boosted decision trees

part 1: decision trees

Paul Seyfert

Heidelberg University

3rd March 2014

Terminology

decision tree a bunch of if-statements which in the end returns -1 or +1
(obtained by machine learning in our case)

node the nodes of a tree (return- and if-statements)

leaf end nodes of a tree (the return statement)

decision forest a collection of decision trees

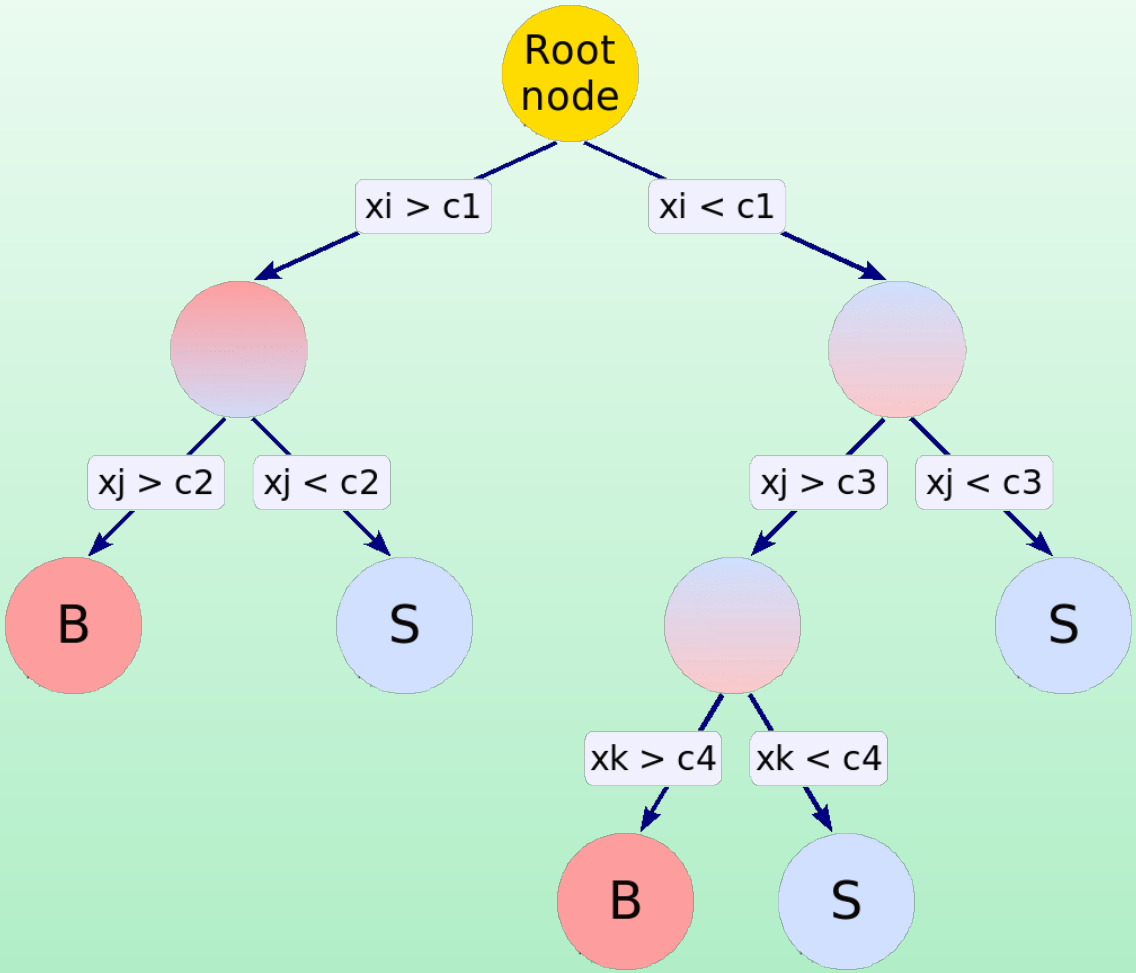
boosted decision tree a decision forest
(where the machine learning was “boosted”)

outline

1. decision trees
2. decision forests
   mainly adaptive boosted decision trees
3. BDTs in practice
   applicable to other MVAs as well

a decision tree example

```
if (x_i > c_1) {
    if (x_j > c_2) {
        return -1;
    } else {
        return +1;
    }
} else {
    if (x_j > c_3) {
        if (x_k > c_4) {
            return -1;
        } else {
            return +1;
        }
    } else {
        return +1;
    }
}
```



how to get a decision tree

several ways to get a decision tree

1. write your selection as decision tree (scnr)
2. use machine learning on MC events
   1. split the sample in two by means of a cut
   2. develope a new cut on each of the two resulting samples
   3. repeat until you end up with 2^n samples with 1 event each (always possible except when there is a signal event identical to a background event)
3. possibly others

properties of decision trees

pro

- relatively understandable
- 100% correct on training sample

con

- decisions are based on statistical fluctuations in training sample
  aka overtraining

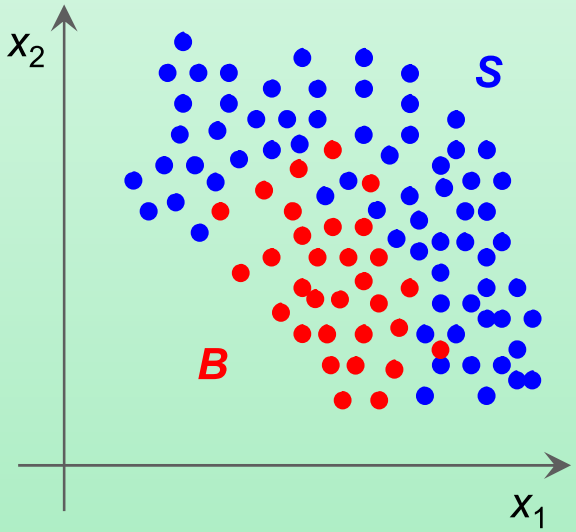
warning

Wikipedia uses the term `overfitting`, TMVA and most people in LHCb use `overtraining`.

interlude: overtraining

definition

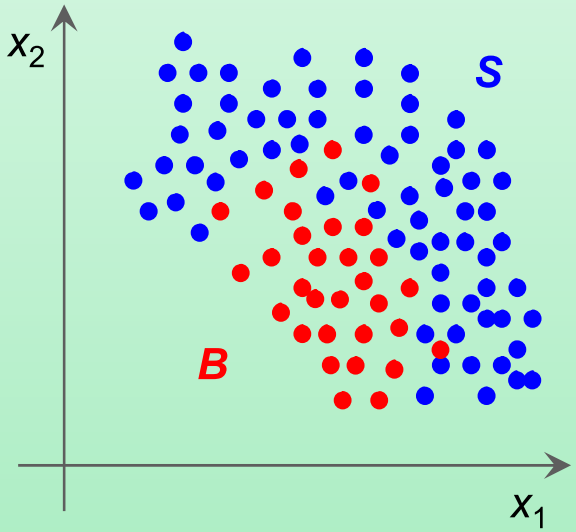
- the machine learning didn't pick up actual signal/background properties, but statistical fluctuations

