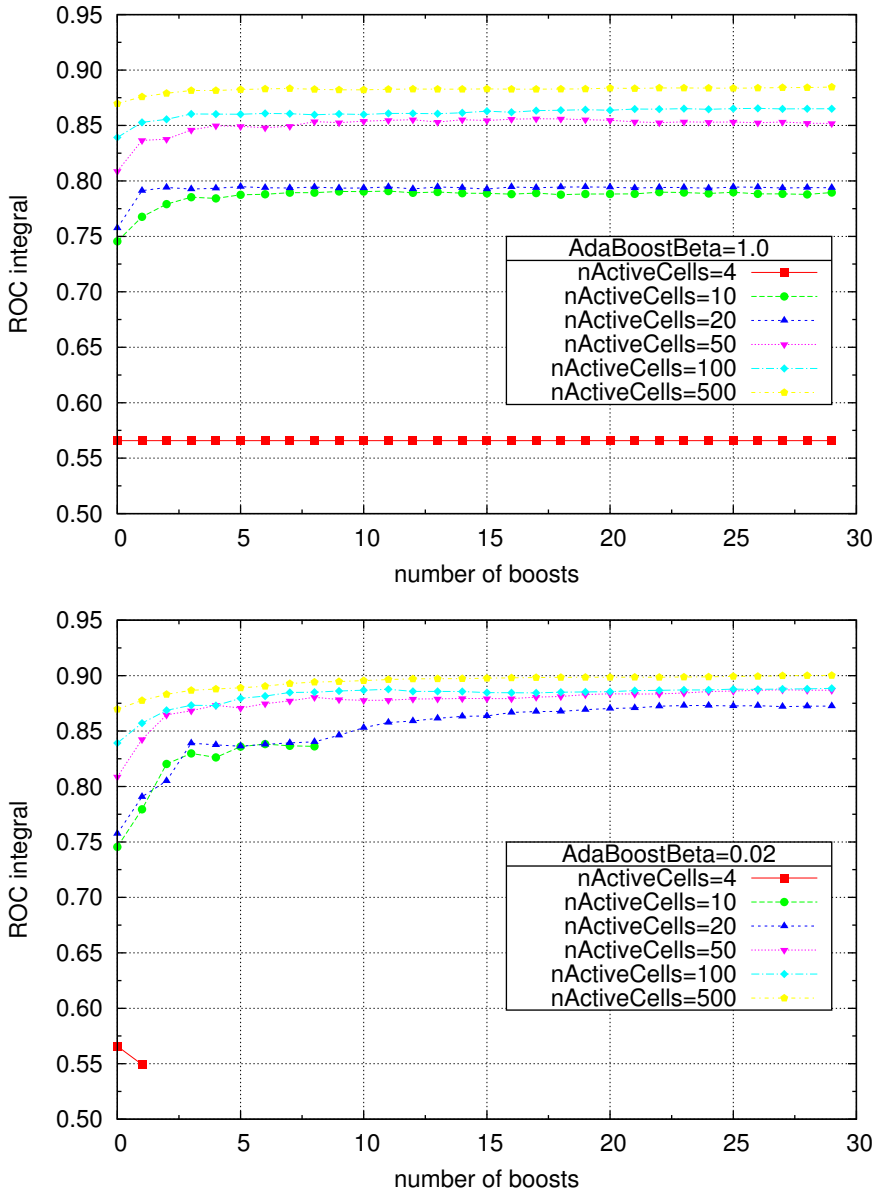


Figure 5: Standard PDE-Foam boosted with `HighEdgeGauss` for different number of active cells for example 3. The upper plot uses `AdaBoostBeta=1` and the one below `AdaBoostBeta=0.02`. The latter was found to be an optimal value for this event sample.

