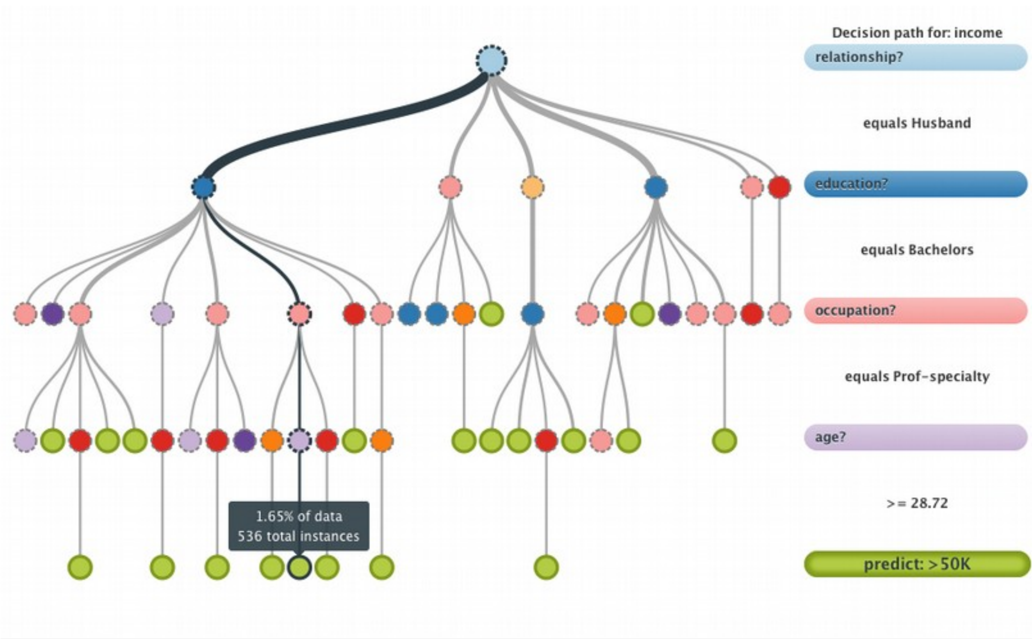
Machine learning
Lecture 2



Marcin Wolter

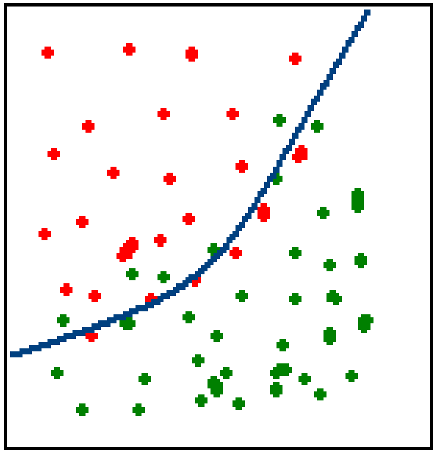
IFJ PAN

12 February 2019

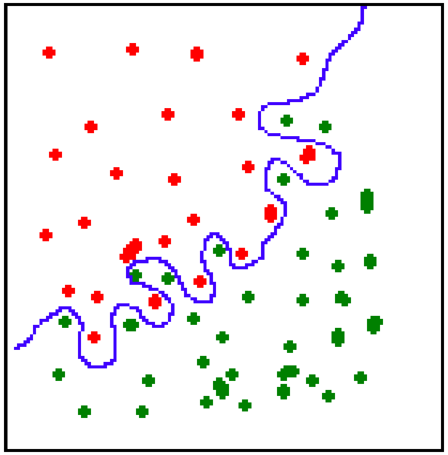
- Simple non-linear methods like naive Bayes classifier, k-nearest neighbours, Parzen kernel methods.
- <u>Boosted Decision Trees (BDT)</u>

Overtraining

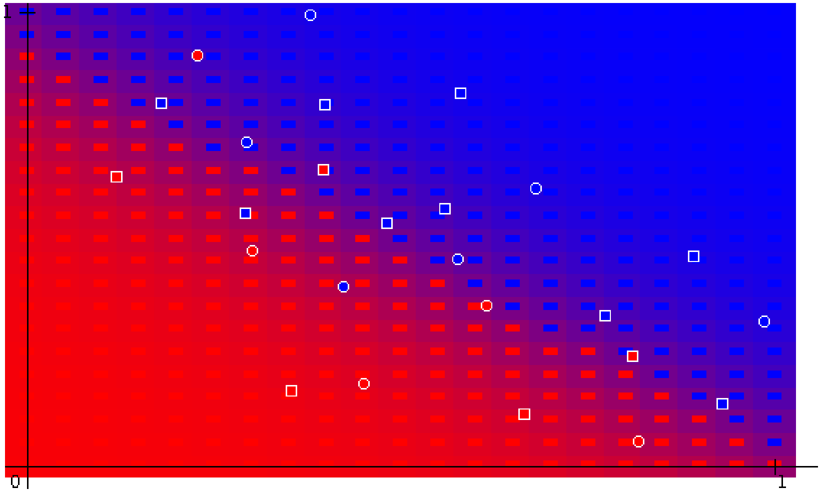
- **Overtraining** – algorithm “learns” the particular events, not the rules.
- This effect important for all ML algorithms.
- Remedy – checking with another, independent dataset.

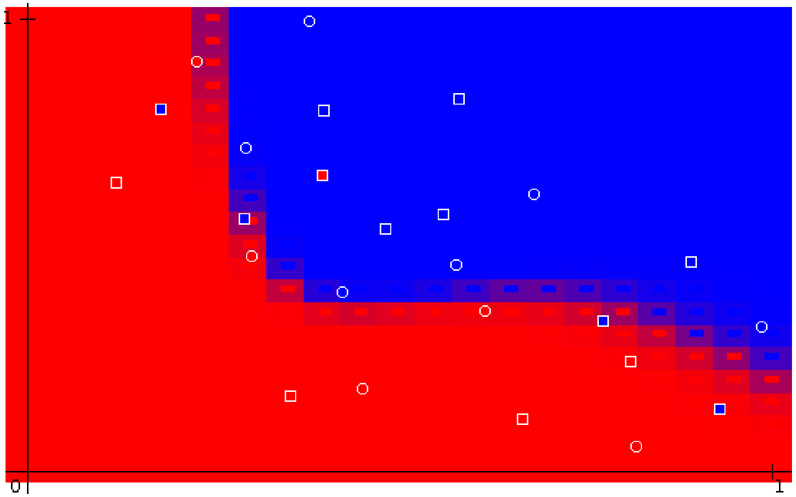


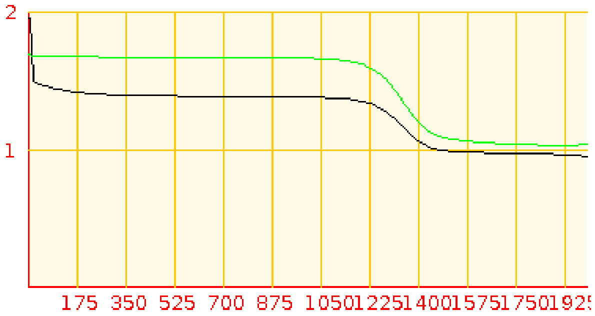
Correct



Overtraining

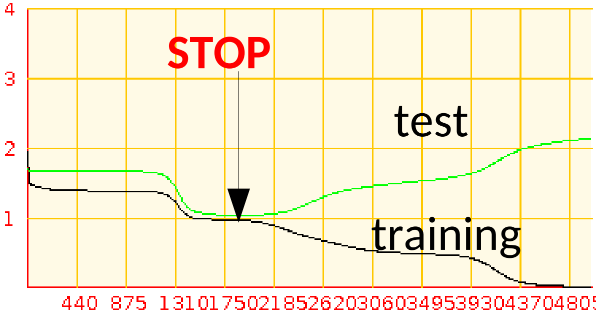






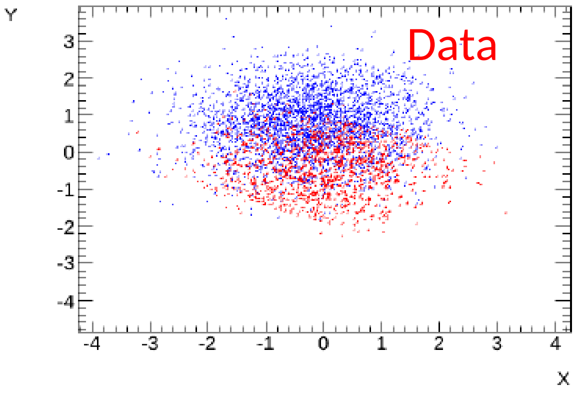
● ● Training sample

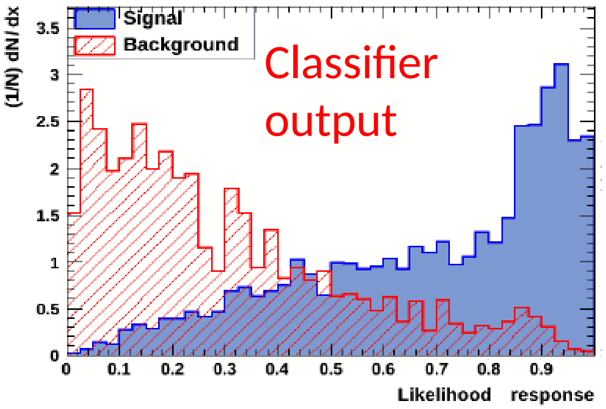
🟦 🟥 Test sample



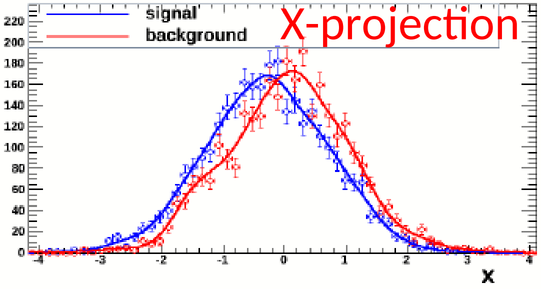
Example of using Neural Network.

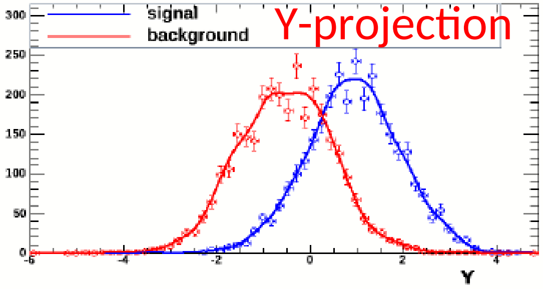
Naive Bayes classifier





Frequently called **“projected likelihood" (TMVA).**





- Based on the assumption, that variables are independent (so „naive"):

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

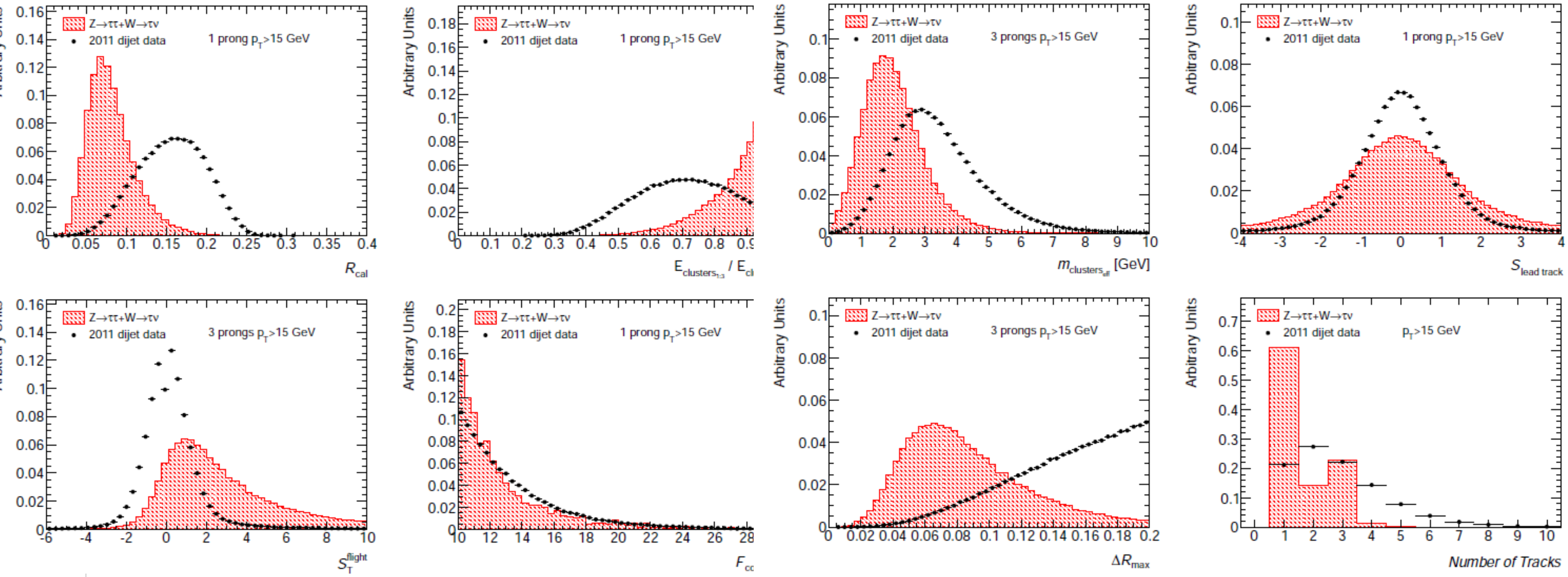
“Naive” assumption: $P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \boxed{\prod_{i=1}^{n} P(x_i \mid y)}}{P(x_1, \dots, x_n)}$$

- Output probability is **a product of probabilities for all variables.**
- Fast and stable, not optimal, but in many cases sufficient.