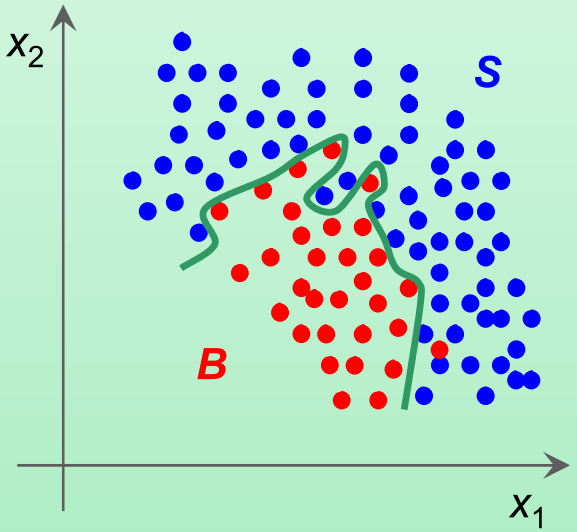


interlude: overtraining

definition

- the machine learning didn't pick up actual signal/background properties, but statistical fluctuations

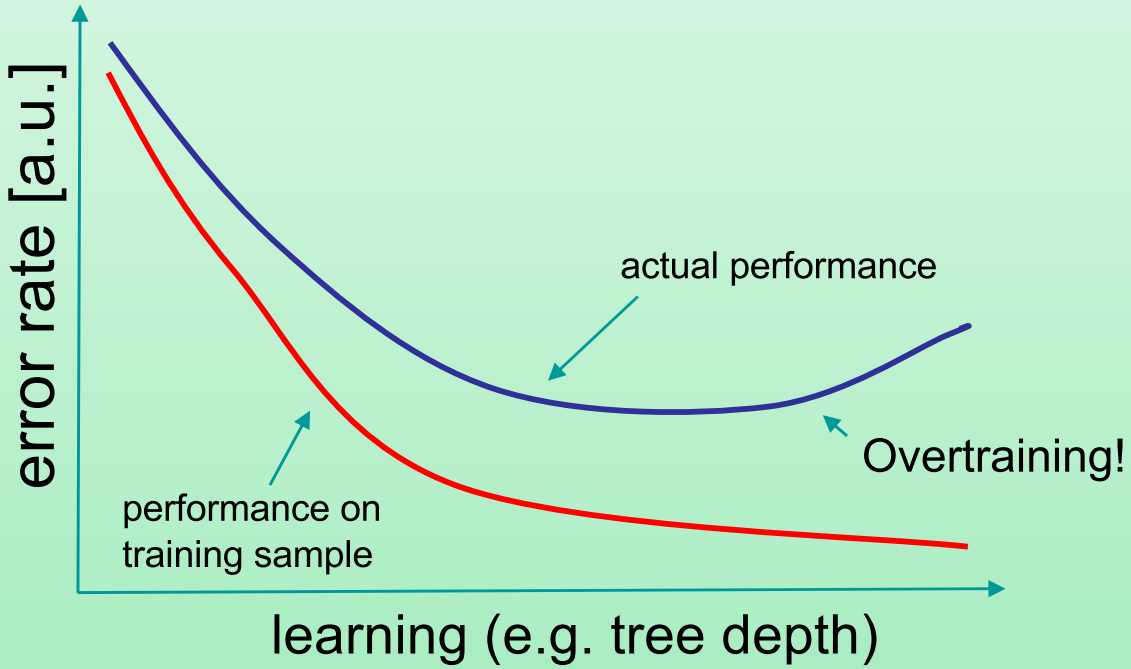




interlude: overtraining

definition

- the machine learning didn't pick up actual signal/background properties, but statistical fluctuations
- performance of a classifier becomes better on the training sample (the algorithm learns more) while the actual performance becomes worse and worse in reality



this is generic

not a property of DT.
happens to BDT, NN, . . . as well

interlude: overtraining

is it bad?

- yes, no, maybe, it depends!
- ✗ in general an overtrained classifier performes worse than it could
- ✗ performance estimates from the training sample overestimate classifier performance
  $\rightarrow$ you shouldn't use the training sample for this anyhow!

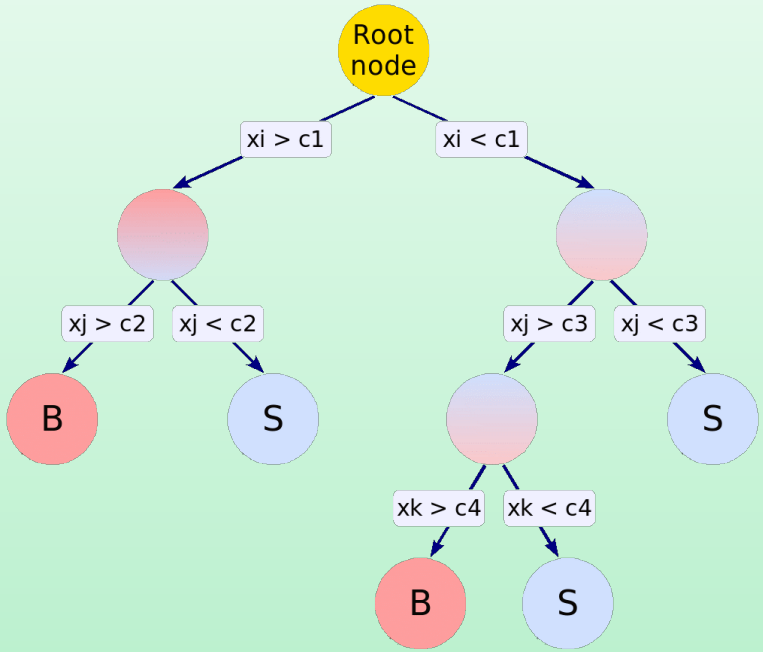
fighting overtrained decision trees

warning

Wikipedia knows `pre-pruning` and `post-pruning`.
Wikipedia-`post-pruning` = TMVA-`pruning`.

fighting overtrained decision trees

- cut away nodes without significance (pruning)
  - ✓ optimal for decision trees
  - ✗ bad for decision forests
- don't create nodes w/o significance in the first place
  - ✗ chess board counter example!
- limit height of the tree
  - ✗ fundamentally limits performance of decision trees
  - ✓ found optimal for decision forests
- minimum number of events for node creation



how to do the splits

- for a fully grown decision tree, random cuts work fine
  BUT: pruning doesn't work with random cuts
- find the cut-variable and cut-value with the best separation
  ✓ sounds good
  ✗ how is "best separation" defined? (give me a FoM)
- luckily most FoMs agree (next slide)