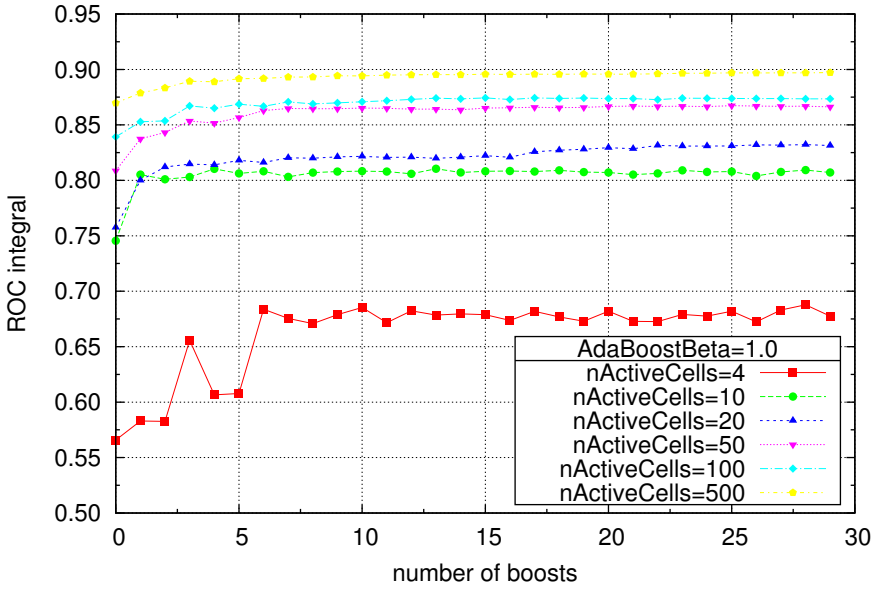


Figure 6: Standard PDE-Foam boosted with `HighEdgeCoPara` for different number of active cells for example 3. We set `AdaBoostBeta=1`, which was found to be an optimal value for this event sample.

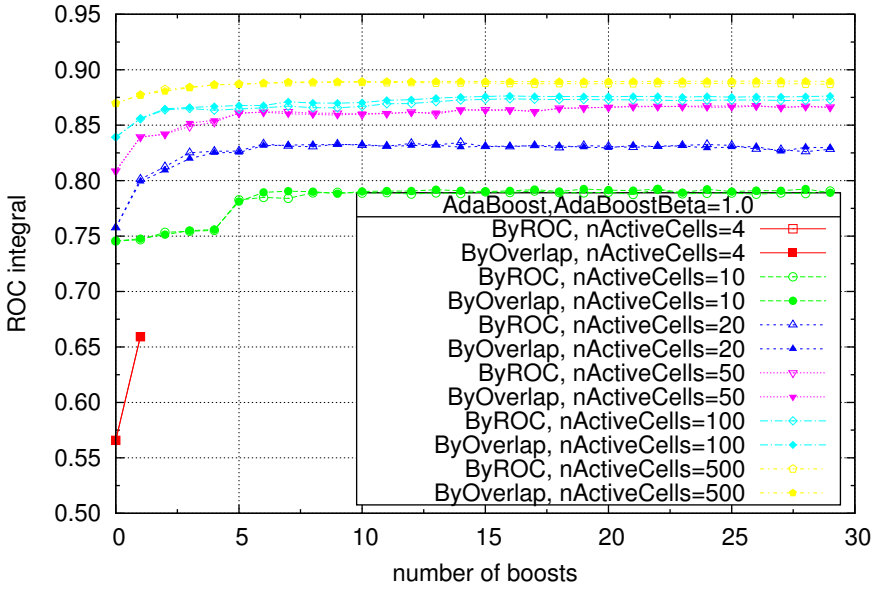


Figure 7: Standard PDE-Foam boosted with `AdaBoost` for different number of active cells and different method weight types for example 3. We set `AdaBoostBeta=1`, as required for the real AdaBoost algorithm.