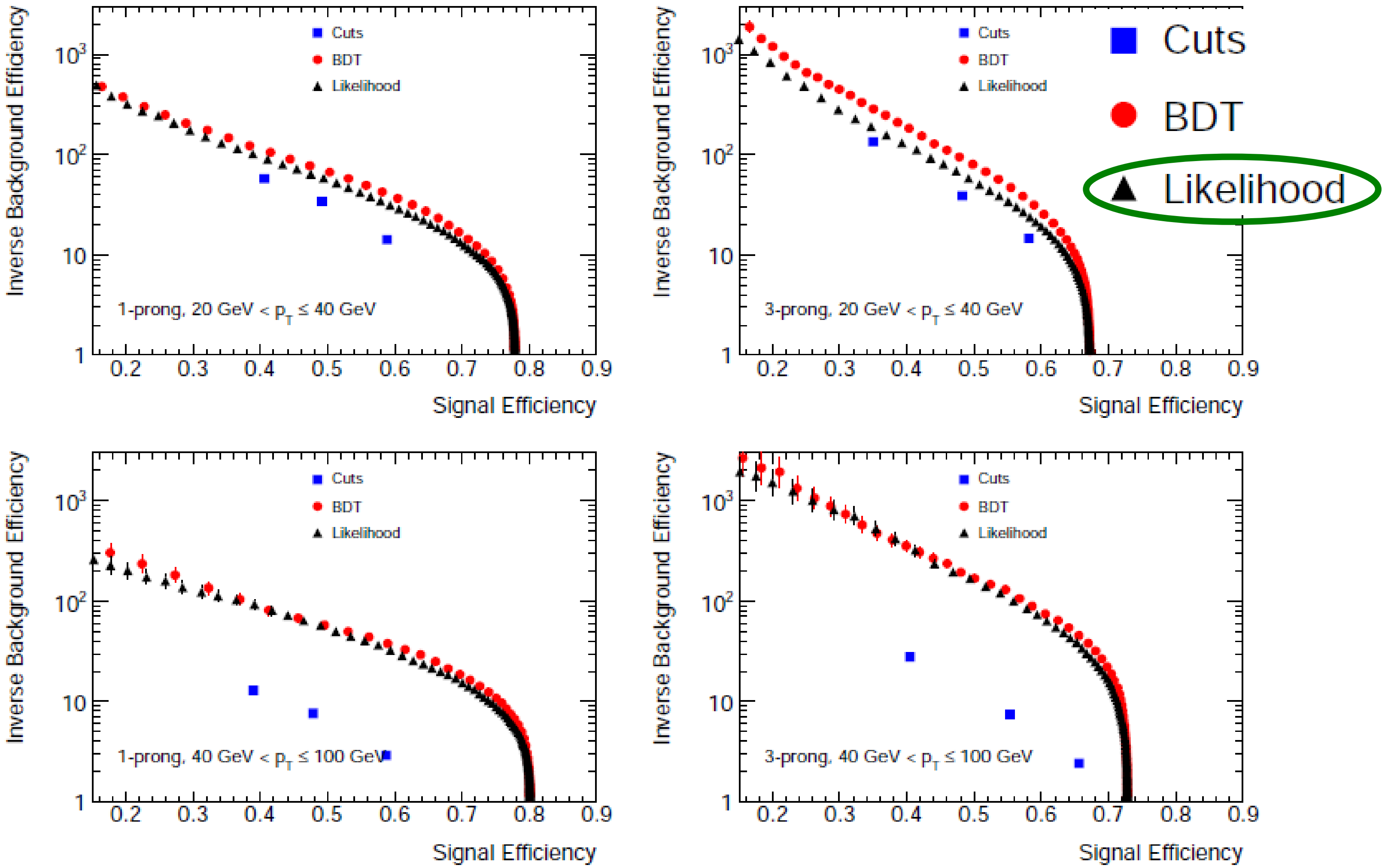
Tau identification in ATLAS experiment

- Many identification variables, none of them gives sufficient identification: => use information from all variables.



Identifying variables

Tau identification - results



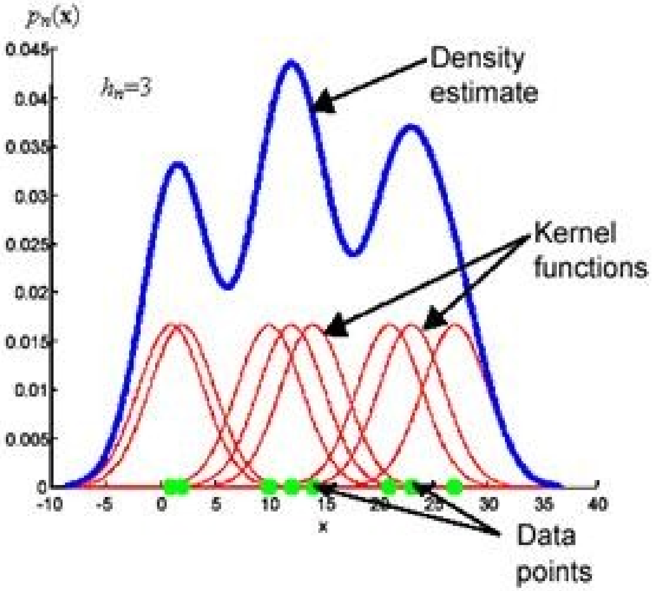
- **Cuts – well known.**
- **Likelihood (Naive Bayes) – just described.**
- **<u>Boosted Decision Trees (BDT)</u> – we will describe soon.**

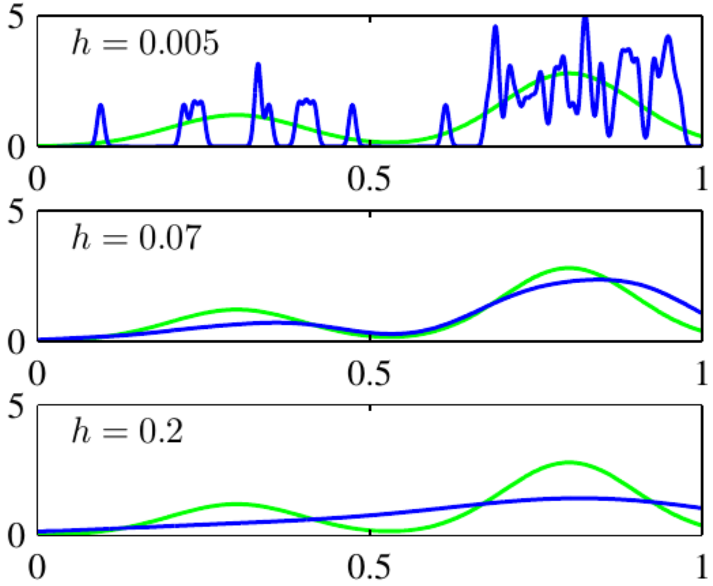
Kernel density estimators

Approximation of the unknown probability density as a sum of kernel functions centered in the points x_n of the training sample (Parzen, years 1960-ties).

$$D(x) = \frac{P(S)}{P(S) + P(B)}$$

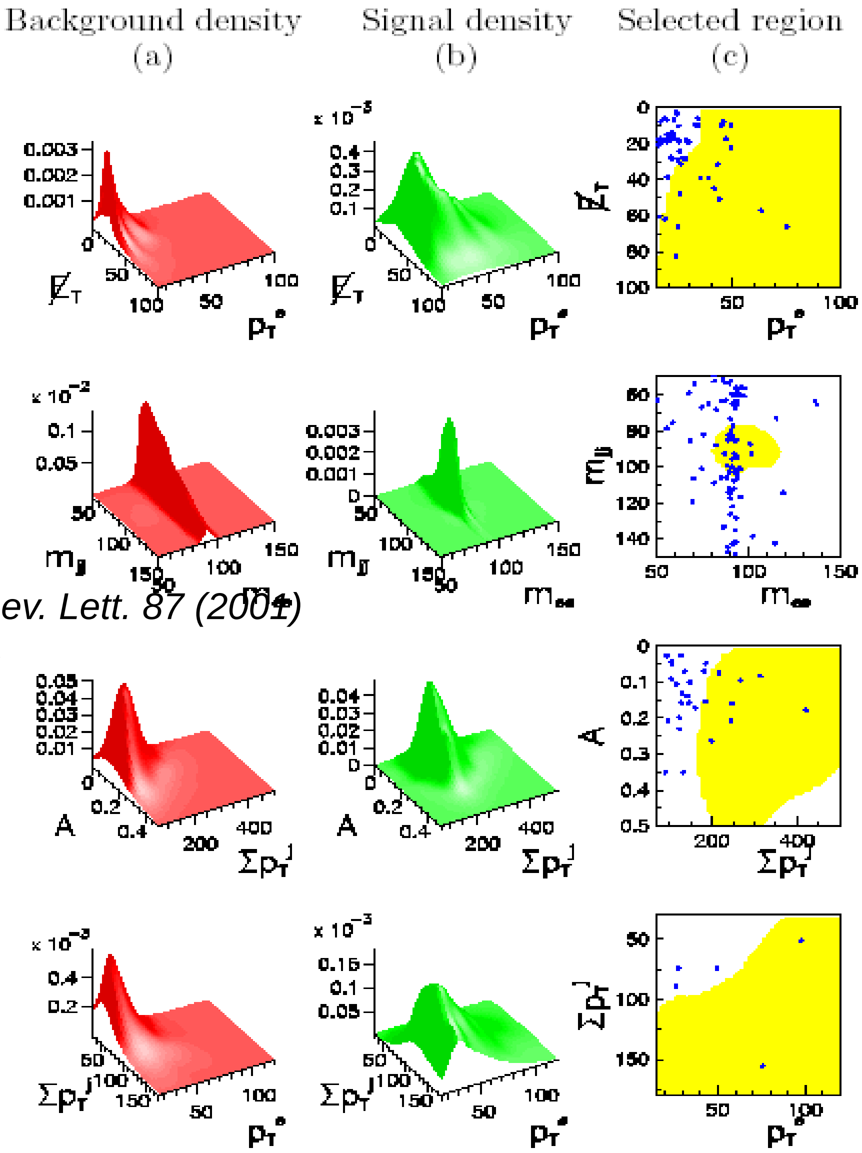
- Typical kernel functions: Gauss, $1/x^n$ itp.
- Simple idea, but using this method requires a lot of memory and CPU.





Approximated probability density (blue) comapred to the true probability density (green) as a function of the width of the Gaussian kernel function. We can see, that the width is a smoothing parameter.

QUAERO package from the D0 experiment



Phys. Rev. Lett. 87 (2001) 231801,

FIG. 1. The background density (a), signal density (b), and selected region (shaded) (c) determined by QUAERO for the standard model processes discussed in the text. From top to bottom the signals are: $WW \to e\mu\not{E}_T$, $ZZ \to ee\,2j$, $t\bar{t} \to e\not{E}_T\,4j$, and $t\bar{t} \to e\mu\not{E}_T\,2j$. The dots in the plots in the rightmost column represent events observed in the data.

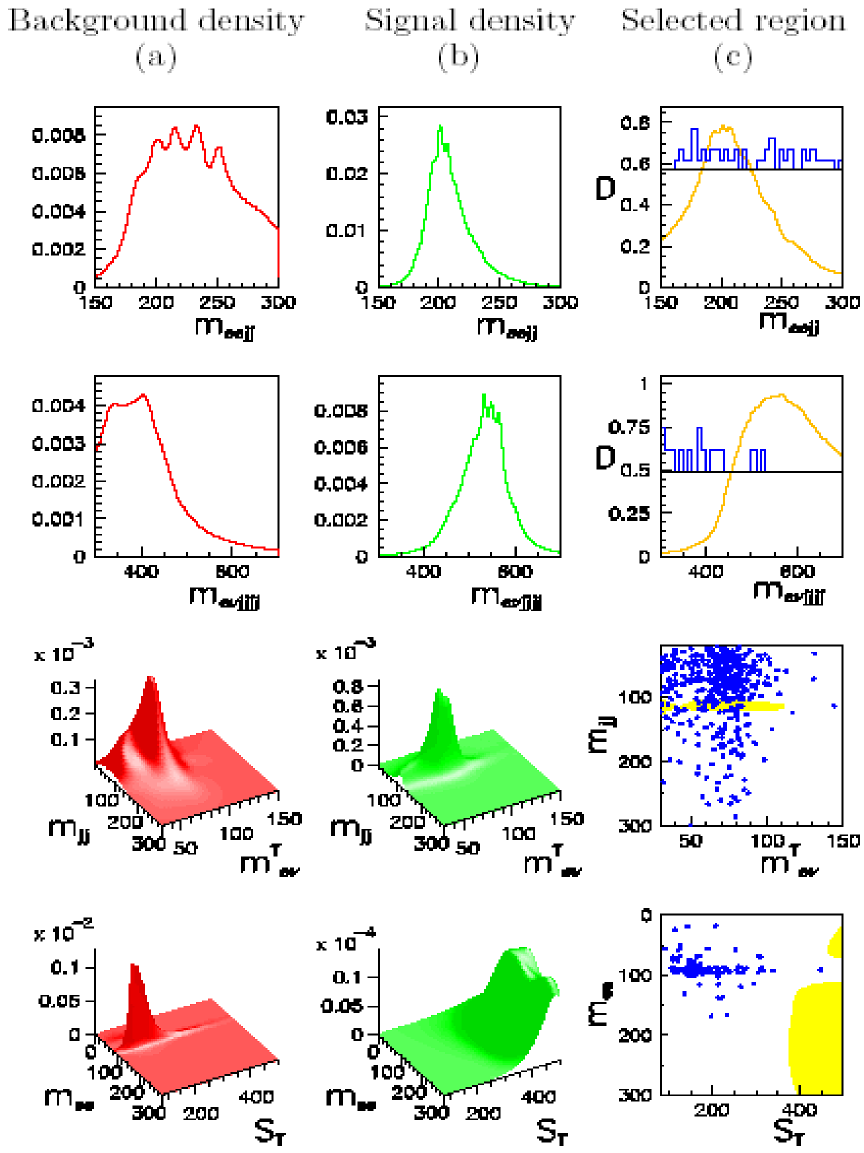


FIG. 2. QUAERO's analysis of signatures involving undiscovered particles. From top to bottom the hypothetical signals are: $h_{200} \to ZZ \to ee\,2j$, $Z'_{350} \to t\bar{t} \to e\not{E}_T\,4j$, $Wh_{115} \to e\not{E}_T\,2j$, and $LQ_{225}\overline{LQ}_{225} \to ee\,2j$. Plots (c) of the first two rows show the discriminant D (curve), the threshold $D_{\rm cut}$ (horizontal line), and the data (histogram); the region with $D > D_{\rm cut}$ is selected.

PDE_RS – extension of the Parzen methods

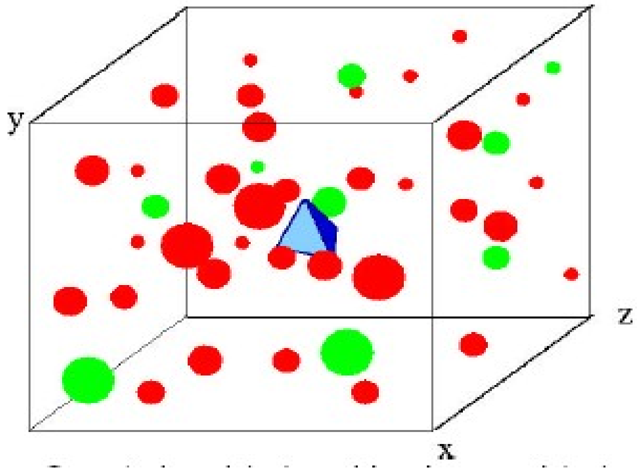
**published by T. Carli, B. Koblitz, NIM A 501 (2003) 576-588*

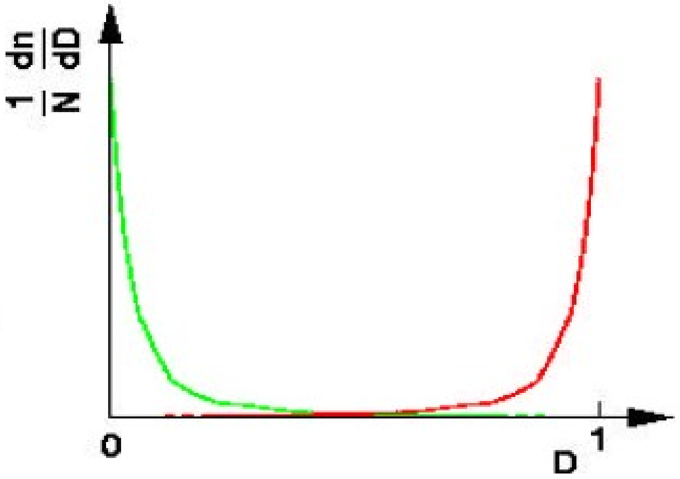
Counts signal (n_s) and background (n_b) events in N-dimensional cube around the classified event – only few events from the training sample are needed.