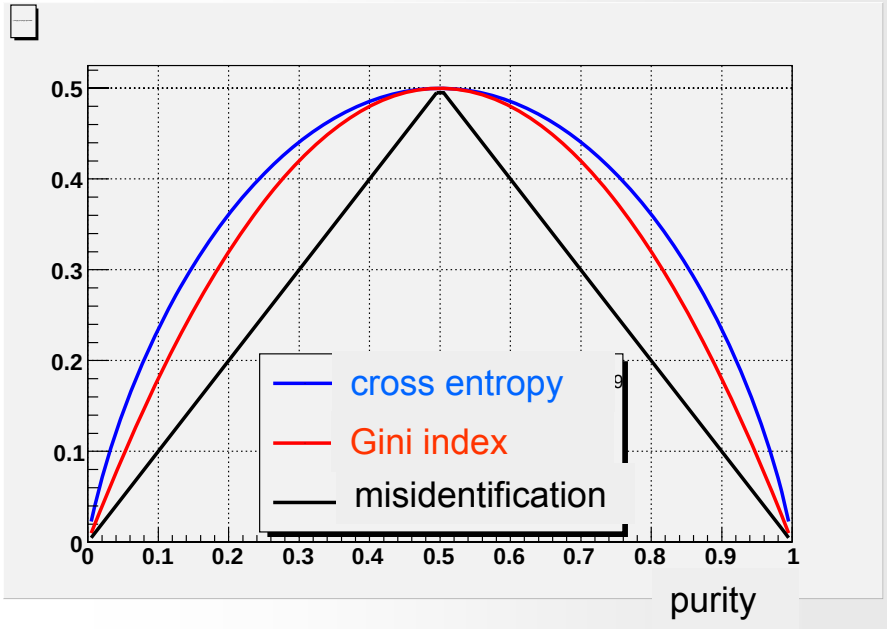
Separation Gain

- What do we mean by “best separation gain”?
- define a measure on how mixed S and B in a node are:
  - Gini-index: (Corrado Gini 1912, typically used to measure income inequality)

    p (1-p) : p=purity
  - Cross Entropy:

    -(plnp + (1-p)ln(1-p))
  - Misidentification:

    1-max(p,1-p)



- difference in the various indices are small, most commonly used: Gini-index

separation gain: e.g. $N_{Parent}*Gini_{Parent} - N_{left}*Gini_{LeftNode} - N_{right}*Gini_{RightNode}$

- Choose amongst all possible variables and cut values the one that maximised the this.

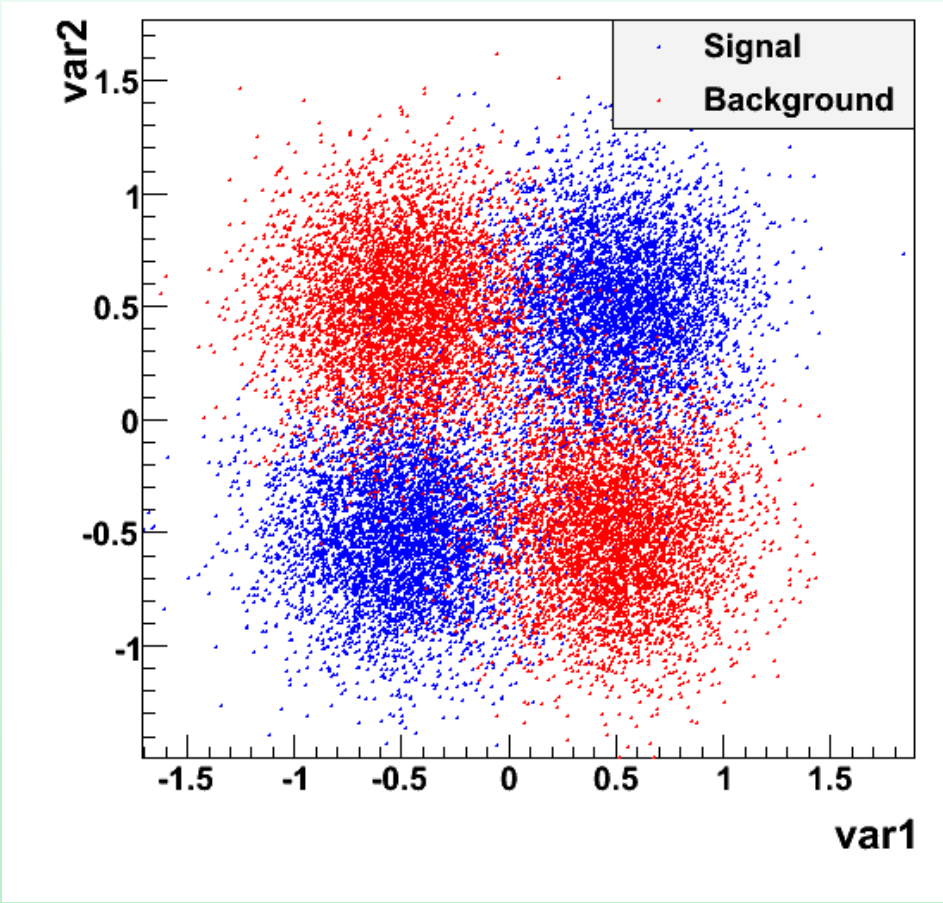
notes on FoMs

- smooth FoMs are better for weighted events
  ! important for boosting
- should be fast to compute
  i.e. entropy = bad

parameter

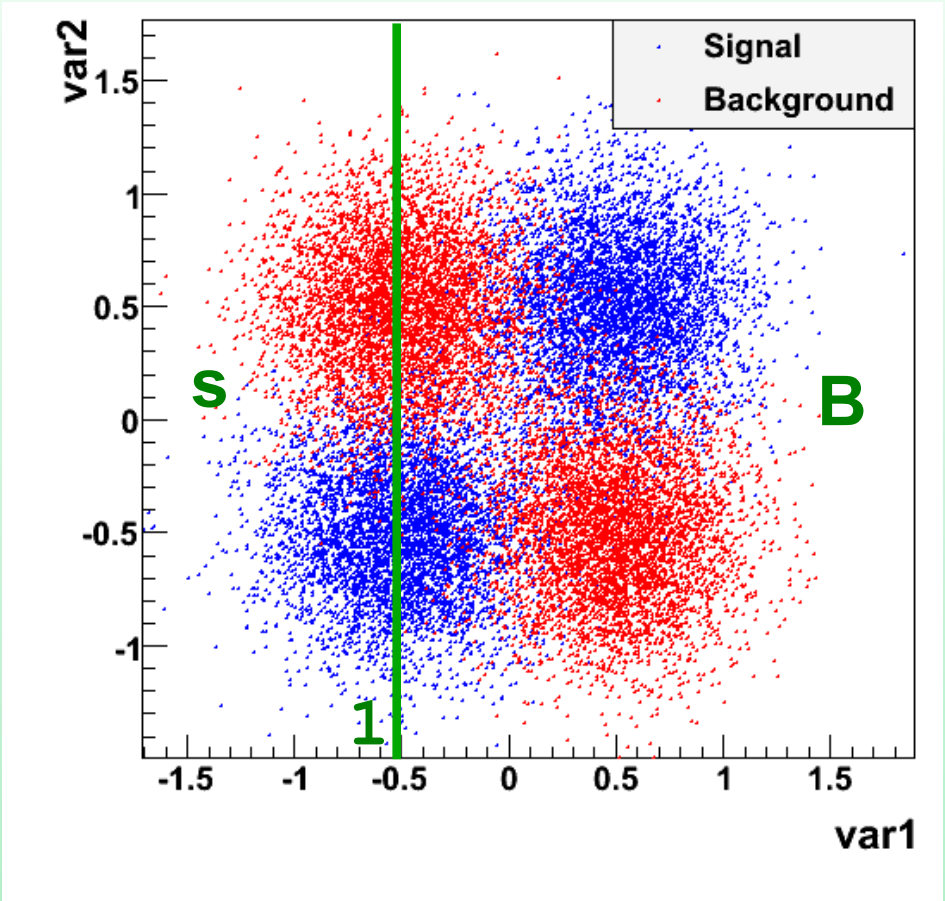
`nCuts` controls how many cut values are tested when creating a node