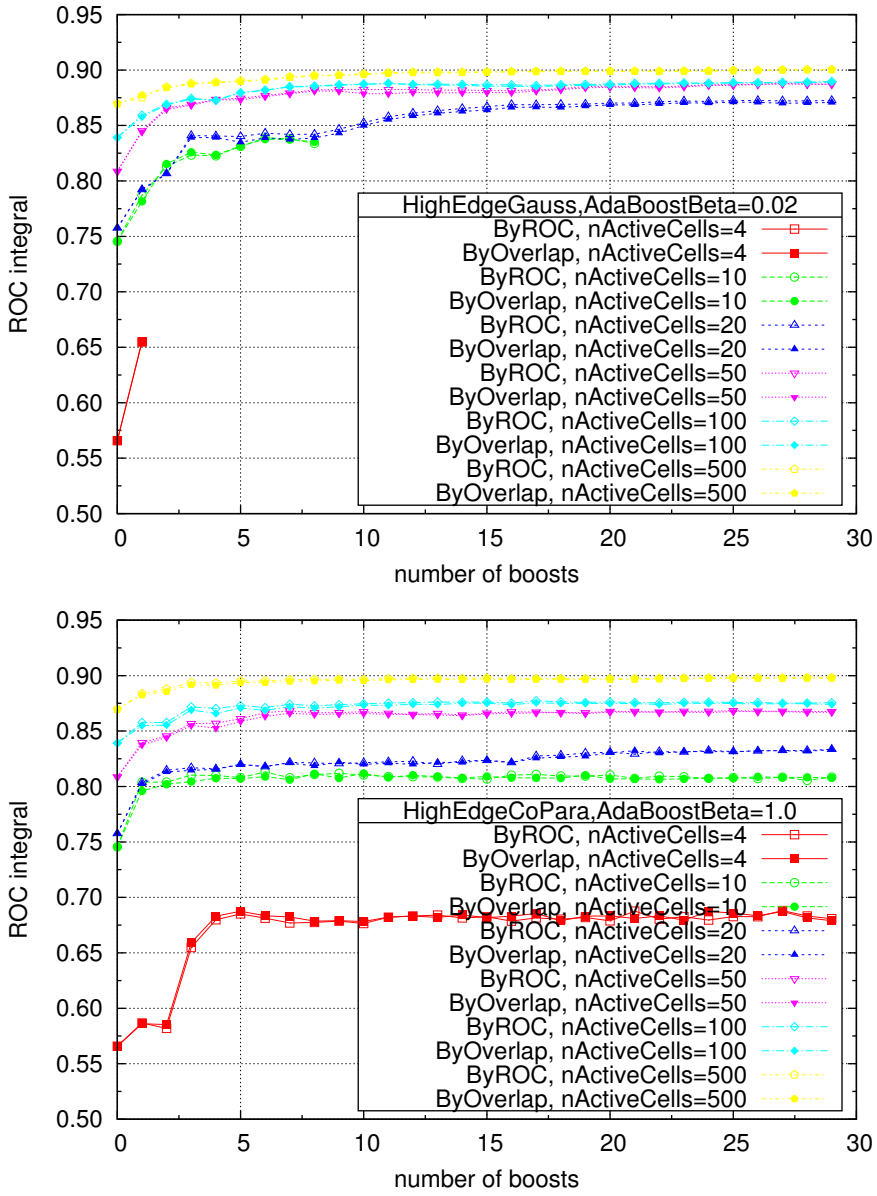


Figure 8: Standard PDE-Foam boosted with `HighEdgeGauss` (upper plot) and `HighEdgeCoPara` (lower plot) for different number of active cells and different method weight types for example 3. We set `AdaBoostBeta` to the optimal values for each boost type.

Table 3: New PDE-Foam options

Name	Values	Default	Description
`UseYesNoCell`	`true, false`	`false`	When `true` return $D \in \{1, -1\}$ for signal and background respectively, when `false` return the discriminant $D = N_{\text{sig}}/(N_{\text{sig}}+N_{\text{bkg}})$ as classification output.
`FillFoamWithOrigWeights`	`true, false`	`false`	Fill original (`true`) or full event weights (`false`) into PDE-Foam cells.
`MaxDepth`	$\geq$ `0`	`0`	Maximum depth of cell tree (root cell has depth 1). If set to 0, the depth is unlimited.
`DTLogic`	`None, GiniIndex, MisClassificationError, CrossEntropy`	`None`	Specifies decision tree algorithm used to split a cell. If set to `None`, the original PDE-Foam algorithm is used.
`PeekMax`	`true, false`	`true`	Peek cell with maximum separation gain (`true`), or last cell created (`false`) for the next split.