Size of the cube – a free parameter.

Discriminator *D(x)* :

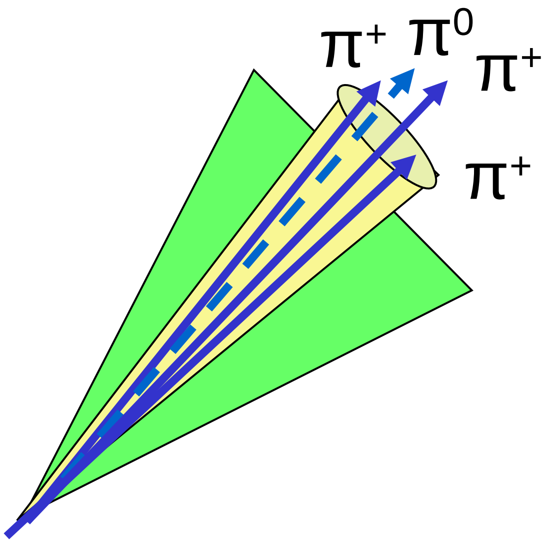
$$D(x)=\frac{n_S}{n_S+n_b}$$





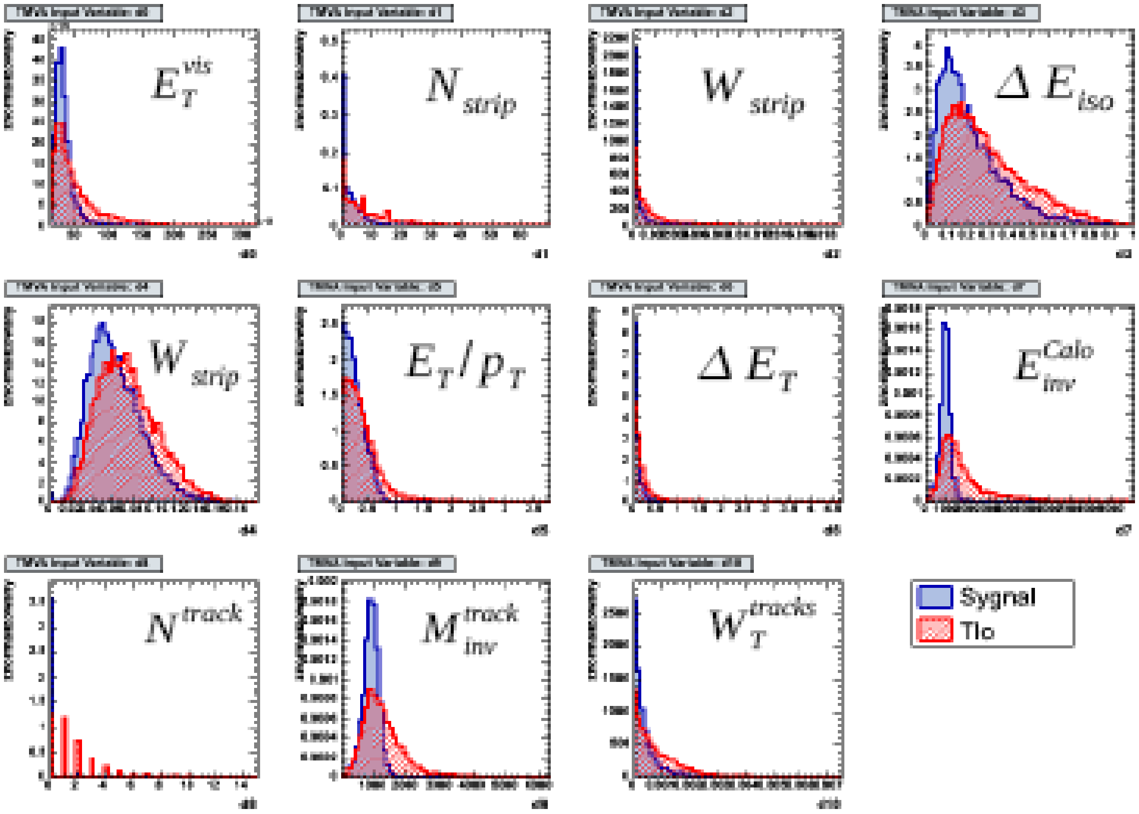
- Simple analysis.
- Events stored in the binary tree – the neighbor events are found quickly.
- It’s a special case of Parzen method – the kernel function: $f(x)=\begin{cases}1\,(|x|\le d)\\0\,(|x|>d)\end{cases}$

Tau lepton identification in ATLAS experiment

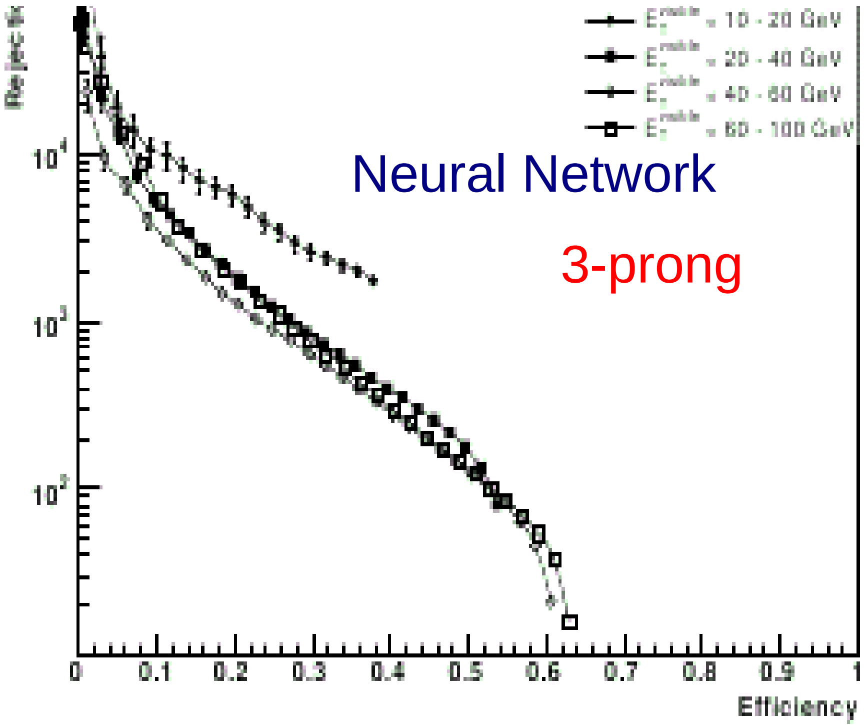


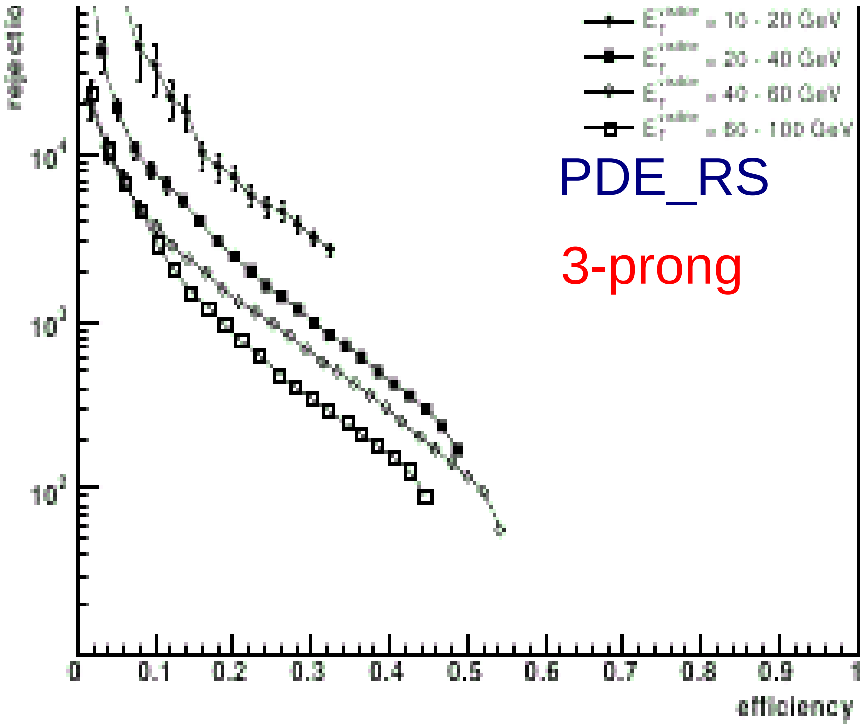
None of the variables separates tau’s from background sufficiently well – we need to use all of them.

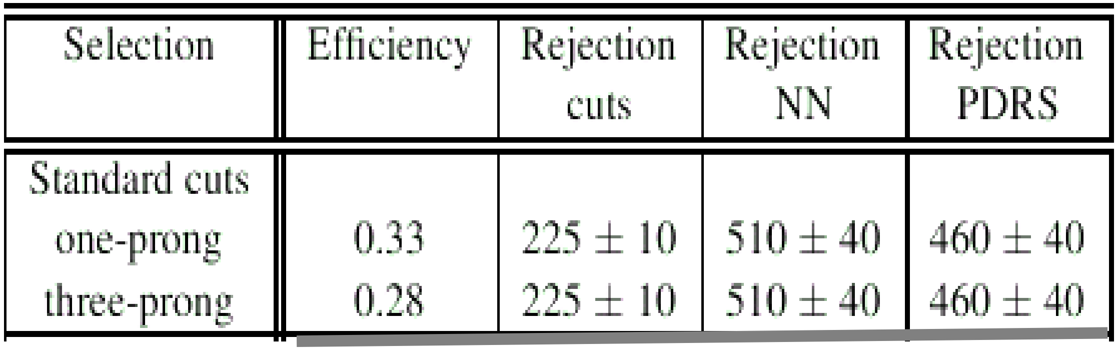
3-prong events



Identification using Neural Network and PDE_RS







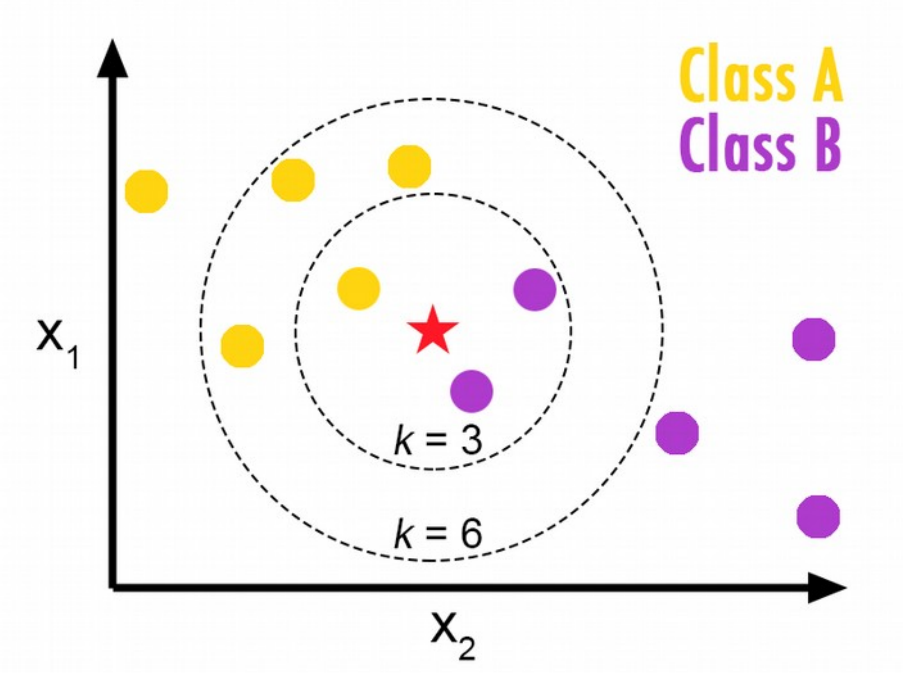
Selection	Efficiency	Rejection cuts	Rejection NN	Rejection PDRS
Standard cuts				
one-prong	0.33	225 ± 10	510 ± 40	460 ± 40
three-prong	0.28	225 ± 10	510 ± 40	460 ± 40

Significant improvement comparing to the cut-based analysis.

KNN - k nearest neighbors

- Proposed already in 1951
- An event is qualified to the class, to which belongs the majority of it’s k nearest neighbors,
- or we calculate the probability of belonging to the class “signal” as:

$$p(S|x) = \frac{k_S}{k}$$



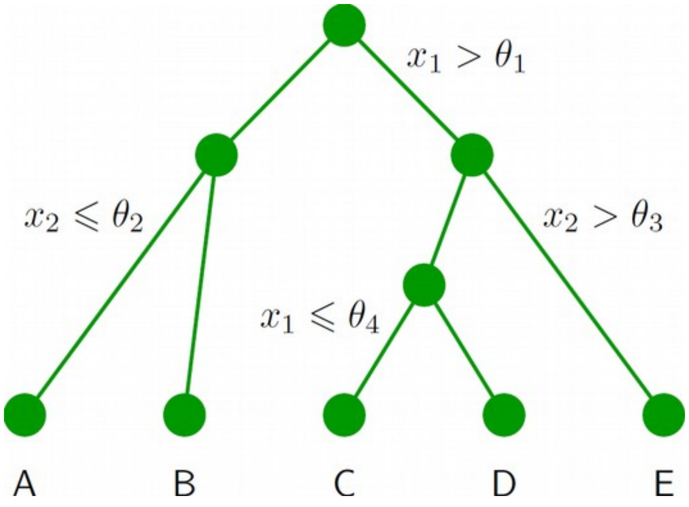
Boosting, bagging, BDT...
Ensemble learning

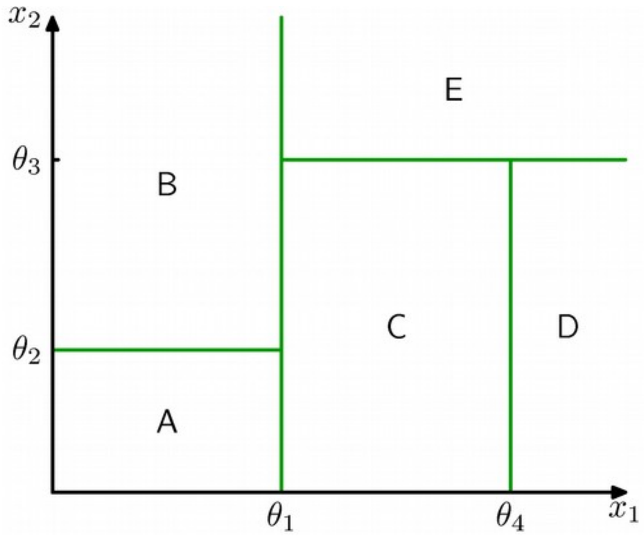
What does BDT mean???

- **BDT – Boosted Decision Tree:**
  - **Decision Tree** – an algorithm know for a long time, used in most of the expert systems. For example, the first aid manual is frequently written in the form of a decision tree: if (condition) do something, else do something different, then check another condition...
  - **Boosted** - a method of joining many weak classifiers in an ensemble to get one strong classifiers. It is not limited to decision trees, however with them it is most commonly used.