the chess board example



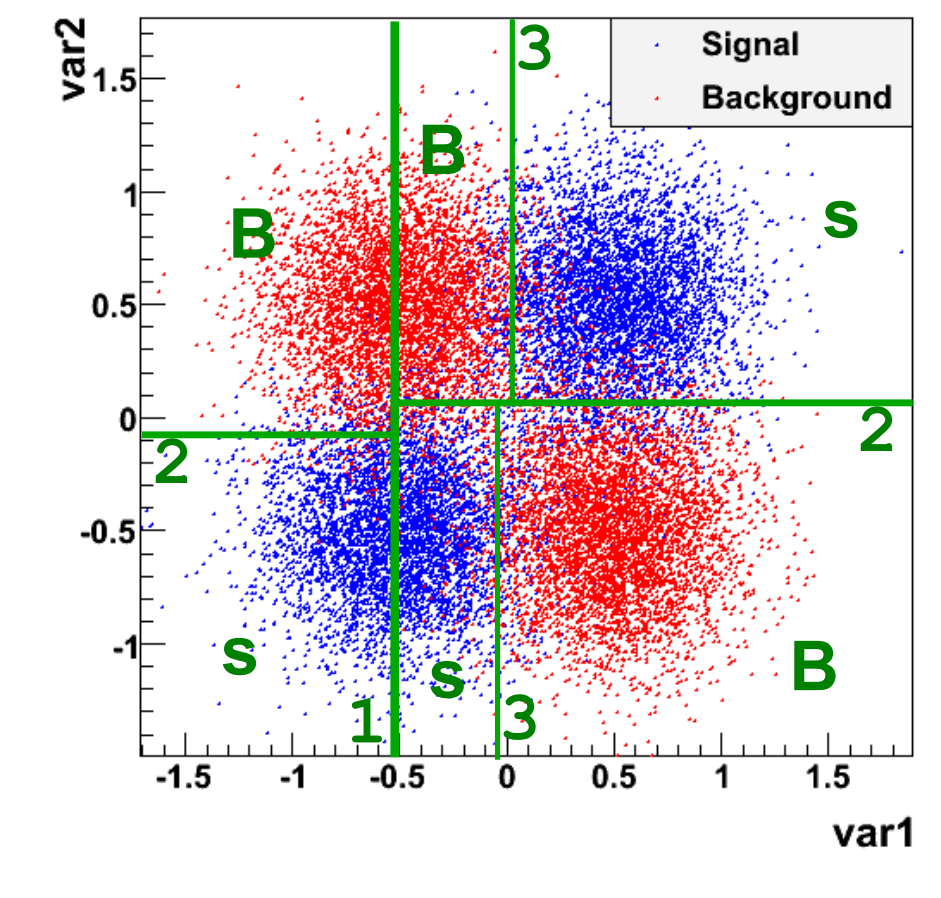
the chess board example

- the first cut won't be significant (pick up fluctuations!)



the chess board example

- the first cut won't be significant (pick up fluctuations!)
- the second cut will be good