Table 4: Analogous PDE-Foam and Boosted decision tree (BDT) options

PDE-Foam	BDT	Description
`UseYesNoCell`	`UseYesNoLeaf`	Return $D \in \{1, -1\}$ for signal and background respectively, or the discriminant $D = N_{\text{sig}}/(N_{\text{sig}} + N_{\text{bkg}})$ as classification output.
`MaxDepth`	`MaxDepth+1`	Maximum depth of node/cell tree
`DTLogic`	`SeparationType`	Separation criterion for node/cell splitting
`nBin`	`nCuts+1`	Number of steps during node/cell cut optimization
`Nmin`	`nEventsMin`	Minimum number of events required in a leaf node/active cell
`2*nActiveCells-1`	`NNodesMax`	Total number of nodes/cells
`Boost_Num`	`NTrees`	Number of boosts
`Boost_Type`	`BoostType`	Boosting type

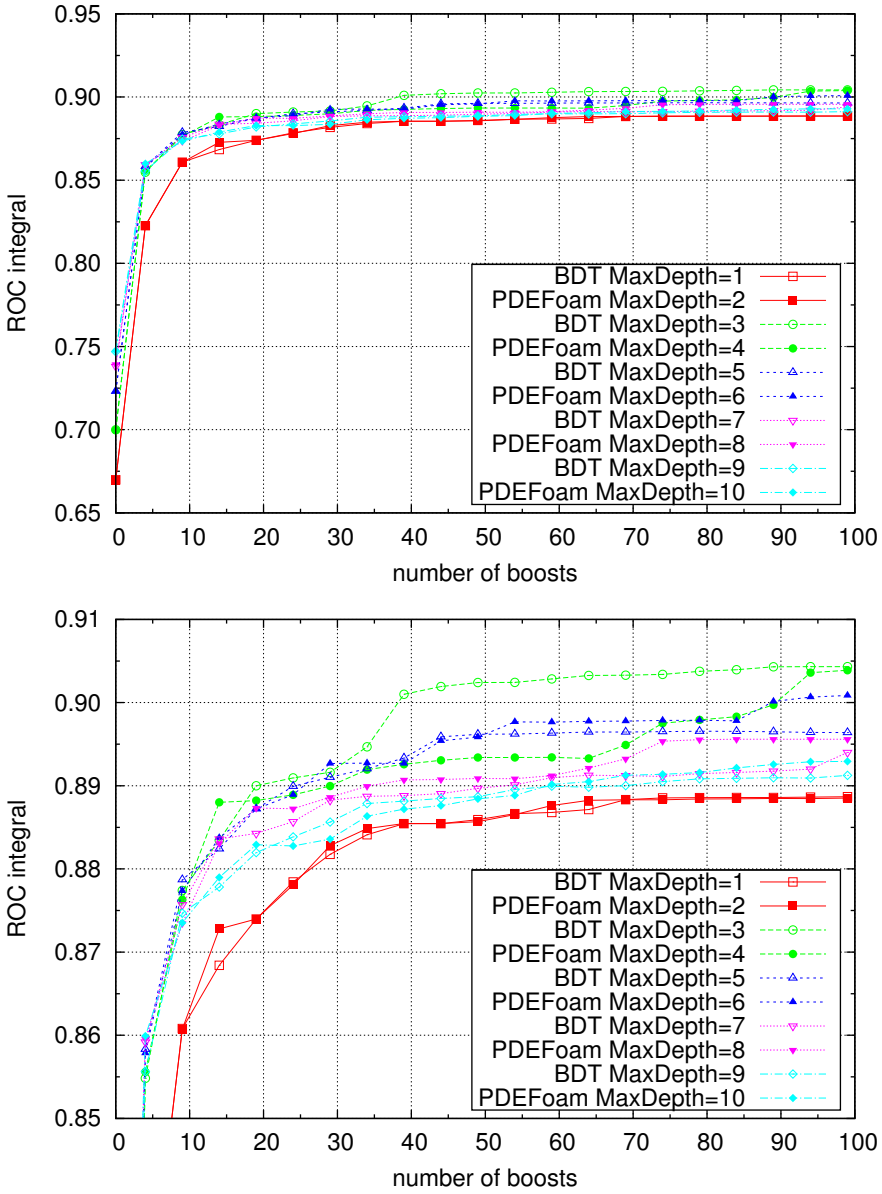


Figure 9: BDT and boosted DT-like PDE-Foam with `GiniIndex` as separation type for different tree depth (example 3). Note that the BDT `MaxDepth` option corresponds to `MaxDepth+1` for PDE-Foam.