Decision trees

- Decision tree – a series of cuts, each „leaf” (A,B,C,D,E) has a label, for example ”signal” and “background”.

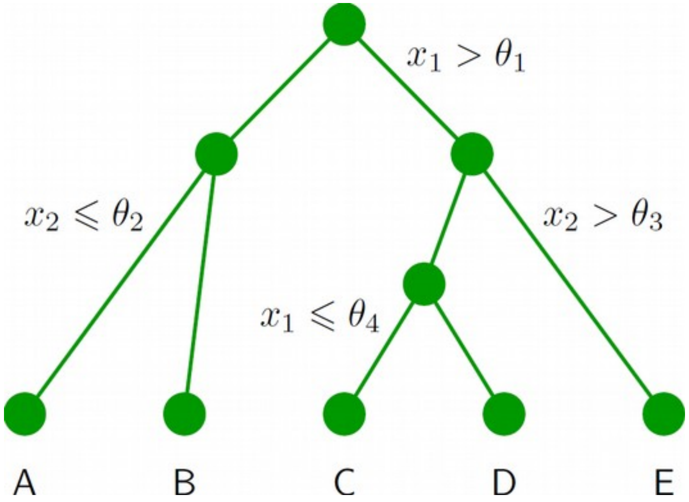




- Easy in visualization and interpretation
- Resistant for *outliers*.
- Weak variables are ignored.
- Fast training and classification.
- Unfortunately: **sensitive for fluctuations, unstable**.

Building the tree

- We start from the root.
- We divide the training sample by the best separating cut on the best variable.
- We repeat the procedure until the stopping conditions are fulfilled, for example: number of leafs, number of events in a leaf etc.
- The ratio S/B in a leaf defines the classification (binary signal or background, or a real number giving the probability, that a given event is a signal).



Definitions of separation:

- Gini inpurity: (*Corrado Gini 1912, invented the Gini index used to measure the inequality of incomes*)

  p (1-p) : p= P(signal), *purity*
- Cross-entropy:

  -(p ln p + (1-p)ln(1-p))
- Missidentification:

  1-max(p,1-p)

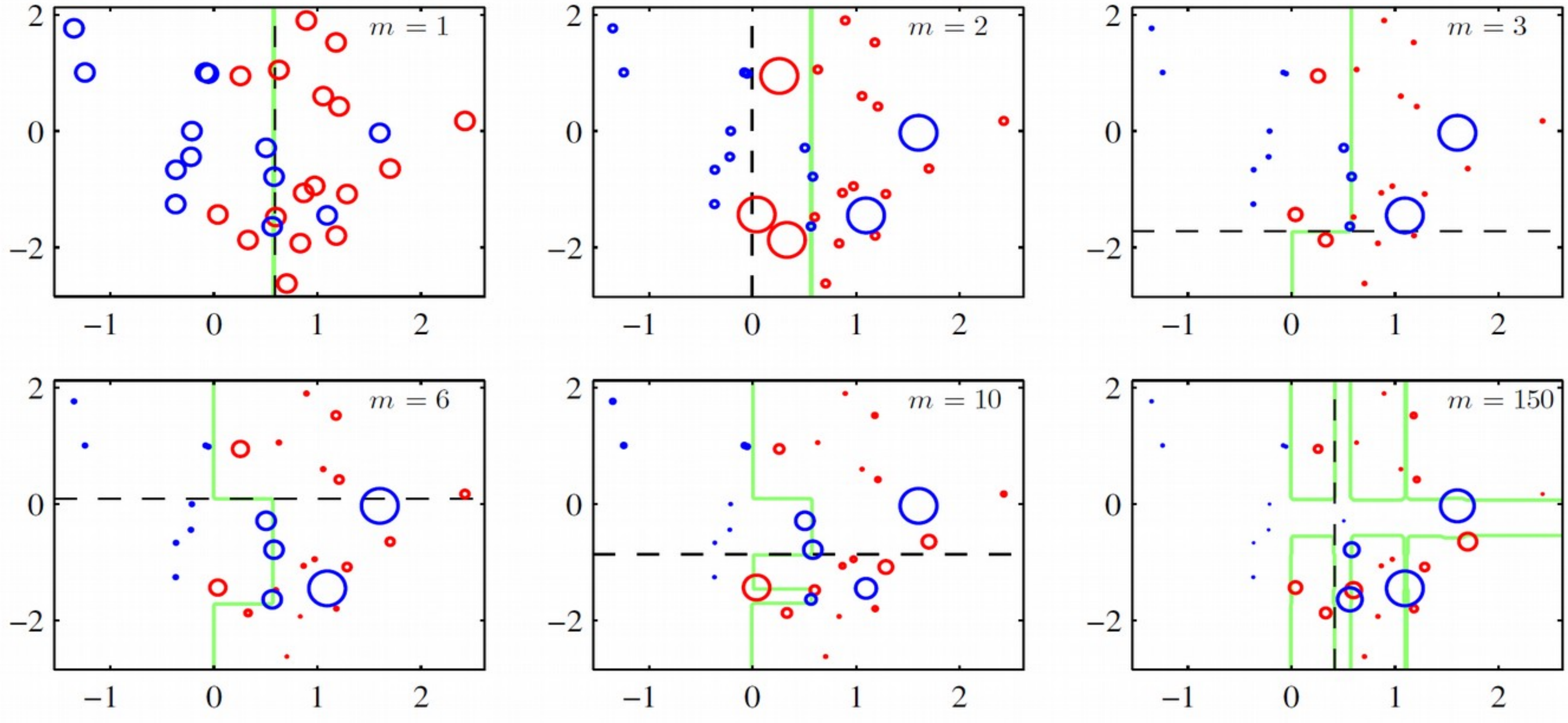
AdaBoost – ensemble of classifiers

Problem: could we improve a weak classifier?

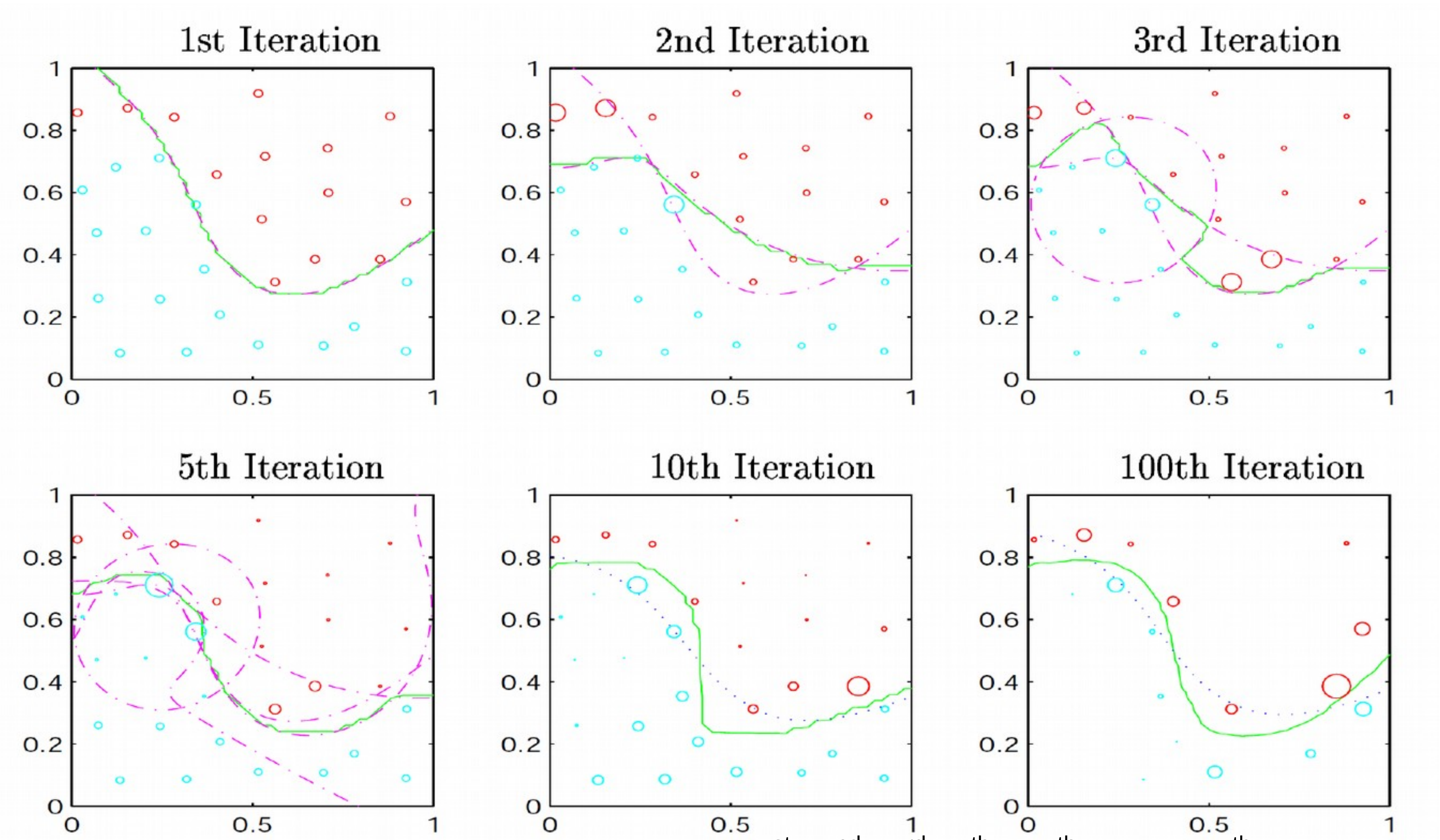
Answer: yes, by applying it many times.

- An algorithm used most frequently: **AdaBoost** (Freund & Schapire 1996 – Gödel prize)

- Build a decision tree.
- Increase the weights of wrongly classified events.
- Repeat many times (typically 100-1000)
- Classify the events by “voting” by all the trees.



AdaBoost



AdaBoost for 2-dimensional data – results after 1st, 2nd, 3rd, 5th, 10th and 100th iteration. The solid green line shows the results of the combined classifier, the dashed line the borders between classes obtained after each step. For the last two plots the dotted line marks the class borders obtained from the bagging algorithm (we will talk about bagging soon).

AdaBoost

- Five years after publication of AdaBoost Friedman proved, that the algorithm in fact minimizes the exponential loss function:

$$E = \sum_{n=1}^{N} \exp\left(-t_n f_m(x_n)\right)$$

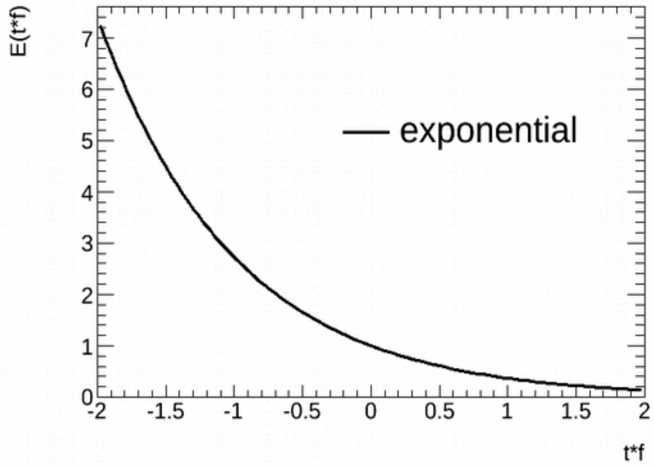
where *f(x)* is the answer of the algorithm

t = 1 signal, *t = -1* background

$t_n \cdot f_m(x_n) > 0$ – correctly classified

$t_n \cdot f_m(x_n) < 0$ – incorrectly classified

- Exponential function goes up quickly => huge punishment for wrongly classified events, so the algorithm is sensitive for single, outstanding points. Classification becomes worse, when data are hardly separable.



  - Another loss function?
  - Friedman in year 2000 proposed few other loss functions, but AdaBoost is still the most popular (we will talk about).

AdaBoost in action

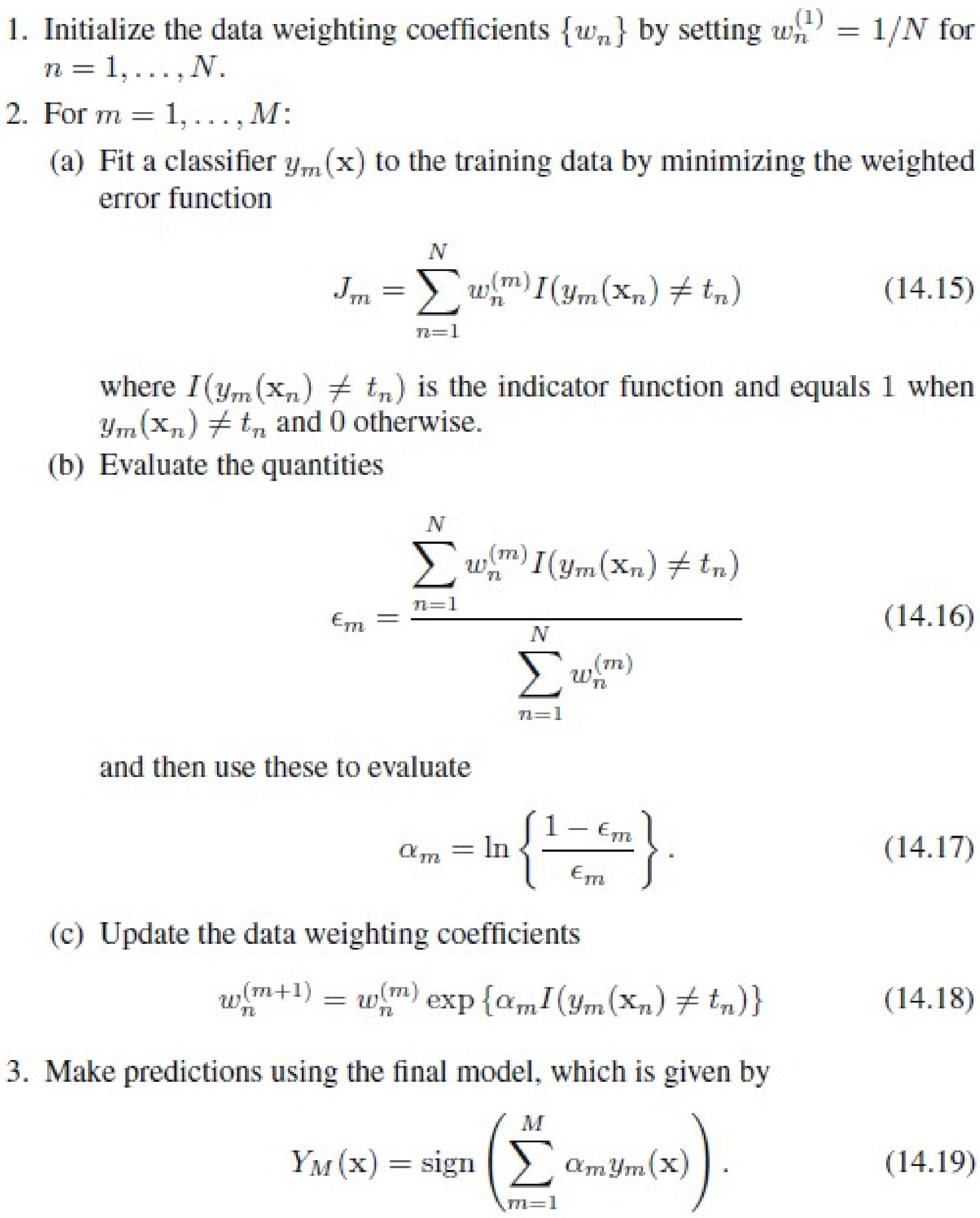
AdaBoost in Action

Kai O. Arras

Social Robotics Lab, University of Freiburg

Nov 2009 Social Robotics Laboratory

AdaBoost



1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathrm{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathrm{x}_n) \neq t_n) \qquad (14.15)$$

   where $I(y_m(\mathrm{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathrm{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathrm{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \qquad (14.16)$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \qquad (14.17)$$