$\rightarrow$ on to part 2

lustiges bild hier

boosted decision trees

part 2: decision forests / boosted decision trees

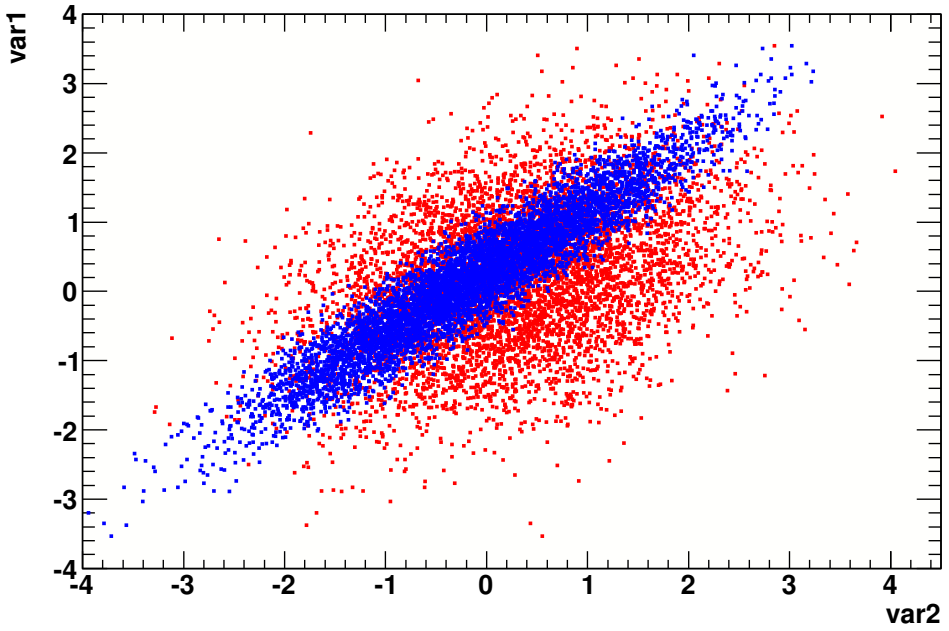
Paul Seyfert

Heidelberg University

3rd March 2014

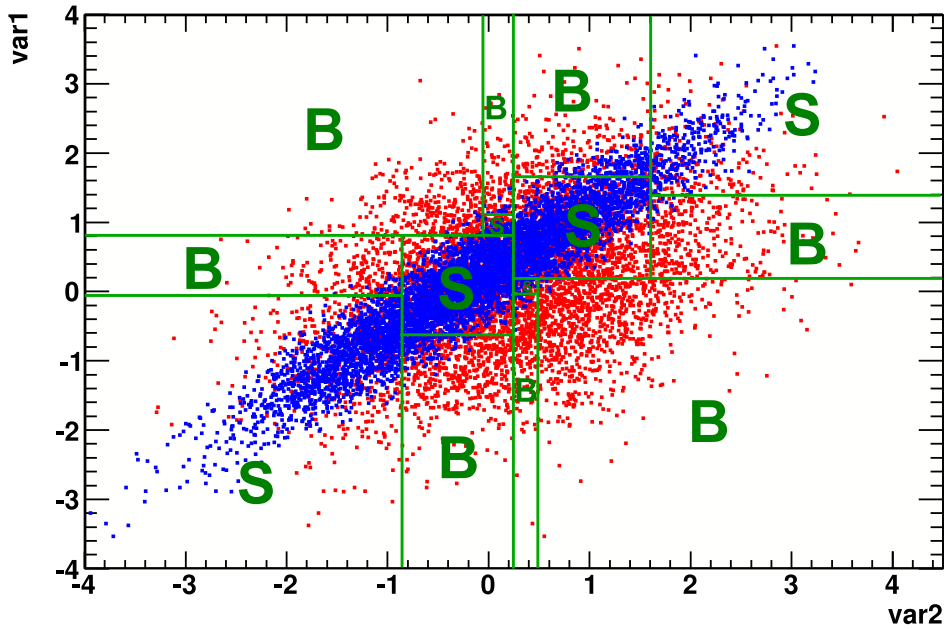
decision trees and correlations

- decision trees are really bad at linear correlations



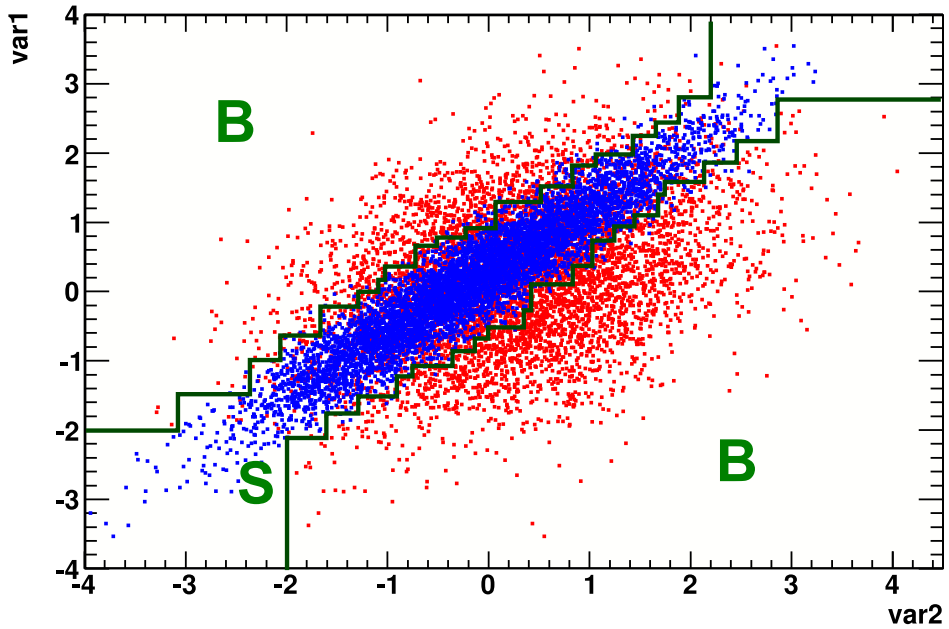
decision trees and correlations

- decision trees are really bad at linear correlations
- e.g. tree depth=4 and still edges wrong



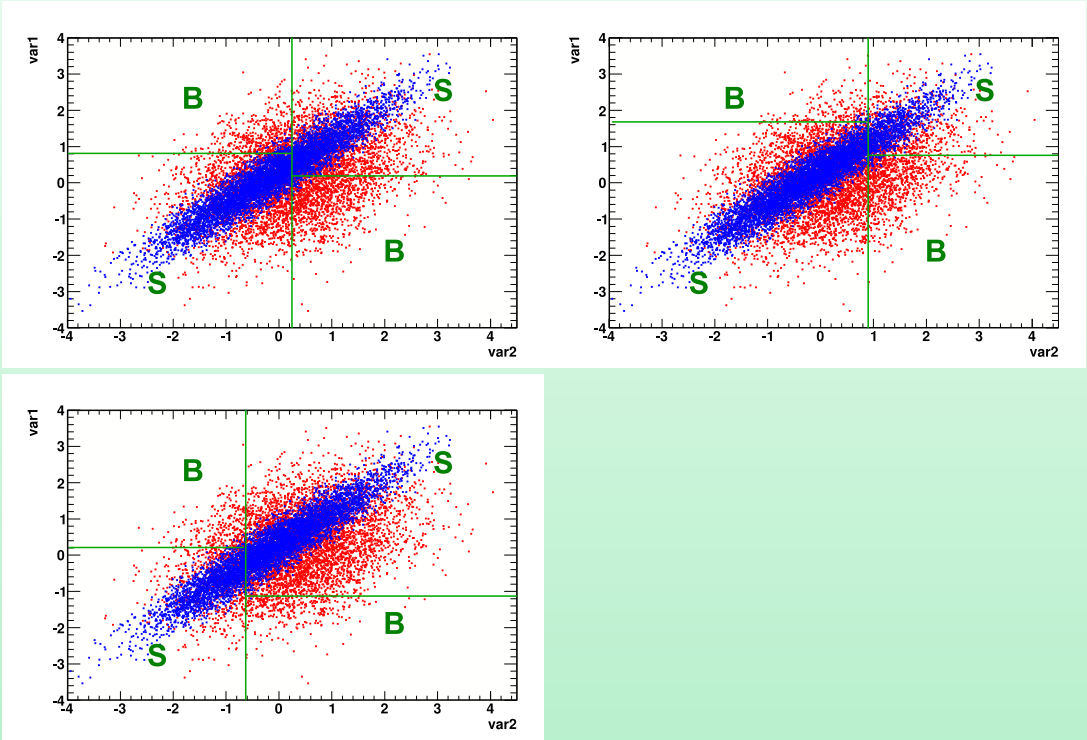
decision trees and correlations

- decision trees are really bad at linear correlations
- e.g. tree depth=4 and still edges wrong
- large tree needed to model steps