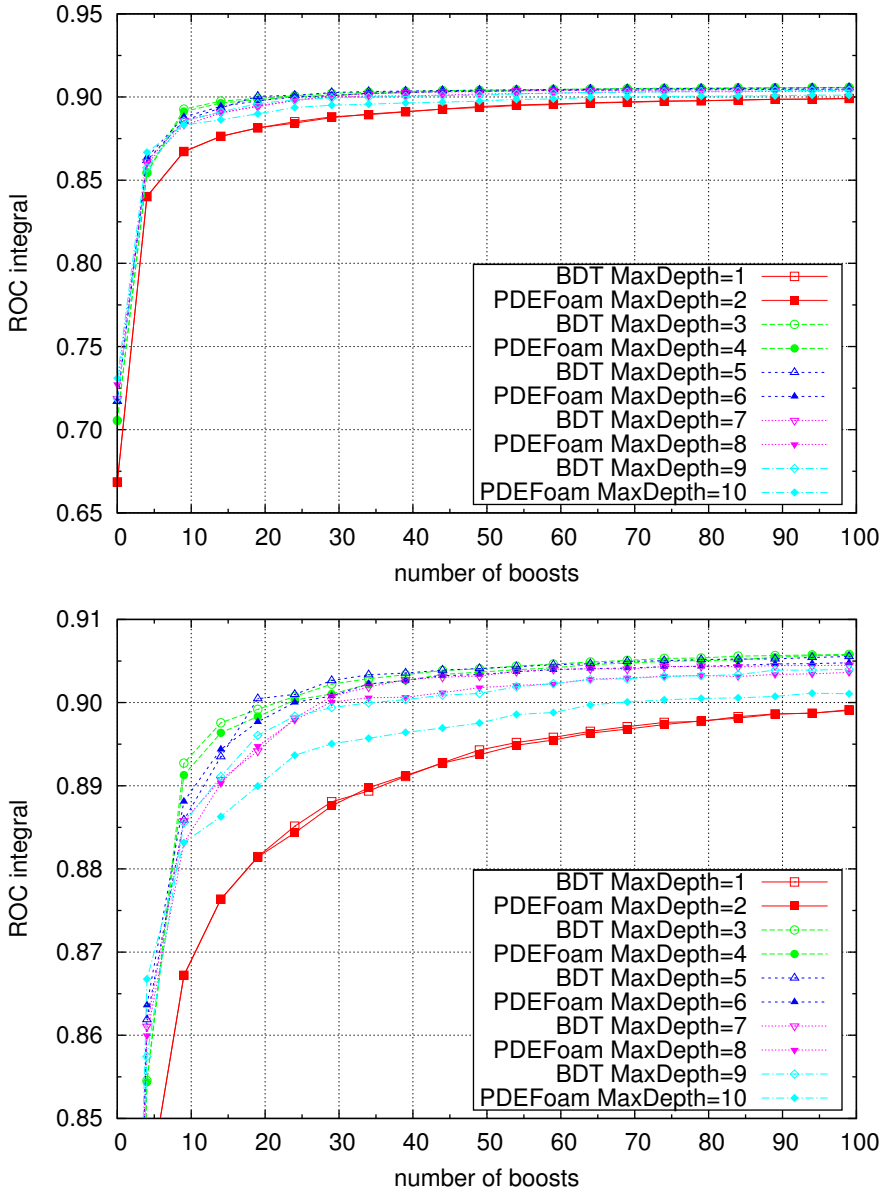


Figure 10: BDT and boosted DT-like PDE-Foam with `MisClassificationError` as separation type for different tree depth (example 3). Note that the BDT `MaxDepth` option corresponds to `MaxDepth+1` for PDE-Foam.