   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp \left\{ \alpha_m I(y_m(\mathrm{x}_n) \neq t_n) \right\} \qquad (14.18)$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathrm{x}) = \operatorname{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathrm{x}) \right). \qquad (14.19)$$

C. Bishop, Pattern recognition and machine learning. Springer, 2009.

Classifier boosting

- Boosting, bagging – in a “magical” way we get a strong classifier out of weak ones

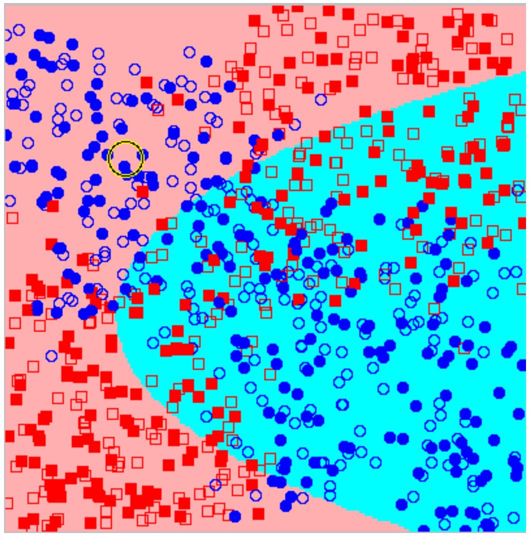
- **Typically boosting used on decision trees – Boosted Decision Trees BDT.**

- Good results without time consuming tuning of parameters:

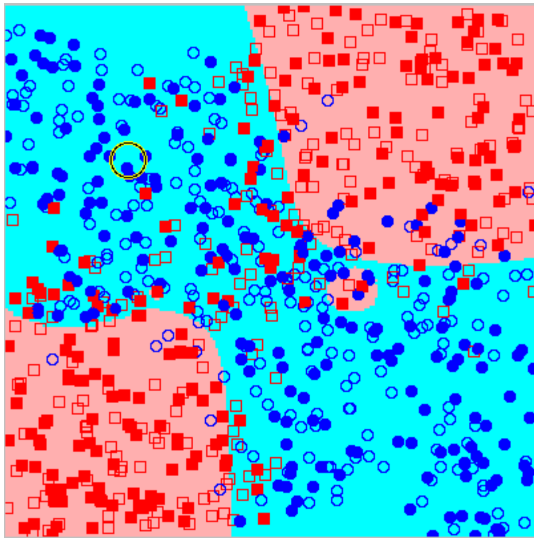
  „*the best out-of-box classification algorithm*”.

- Relatively resistant on overtraining.

- Quite frequently used nowadays. And with good results!



Naive Bayes classifier



Boosted Naive Bayes classifier

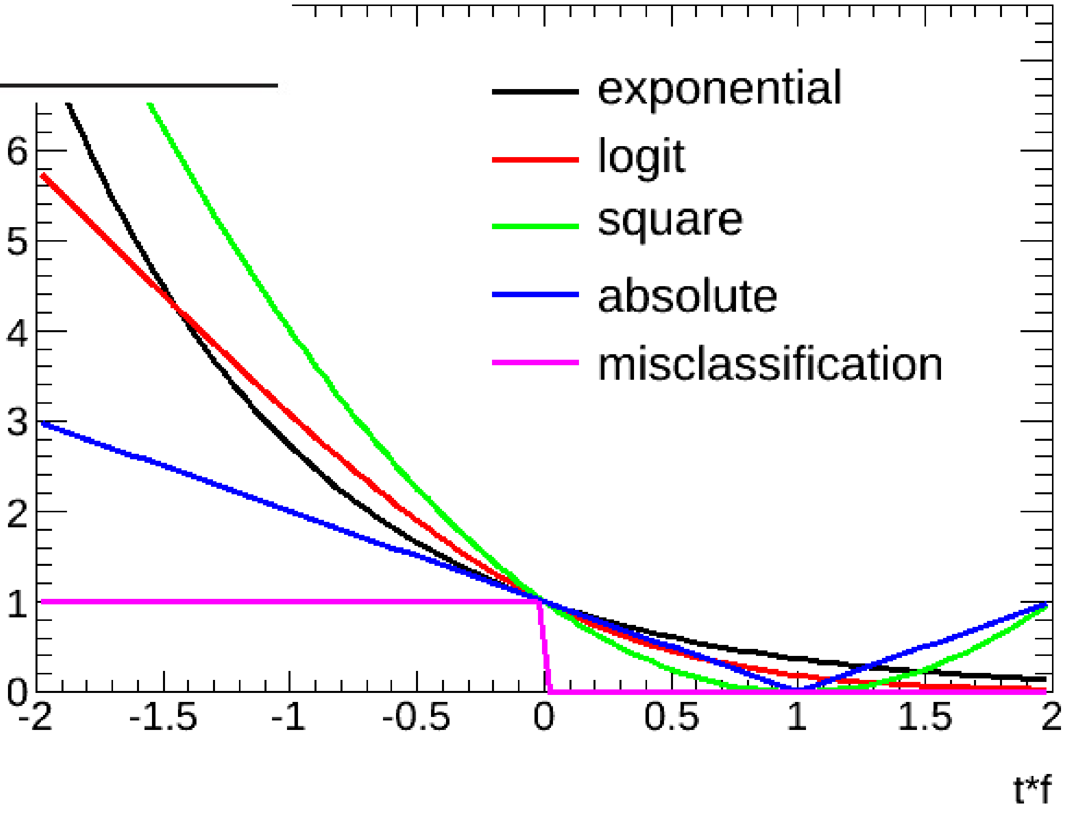
Application of a boosting algorithm (5 iterations) for Naive Bayes classifier.

Boosting – different loss functions

- Another loss functions:

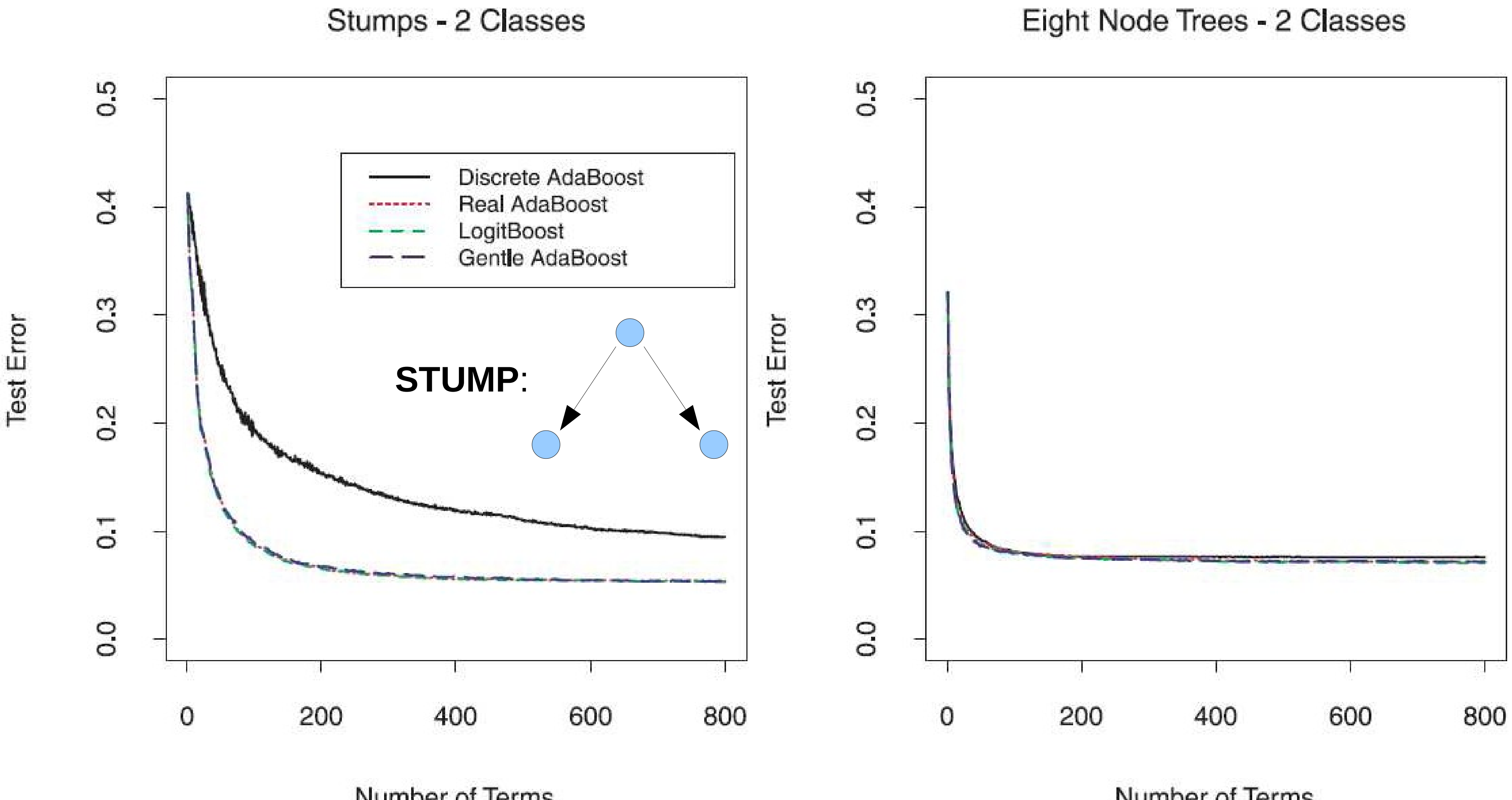
Nazwa funkcji	Równanie	Nazwa algorytmu
wykładnicza	$\exp\left(-t_n f_m(x_n)\right)$	*AdaBoost*
prawdopodobieństwa	$\log\left(1+\exp(-2t_n f_m(x_n)\right)$	*LogitBoost*
błąd kwadratowy	$\left(t_n - f_m(x_n)\right)^2$	*SquareBoost*
błąd absolutny	$\lvert t_n - f_m(x_n)\rvert$	
zero-jedynkowa	$\begin{cases} 0 & sign(f_m(x_n)) \neq t_n \\ 1 & sign(f_m(x_n)) = t_n \end{cases}$	

Pro:
Not so sensitive for outliers.
The properly chosen “sensitivity” is important, when there are regions with mixed signal and background.
Contra:
There is no fast and simple minimization algorithm like AdaBoost => minimization using general algorithms, like for function fit.



Boosting – different loss functions

- Discrete AdaBoost – the basic AdaBoost described on previous slides
- Real AdaBoost, LogitBoost, Gentle AdaBoost- modifications (different loss functions and minimization algorithms) proposed by Friedman.
- For bigger trees the results are identical.

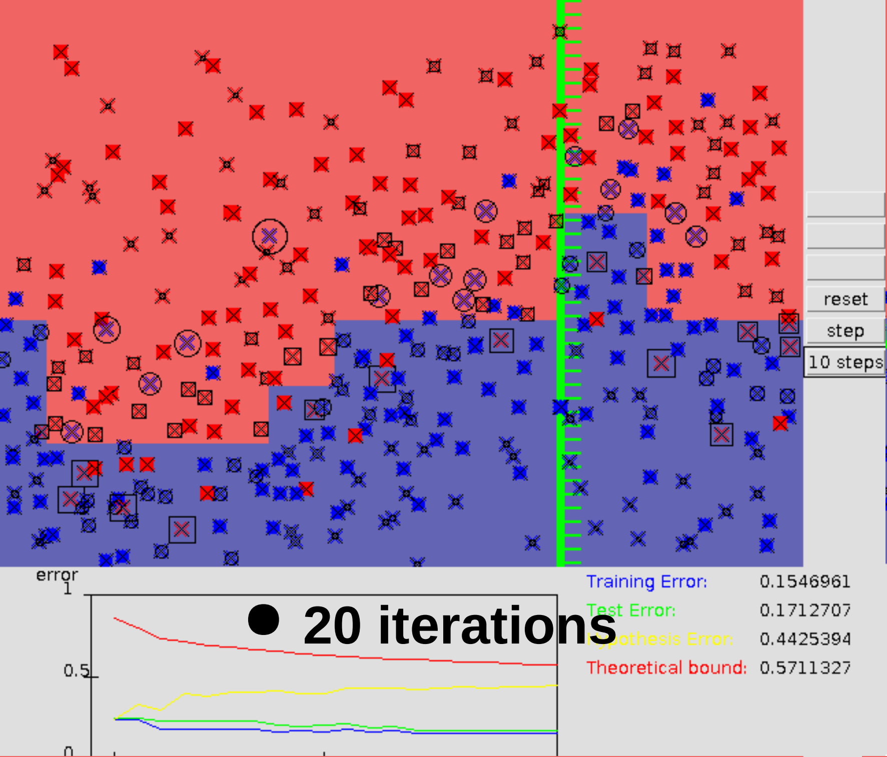


STUMP:

Overtaing of the Boosted Decision Tree (BDT)

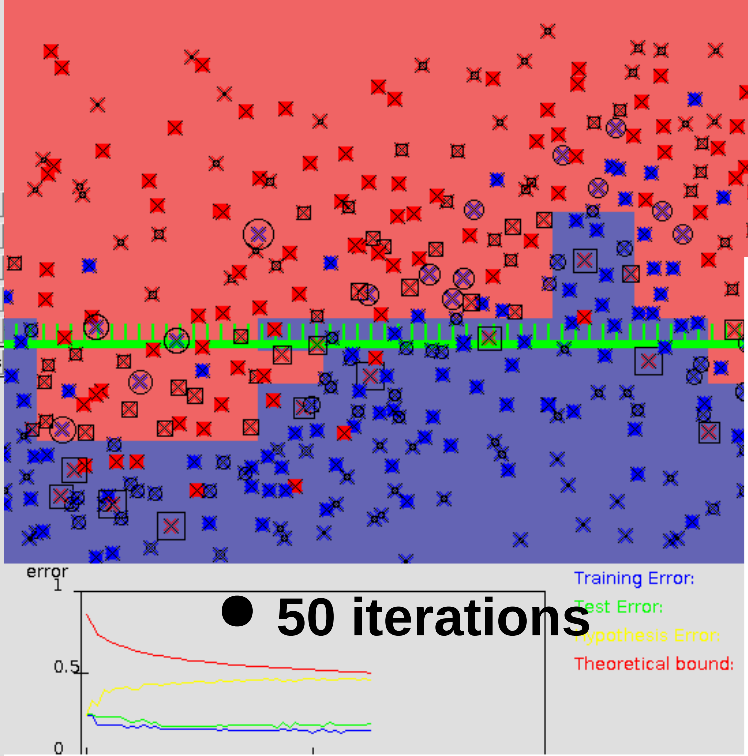
Algorithm uses stumps – decision trees with two branches.