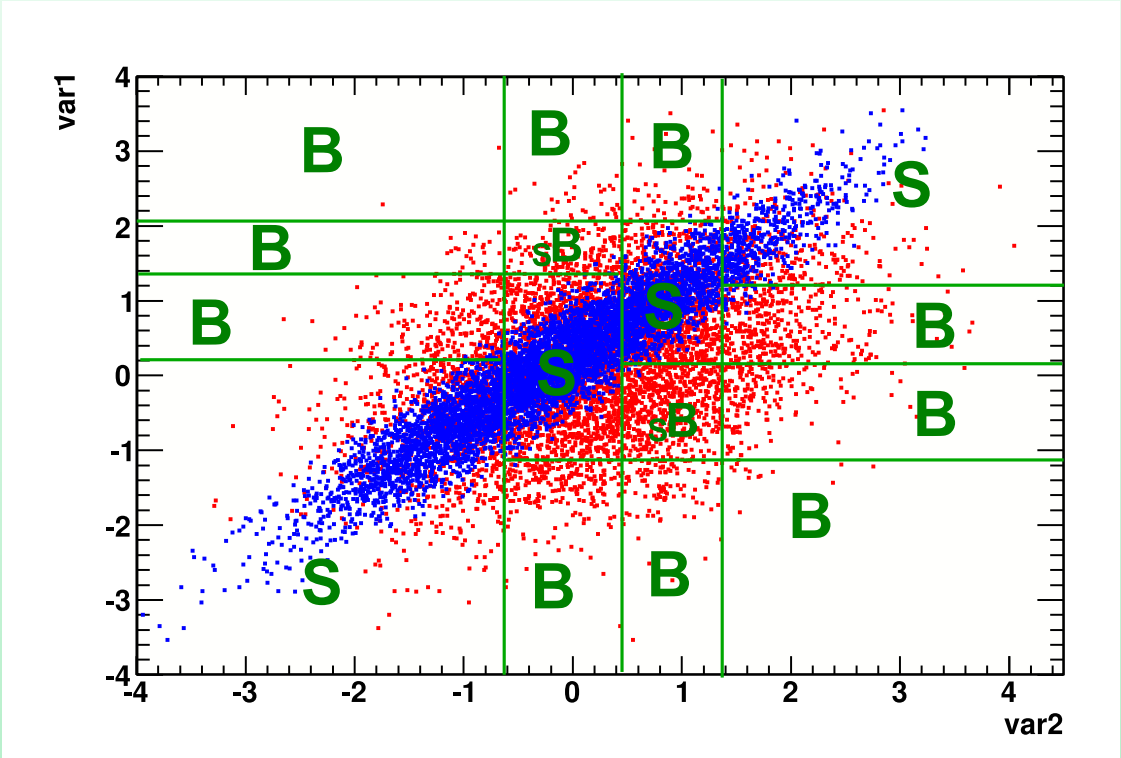
decision forests

- alternatively: average different decision trees



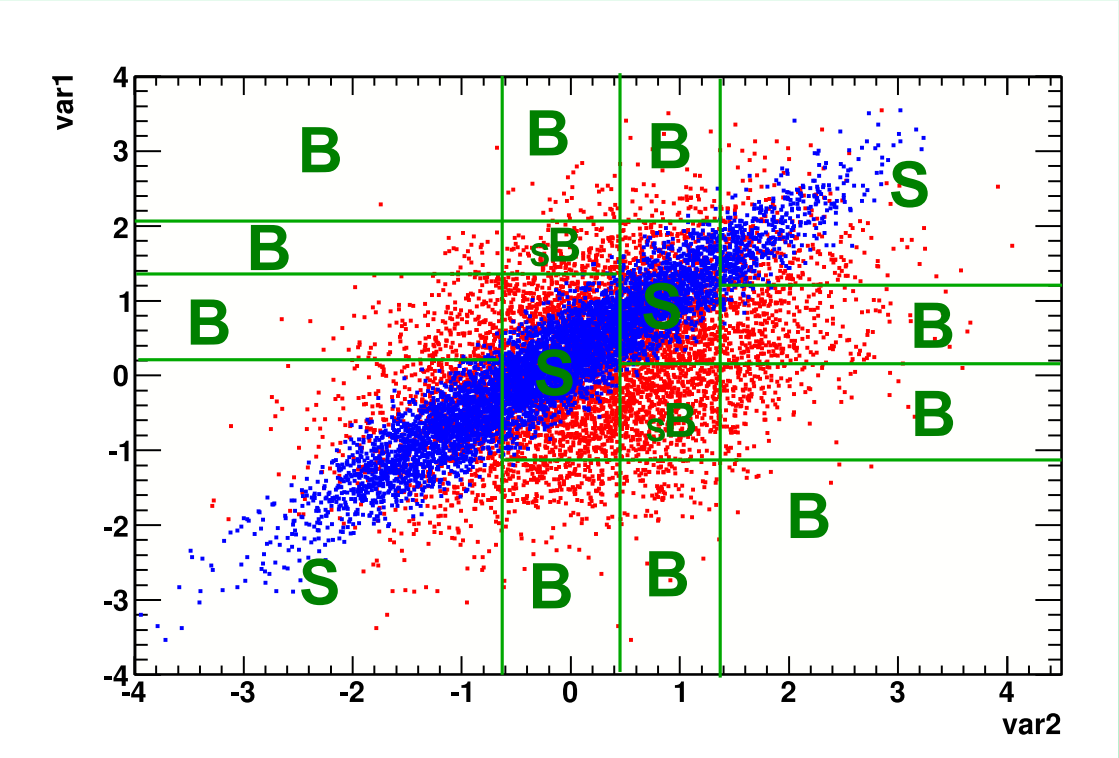
decision forests

- alternatively: average different decision trees
- combination not binary here:
  - definitively signal
  - probably background
  - definitively background
- three trees w/ depth=2 same resolution as one tree w/ depth=6



decision forests

- alternatively: average different decision trees
- combination not binary here:
  - definitively signal
  - probably background
  - definitively background
- three trees w/ depth=2 same resolution as one tree w/ depth=6
- BUT: less vulnerable to overtraining



how to get different trees?

Random trees

- split training sample into random subsamples
- train a decision tree on each of them
- average all trees