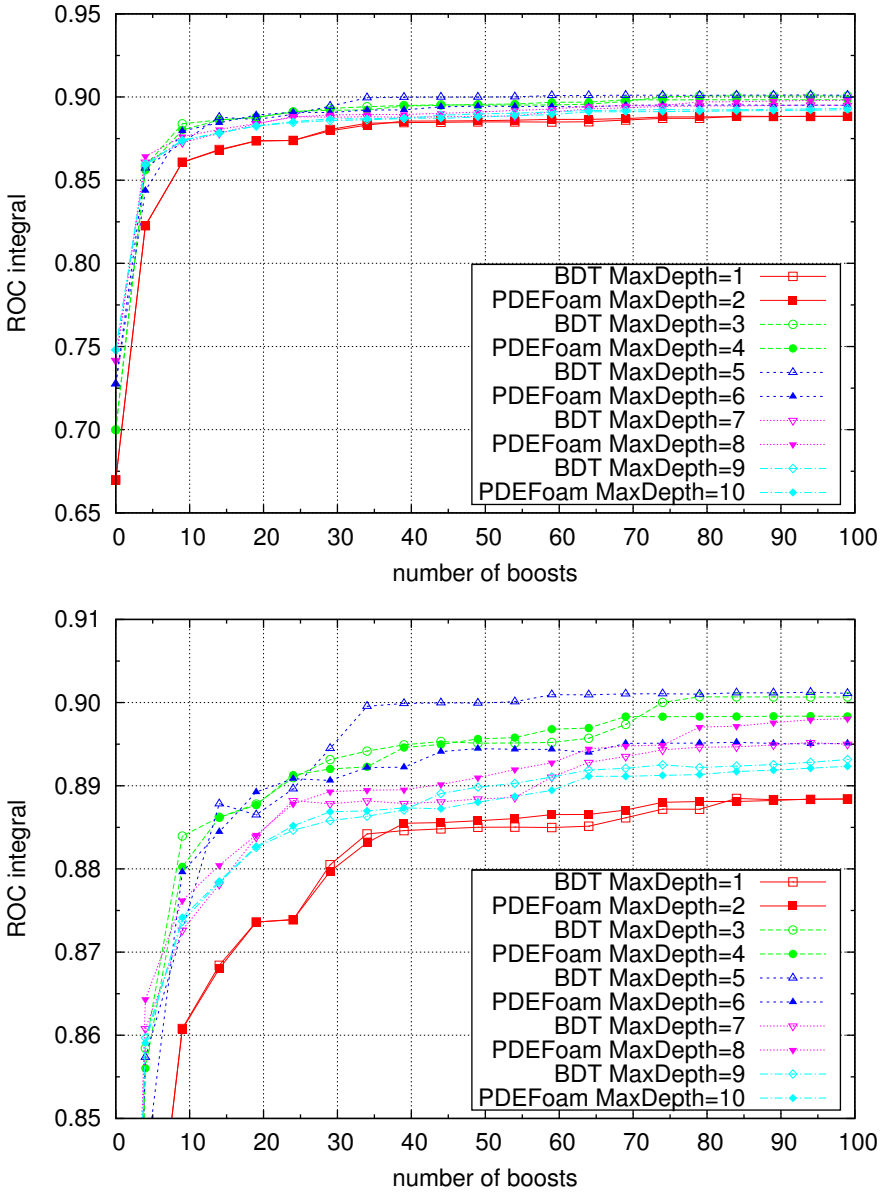


Figure 11: BDT and boosted DT-like PDE-Foam with `CrossEntropy` as separation type for different tree depth (example 3). Note that the BDT `MaxDepth` option corresponds to `MaxDepth+1` for PDE-Foam.