At the end the space is divided into very small areas - overtraining.



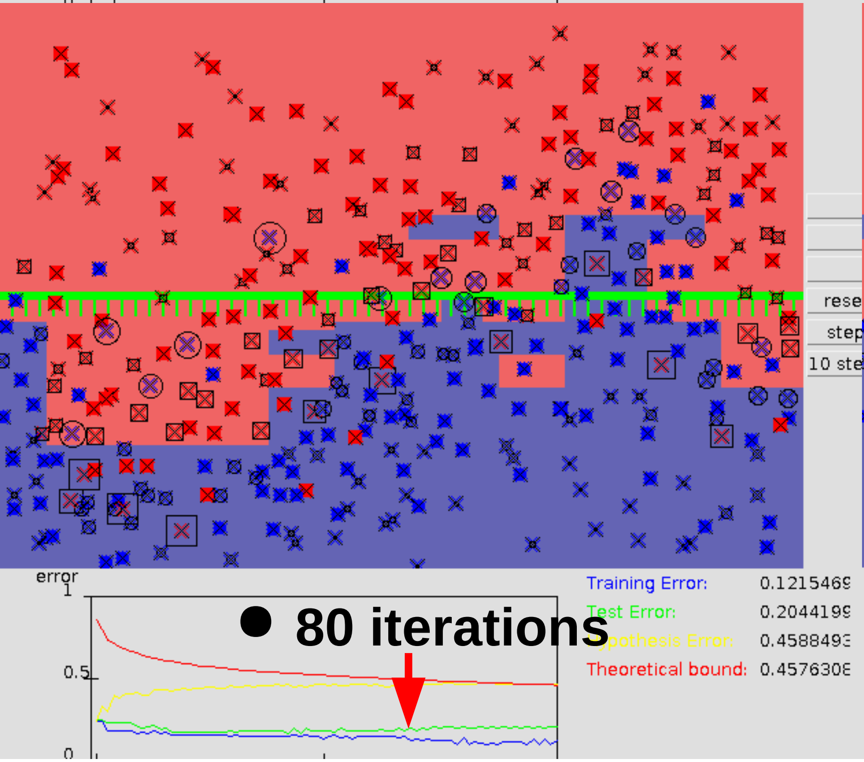
- 20 iterations

```
Training Error:       0.1546961
Test Error:           0.1712707
Hypothesis Error:     0.4425394
Theoretical bound:    0.5711327
```



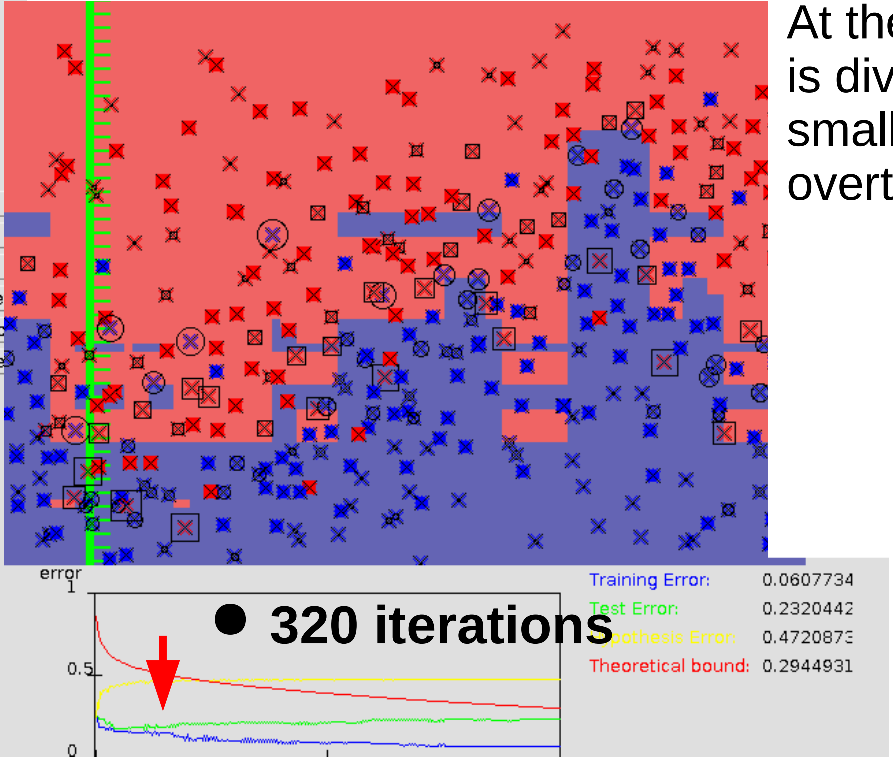
- 50 iterations

```
Training Error:
Test Error:
Hypothesis Error:
Theoretical bound:
```



- 80 iterations

```
Training Error:       0.1215469
Test Error:           0.2044199
Hypothesis Error:     0.4588493
Theoretical bound:    0.4576308
```

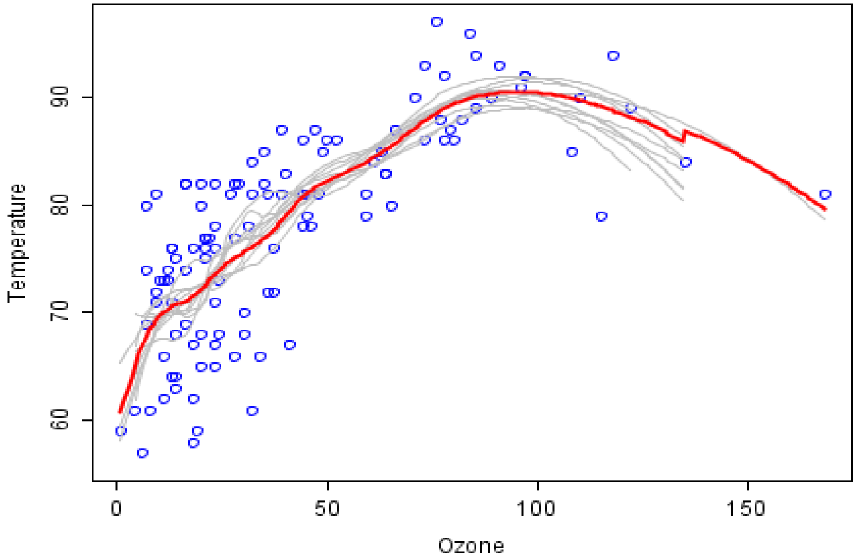


- 320 iterations

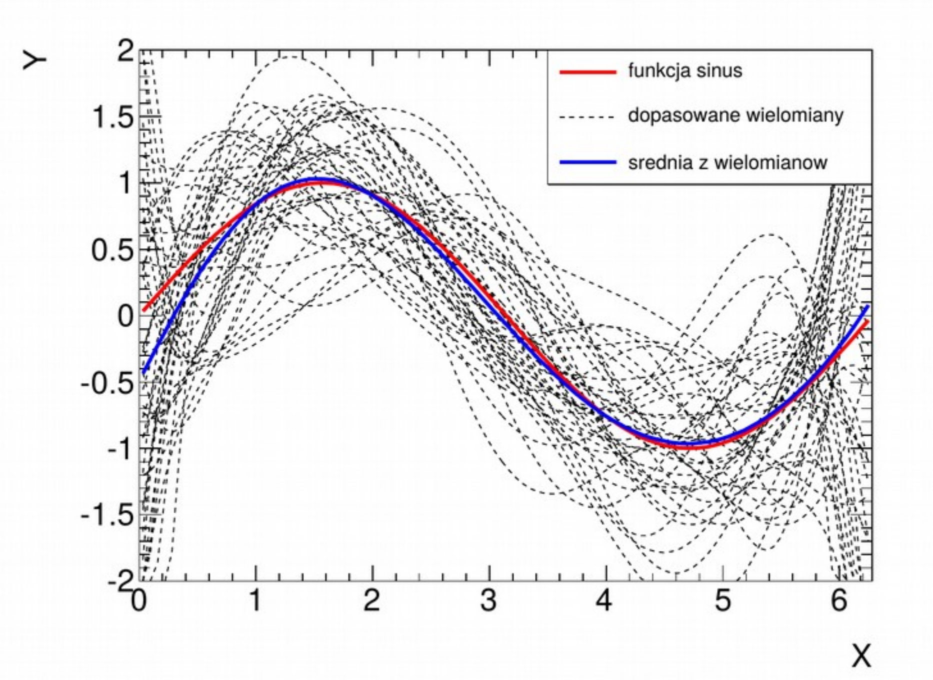
```
Training Error:       0.0607734
Test Error:           0.2320442
Hypothesis Error:     0.4720873
Theoretical bound:    0.2944931
```

Bagging (Bootstrap AGGregatING)

- **Algorithm proposed by Leo Breiman in 1994:**
  - Take N events from N-element training set, but with repetitions.
  - Train the classifier on this set.
  - Repeat many times
  - Classify new events by voting of all the classifiers.



For this data the red line (mean of 100 classifiers) is smoother, more stable and more resistant for overtraining the any of the single classifiers.



Analogy: mean of many poorly fitting functions gives a good fitting function.

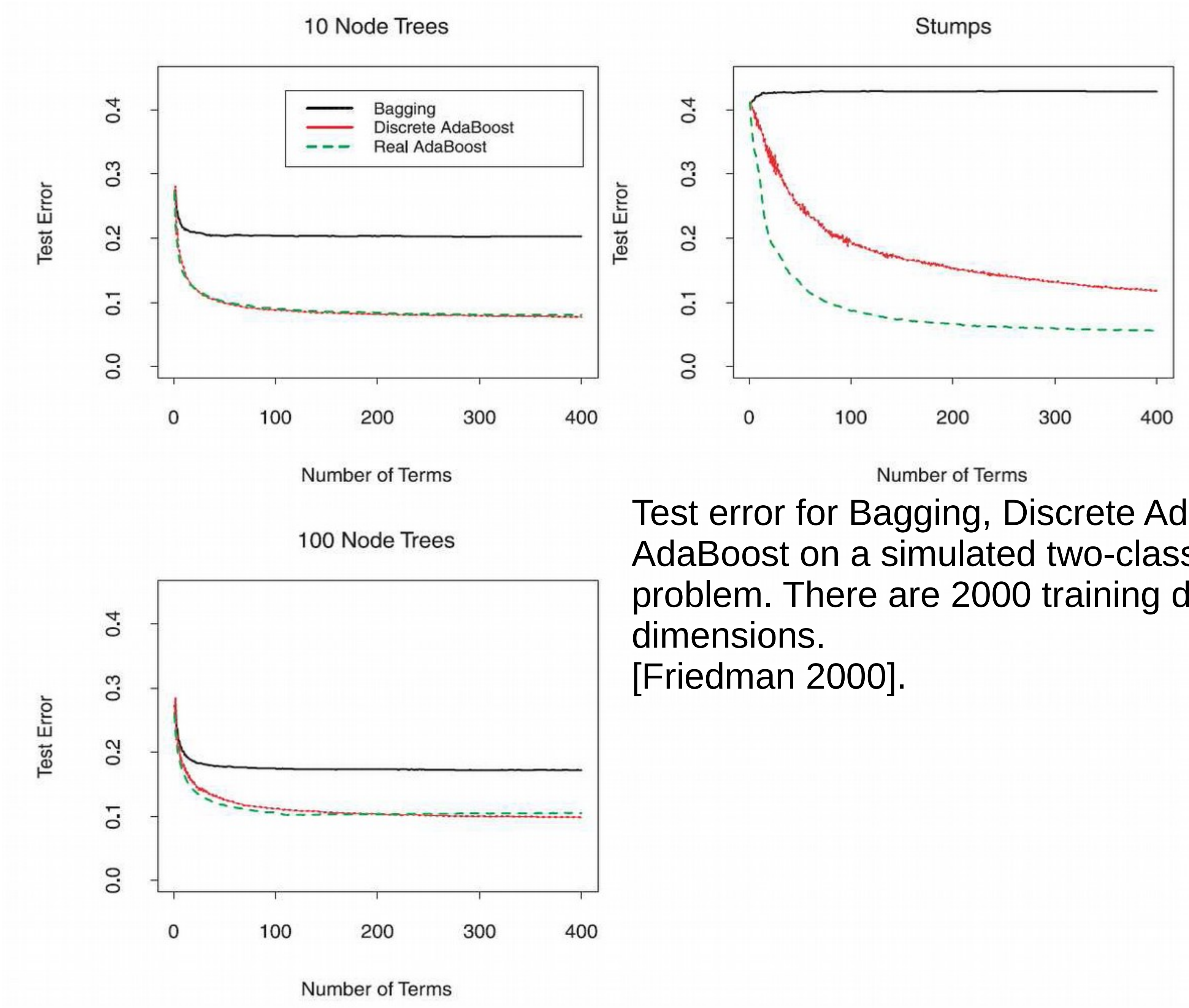
Bagging

- **Random Forest** – further development of Bagging:
  - K times pick N elements from the N element sample.
  - For each sample train a decision tree.
  - Building the tree in every decision point use only m randomly chosen variables

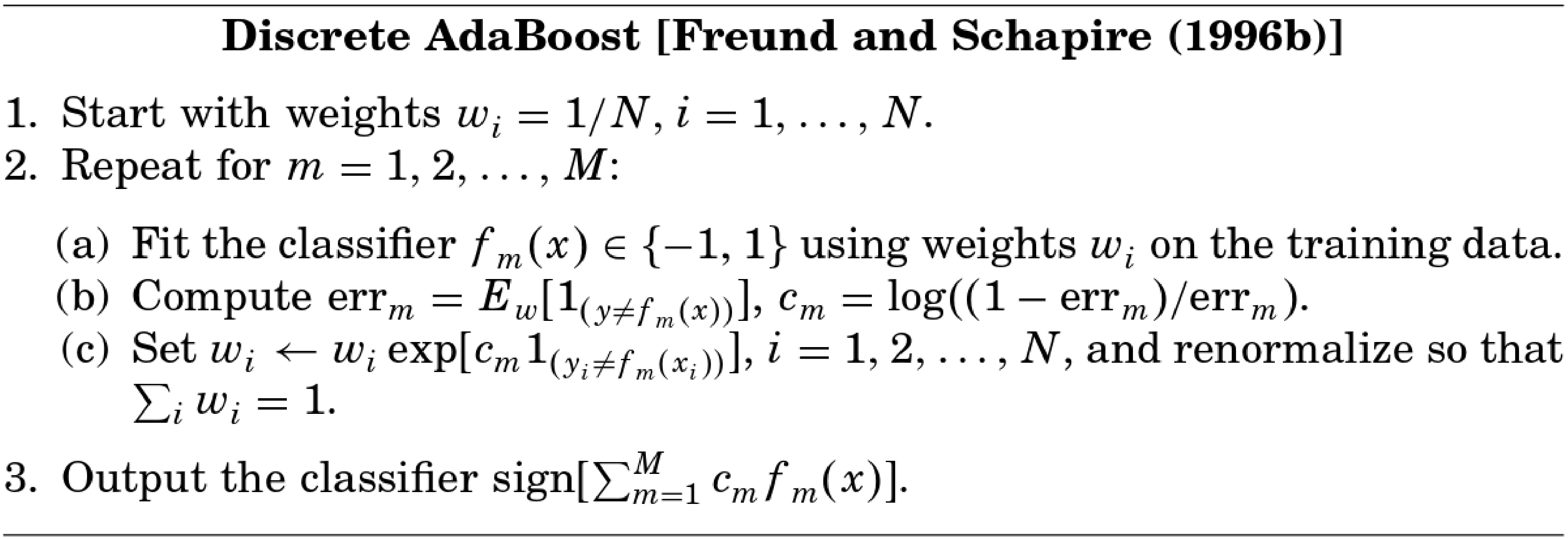
- Typically boosting gives better results, but bagging has a tendency to be more stable at high noise or when there are many “outliers”.
- Bagging is a parallel algorithm, boosting a sequential one.