$\rightarrow$ individual trees will be different due to fluctuations

$\rightarrow$ averaging of trees averages the training fluctuations out

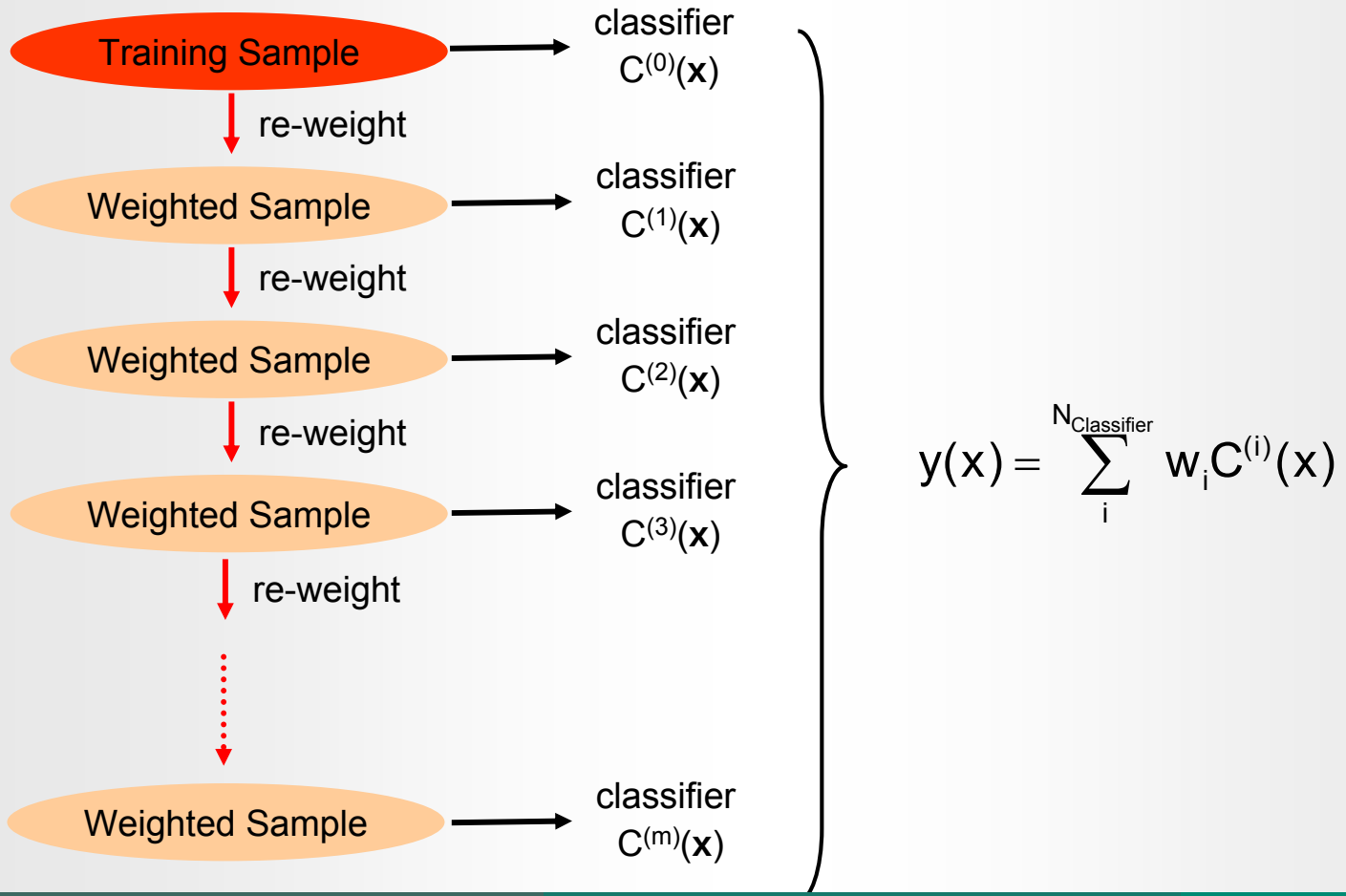
$\Rightarrow$ combined performance driven by actual properties of the data (e.g. physics)

can you do better than just randomising?

Yes you can! (Boosting)

Attempt to make a power series (most significant tree first)

Boosting



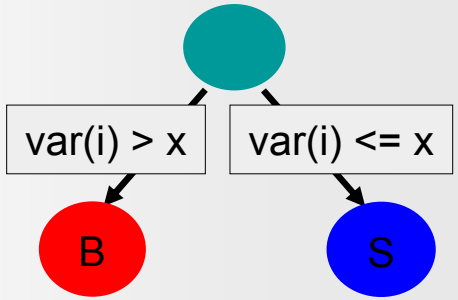
Boosting

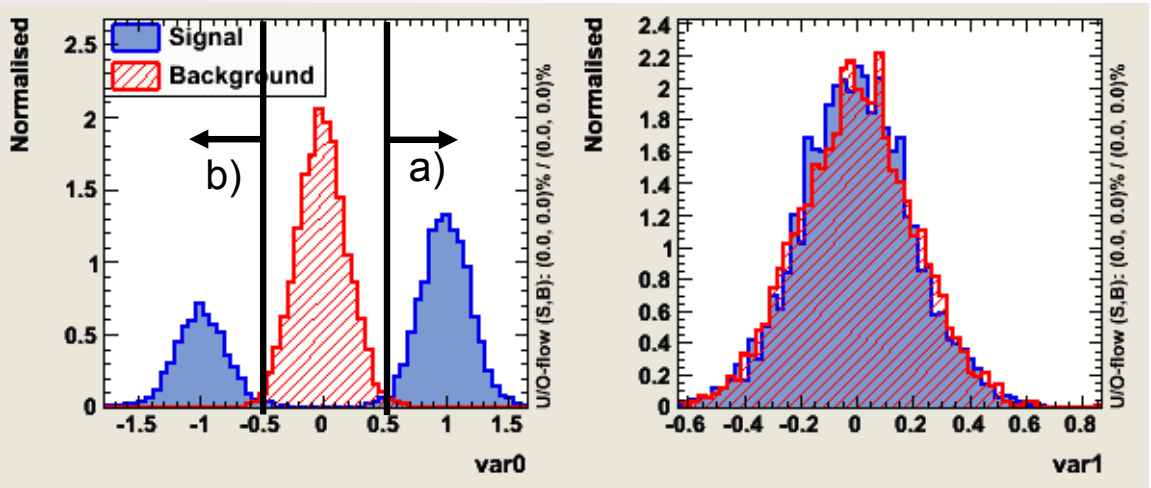
- train first tree on the normal, full MC sample
  $\rightarrow$ this tree will be as as correct as a single tree can be
- train a second tree to correct the errors of the first tree:
  - give lower weight to events which have been classified correctly
  - give higher weight to events which have been classified wrongly
- train third tree to correct the errors of the first two trees
- iterate

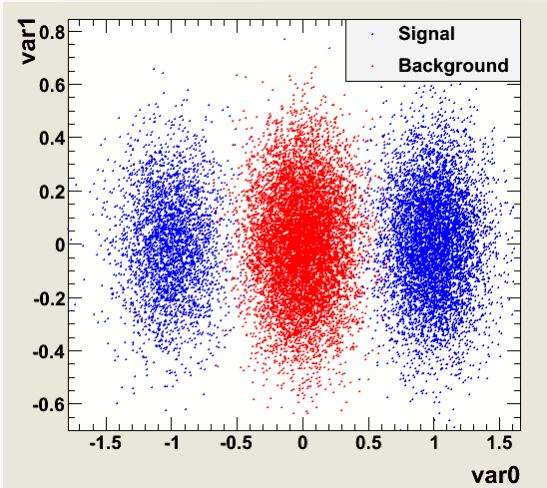
AdaBoost: A simple demonstration

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

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







Two reasonable cuts: a) Var0 > 0.5 → ε_{signal}=66% $\varepsilon_{bkg} \approx 0\%$ misclassified events in total 16.5%

or

b) Var0 < -0.5 → ε_{signal}=33% $\varepsilon_{bkg} \approx 0\%$ misclassified events in total 33%

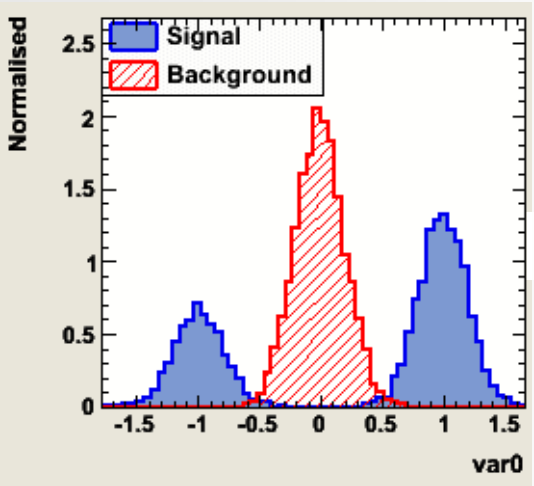
the training of a single decision tree stump will find “cut a)”