Table 5: Comparison of ROC integrals of BDT and the analog decision tree-like PDE-Foam after 100 boosts for example 3 for different separation types and tree depth

`SeparationType`	`MaxDepth` (BDT)	ROC (BDT)	ROC (PDE-Foam)
`GiniIndex`	1	0.888682	0.888517
	3	0.904329	0.903888
	7	0.894016	0.900856
	5	0.896398	0.895599
	9	0.891258	0.892929
`MisClassificationError`	1	0.899144	0.899039
	3	0.905796	0.905674
	5	0.905544	0.904788
	7	0.904496	0.903631
	9	0.904007	0.901067
`CrossEntropy`	1	0.888426	0.888399
	3	0.900675	0.898336
	5	0.901119	0.895094
	7	0.894982	0.898072
	9	0.893152	0.892344

A Framework for parameter studies in TMVA

In order to make parameter scans of TMVA classifiers using a batch system, a job submission framework was written. It consists of four files:

scan.py The job submission script. Here the user specifies the classifier, the option string, the event sample, the signal and background tree, the variables and the TMVA factory options.

analysis.cxx The job. This C++ program books a single TMVA classifier and writes the output to a given location. It is submitted from `scan.py` to the batch system with the corresponding command line options. Before usage it must be compiled using the `makefile`.

makefile The makefile. The user must edit this file to set the location of the TMVA directory.

vary.py Helper functions. This python module contains helper functions used by `scan.py` to parse the classifier option string.

The syntax of the classifier option string, which is set in `scan.py`, is extended to contain ranges and lists of parameter values for the scan.

Parameter range: Ranges are specified via

$$\texttt{x=[}\langle start\rangle\texttt{,}\langle stop\rangle\texttt{,}\langle step\rangle\texttt{]} \tag{11}$$

where `x` is the parameter name, $\langle start\rangle$ is the start value, $\langle stop\rangle$ is the end value and $\langle step\rangle$ is the step size. Note, that the end value is included in the range.

Parameter list: Lists of possible parameter values are set via

$$\texttt{x={}}\langle value1\rangle\texttt{,}\langle value2\rangle\texttt{,}\langle value3\rangle\texttt{, ...}\} \tag{12}$$

Both the parameter ranges and lists can be combined. For example the option string

$$\texttt{x=[1,3,1]\{10,20\}} \tag{13}$$

would expand to the set of option strings

$$\texttt{x=1}, \texttt{x=2}, \texttt{x=3}, \texttt{x=10}, \texttt{x=20} \tag{14}$$

When the option string contains two or more variables, all possible combinations of their values are generated. For example the option string

$$\texttt{x=[1,3,1]\{10,20\}:y=\{T,F\}} \tag{15}$$

would expand to the set of option strings

$$\begin{array}{l}\texttt{x=1:y=T}, \texttt{x=2:y=T}, \texttt{x=3:y=T}, \texttt{x=10:y=T}, \texttt{x=20:y=T}, \\ \texttt{x=1:y=F}, \texttt{x=2:y=F}, \texttt{x=3:y=F}, \texttt{x=10:y=F}, \texttt{x=20:y=F}\end{array} \tag{16}$$

References

[1] D. Dannheim, A. Voigt, K.-J. Grahn, P. Speckmayer, and T. Carli, PDE-Foam: A probability density estimation method using self-adapting phase-space binning, *Nucl. Instrum. Meth.* **A606**, 717 (2009).

[2] T. Carli and B. Koblitz, A multi-variate discrimination technique based on range-searching, *Nucl. Instrum. Meth.* **A501**, 576 (2003), hep-ex/0211019.

[3] S. Jadach, Foam: Multi-Dimensional General Purpose Monte Carlo Generator With Self-Adapting Symplectic Grid, *Comput. Phys. Commun.* **130**, 244 (2000), physics/9910004.

[4] S. Jadach, Foam: A general purpose cellular Monte Carlo event generator, *Comput. Phys. Commun.* **152**, 55 (2003), physics/0203033.

[5] Y. Freund, Boosting a weak learning algorithm by majority, *Inf. Comput.* **121**, 256 (1995).

[6] J. Friedman, T. Hastie, and R. Tibshirani, Additive Logistic Regression: a Statistical View of Boosting, *Annals of Statistics* **28**, 2000 (1998).

[7] A. Höcker *et al.*, TMVA: Toolkit for multivariate data analysis, *PoS* **ACAT**, 040 (2007), physics/0703039.