Bauer and Kohavi, “An empirical comparison of voting classification algorithms”, Machine Learning 36 (1999)

Bagging vs. boosting

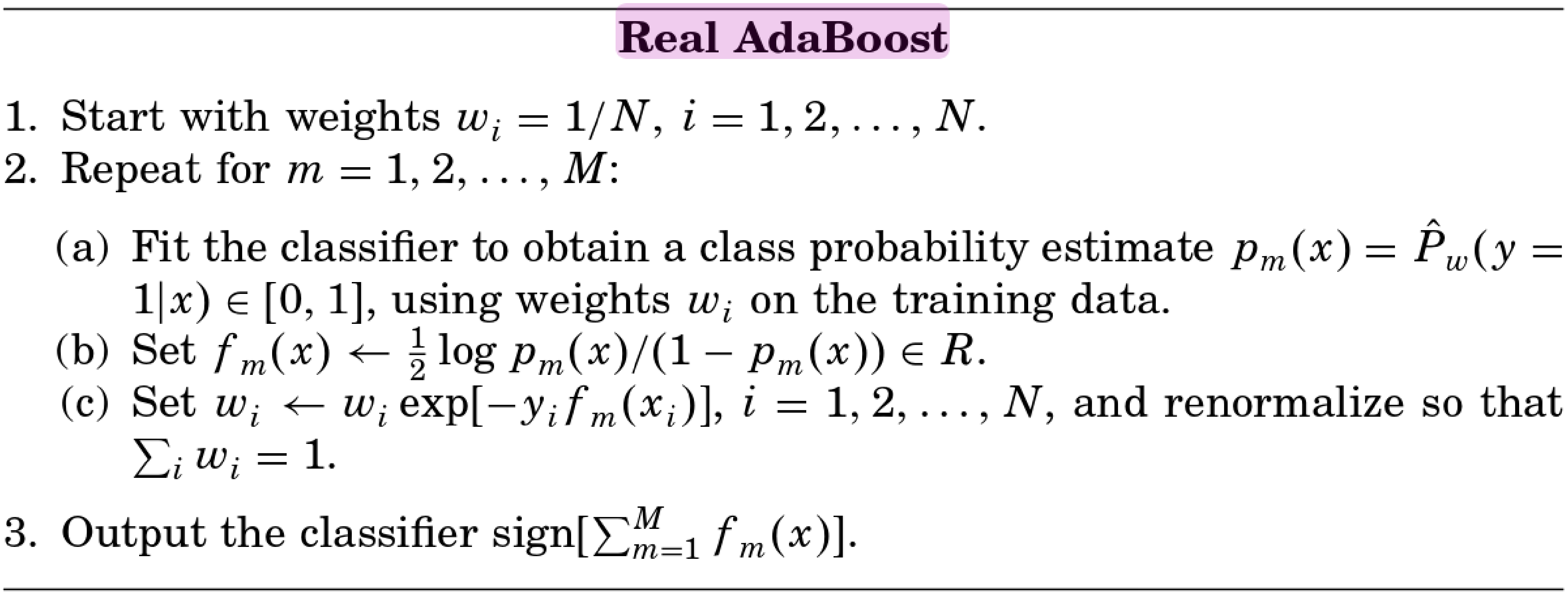


Test error for Bagging, Discrete AdaBoost and Real AdaBoost on a simulated two-class nested spheres problem. There are 2000 training data points in ten dimensions.
[Friedman 2000].



Discrete AdaBoost [Freund and Schapire (1996b)]

```
1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that
       $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.
```

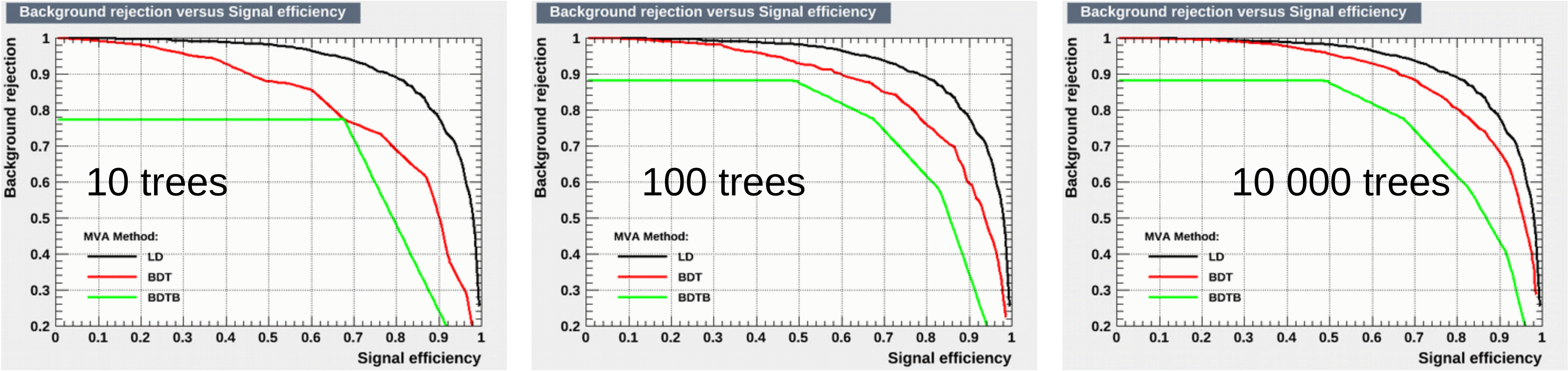


Real AdaBoost

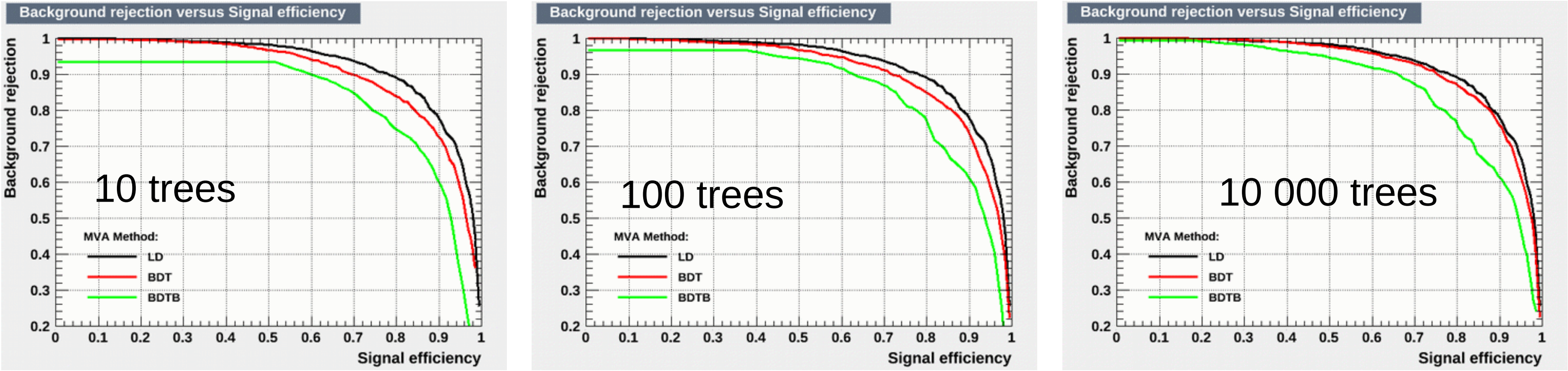
```
1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y =
       1|x) \in [0, 1]$, using weights $w_i$ on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \ldots, N$, and renormalize so that
       $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$.
```

Exercise using TMVA

Boosting and Bagging – separations of two Gaussians in 4 dimensions (linear Fisher Classifier is always the best).

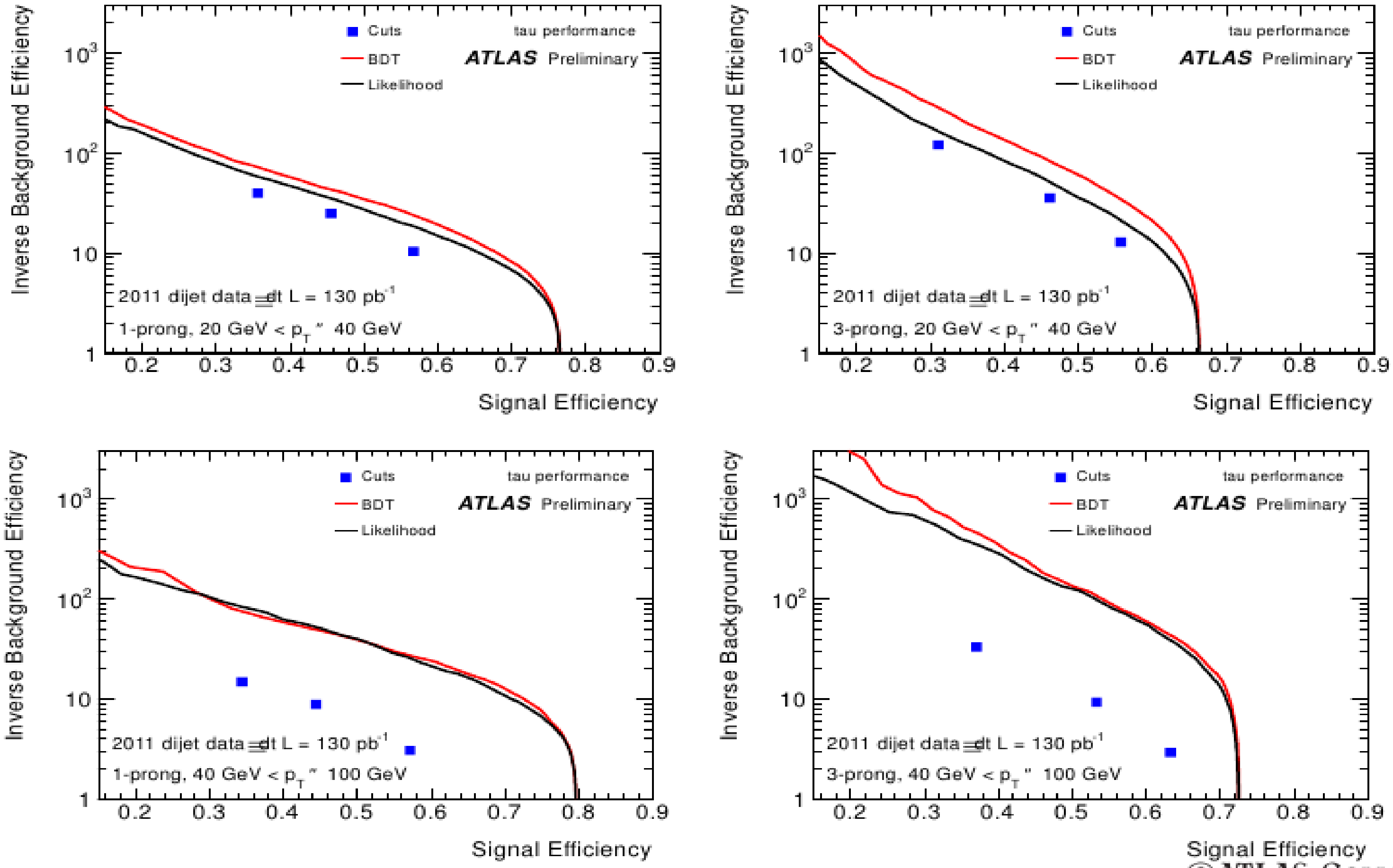


STUMPS (trees with a depth 1)



Trees with a depth 10

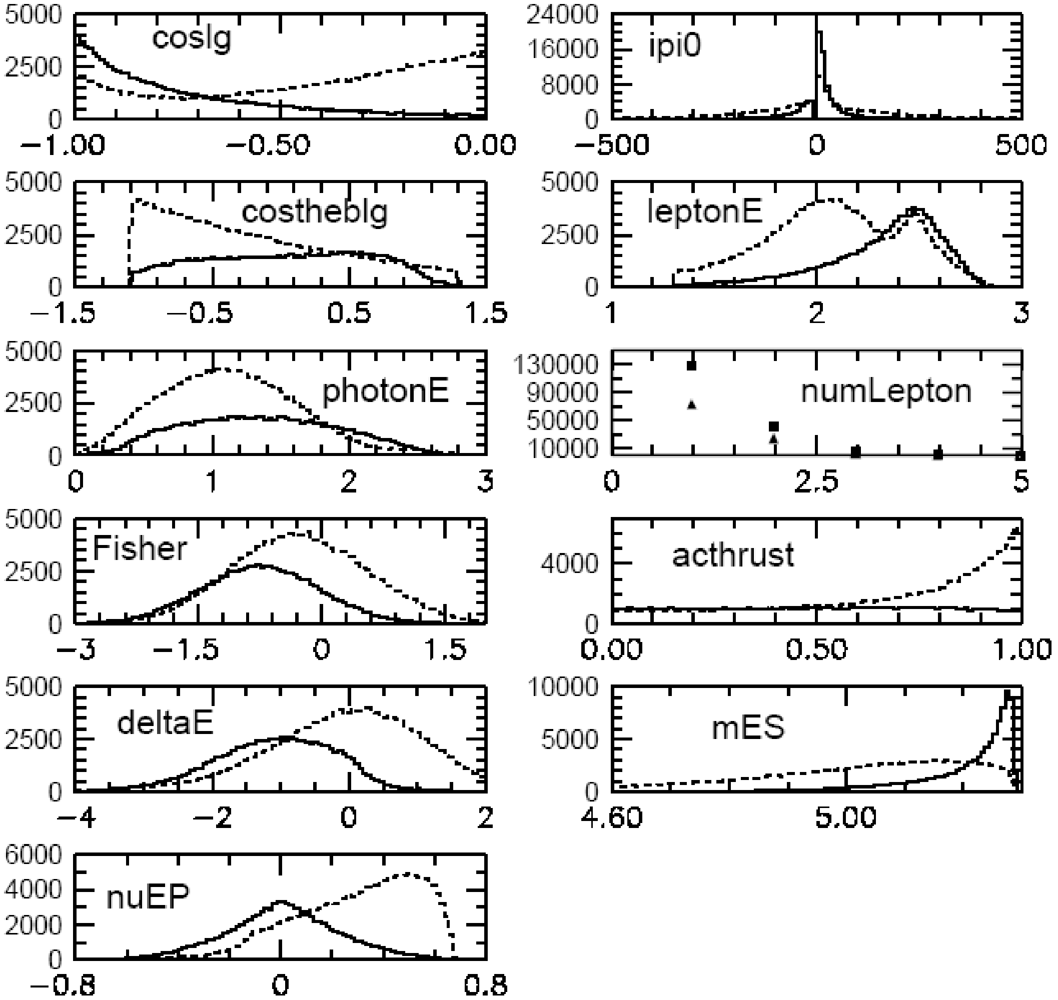
Again tau lepton identification in ATLAS



©ATLAS COLLAB.