AdaBoost: A simple demonstration

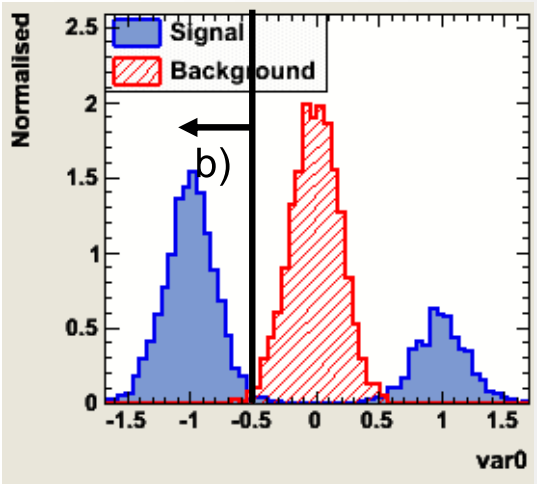
The first “tree”, choosing cut a) will give an error fraction: err = 0.165

before building the next “tree”: weight wrong classified training events by (1-err)/err ≈ 5

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



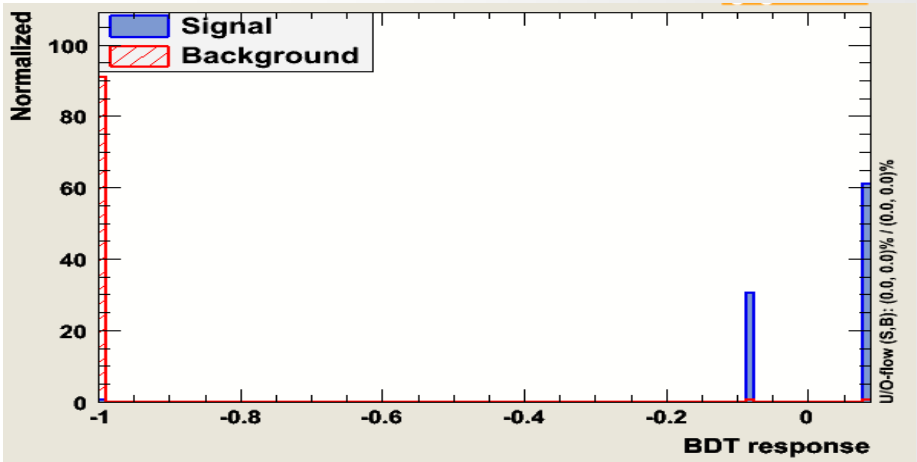
re-weight



.. and hence will chose: “cut b)”: 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



the last tree

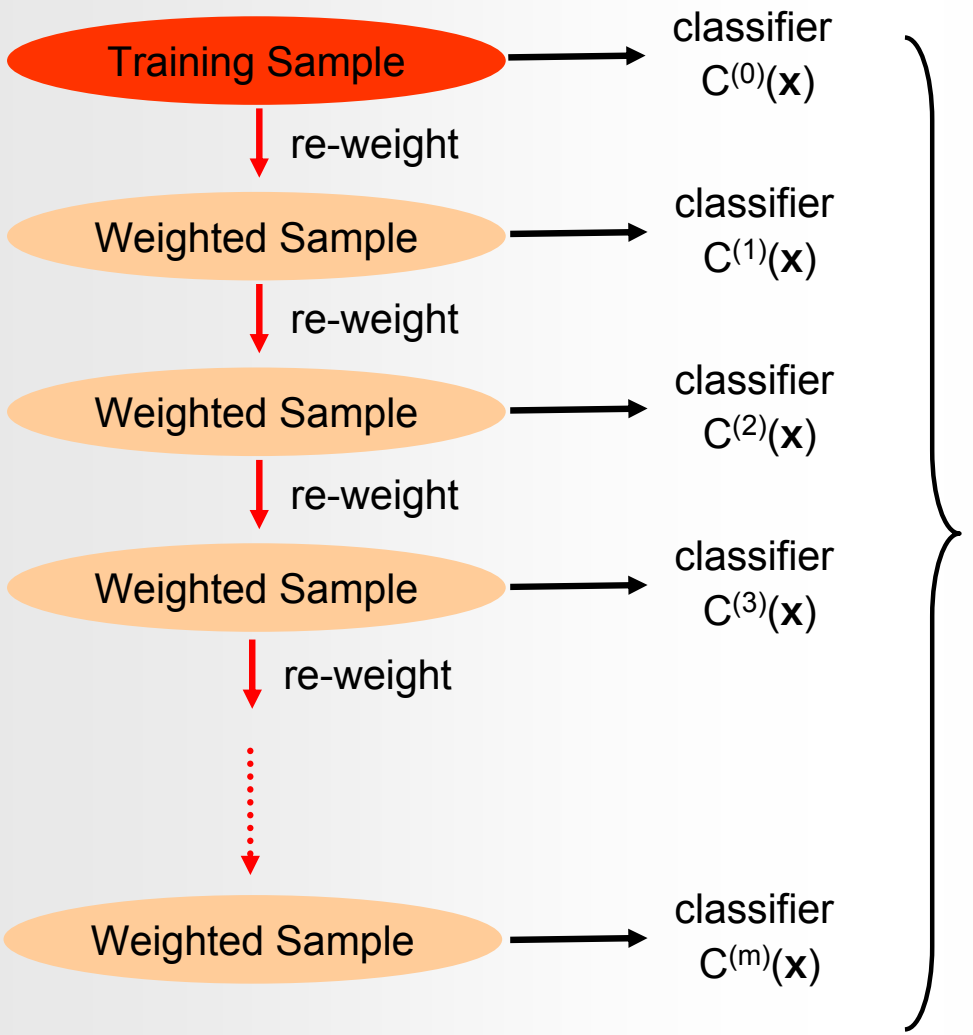
From this example it should be clear that the last tree performes worse than the first tree!

the last tree will do almost everything wrong

- weights have nothing to do with your actual data anymore
- weighted signal and weighted background will look almost identical
- last tree tries to separate those events which are unseparable
- last tree will not remove any easy removable background

! the last tree is only relevant for those events where all other trees have no clue what to do

Adaptive Boosting (AdaBoost)



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

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

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

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

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

final words on BDTs

- learning speed can be manipulated (AdaBoostBeta)

  $$\frac{1-f_{\text{err}}}{f_{\text{err}}} \rightarrow \left(\frac{1-f_{\text{err}}}{f_{\text{err}}}\right)^{\beta}$$

- leaves as training parameters:
  - number of trees
  - AdaBoostBeta
  - tree depth
  - MinNodeSize
  - nCuts
- can be optimised automatically with

```
factory->OptimizeAllMethods("SigEffAtBkgEff001","FitGA");
factory->OptimizeAllMethods("ROCIntegral","Scan");
```

! keep in mind that this turns your test sample into a training sample

$\rightarrow$ on to part 3 or not

lustiges bild hier