- **BDT better than Naive Bayes and cuts**

Searches of decays $B \to \gamma l \nu$ in BaBar experiment



11-D data: 10 continuous and 1 discrete variable.

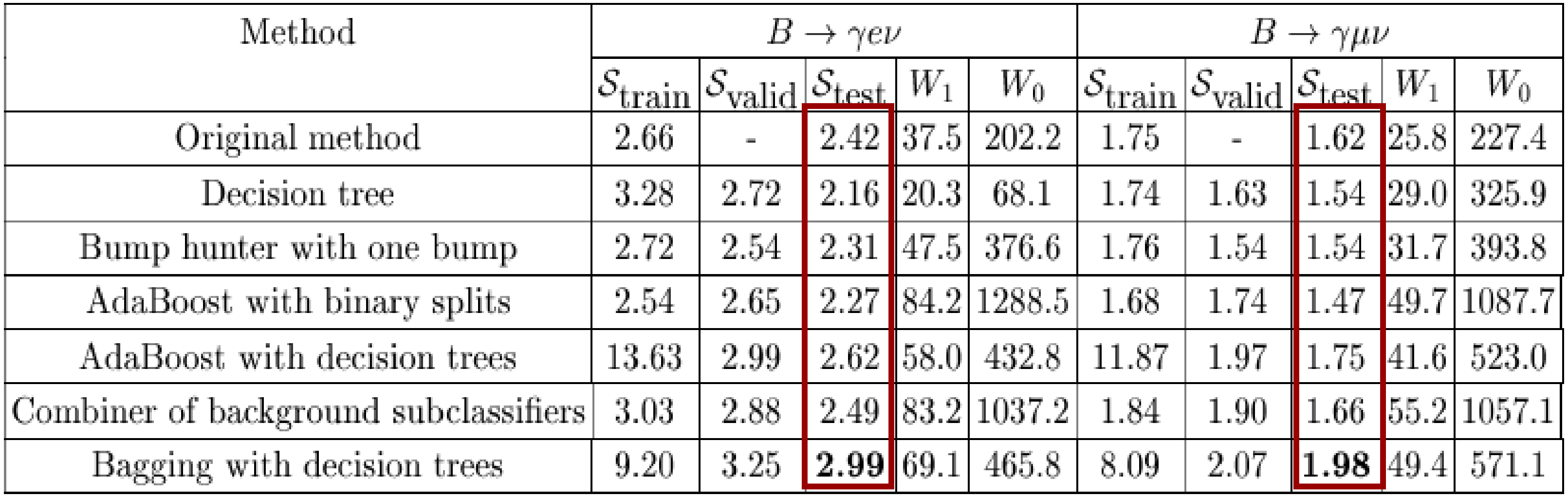
Some variables are strongly correlated: ρ(Fisher,acthrust)=0.8 and ρ(costheblg,nuEP)=-0.87.

Monte Carlo:

- training	= 500K
- validation	= 250K
- test	= 250K

Results

Signal significance obtained by different methods:



Method	$B \to \gamma e \nu$					$B \to \gamma \mu \nu$				
	S_{train}	S_{valid}	S_{test}	W_1	W_0	S_{train}	S_{valid}	S_{test}	W_1	W_0
Original method	2.66	-	2.42	37.5	202.2	1.75	-	1.62	25.8	227.4
Decision tree	3.28	2.72	2.16	20.3	68.1	1.74	1.63	1.54	29.0	325.9
Bump hunter with one bump	2.72	2.54	2.31	47.5	376.6	1.76	1.54	1.54	31.7	393.8
AdaBoost with binary splits	2.54	2.65	2.27	84.2	1288.5	1.68	1.74	1.47	49.7	1087.7
AdaBoost with decision trees	13.63	2.99	2.62	58.0	432.8	11.87	1.97	1.75	41.6	523.0
Combiner of background subclassifiers	3.03	2.88	2.49	83.2	1037.2	1.84	1.90	1.66	55.2	1057.1
Bagging with decision trees	9.20	3.25	**2.99**	69.1	465.8	8.09	2.07	**1.98**	49.4	571.1

Training of 50 „boosted decision trees” or 100 „bagged decision trees” took few hours.

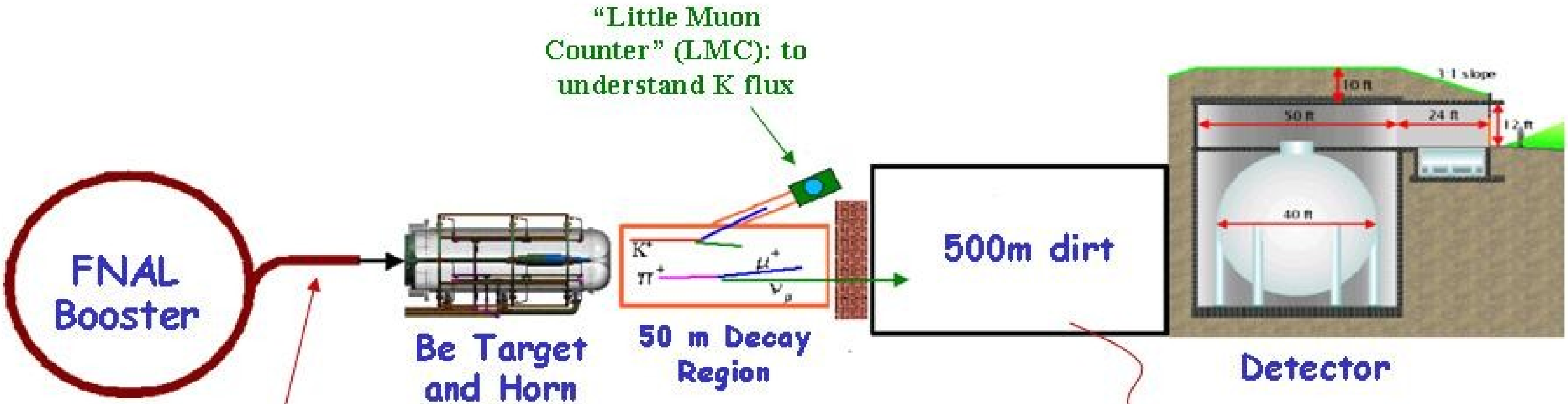
W_1 i W_0 are the expected numbers of signal and background events at luminosity 210 fb^{-1} .

Cleanest signal „bagged decision trees”.

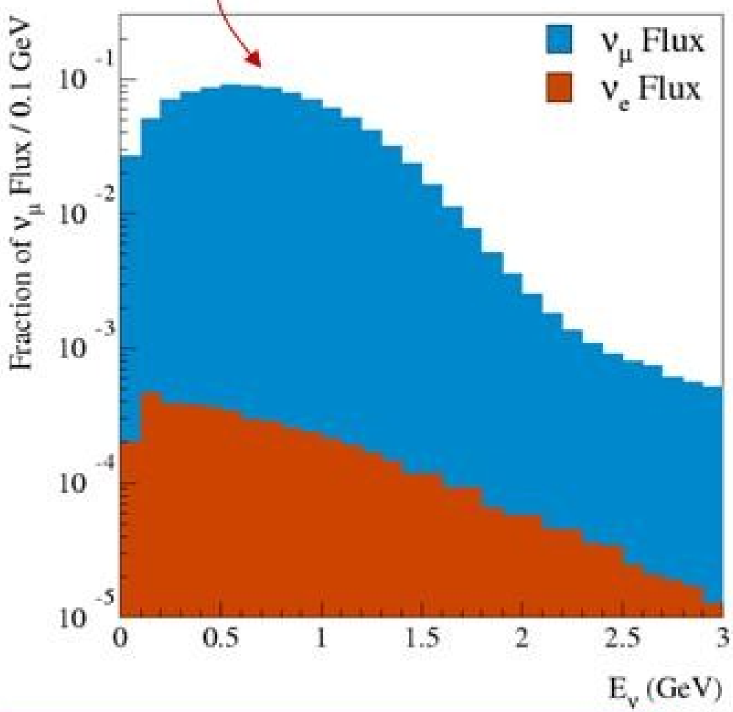
„Bagged decision trees” improve the signal purity by 14% comparing to „boosted decision trees”:

- **8% due to „bagging” instead of „boosting”.**
- **9% due to optimalization S/√(S+B) instead of Gini.**

Experiment Mini-BooNE – neutrino beam



8 GeV protons



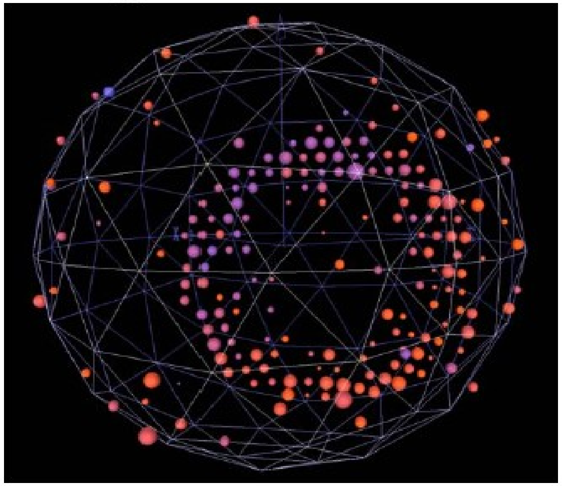
- Proton flux ~ 6E16 p/hr (goal 9E16 p/hr)
    - ~ 1 detected neutrino/minute
    - L/E ~ 1

Example events

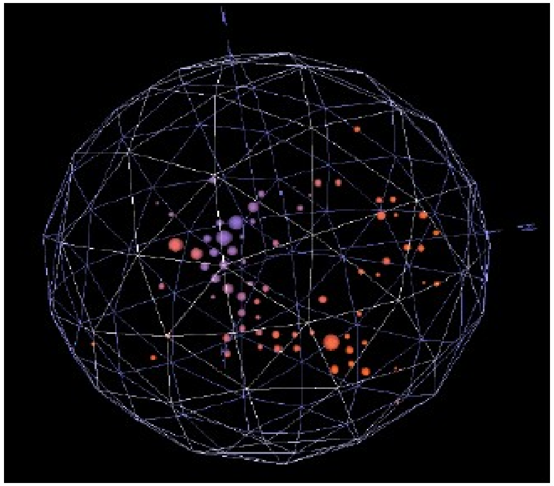
Pattern of hit tubes (with charge and time information)
allows reconstruction of track location and direction
and separation of different event types.

e.g. candidate events:

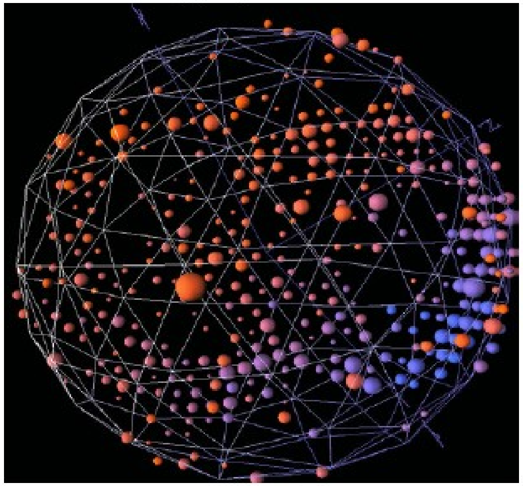
size = charge, color = time



muon
from ν_μ interaction

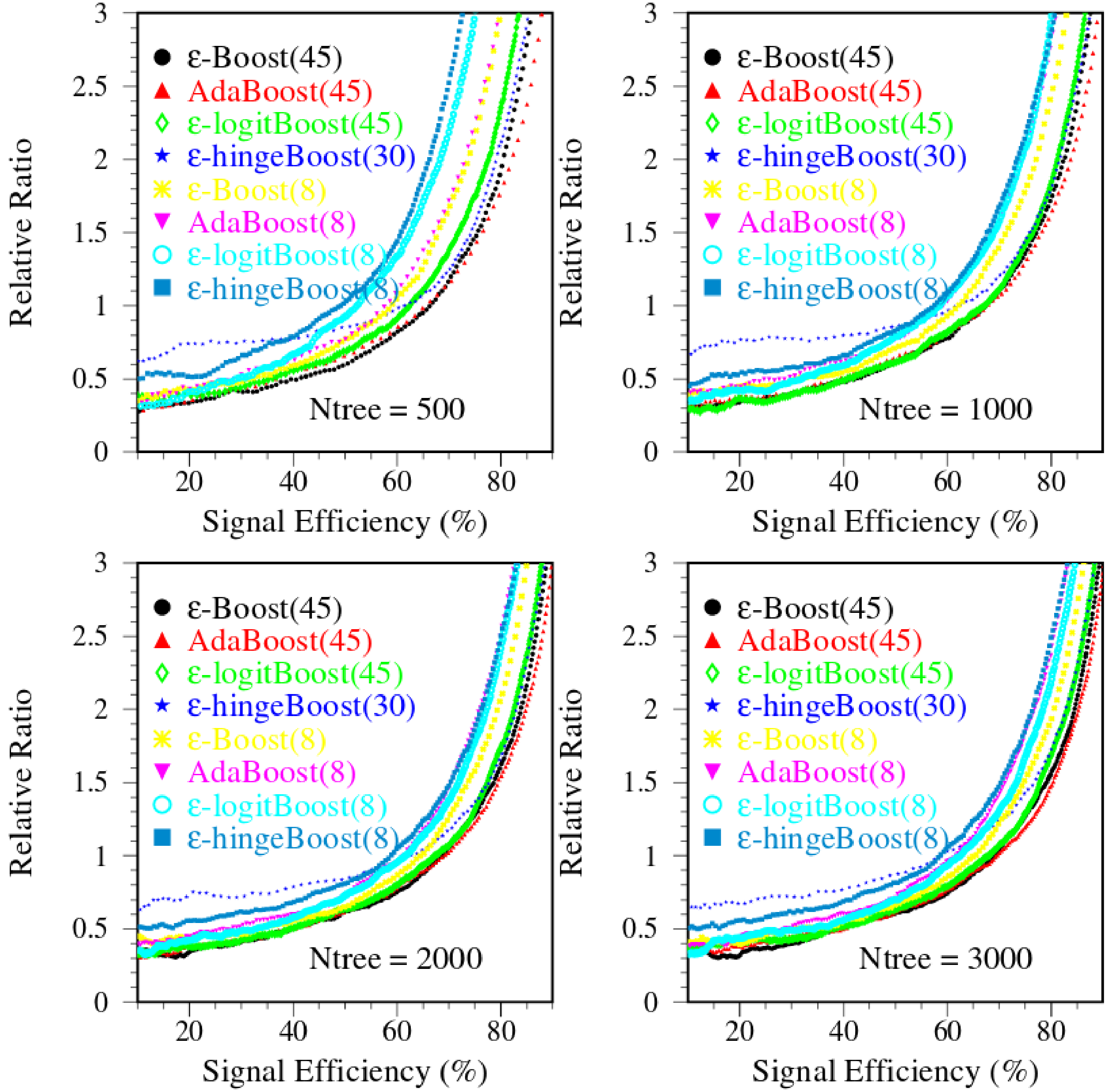


Michel electron
from stopped μ decay
after ν_μ interaction



$\pi^0 \rightarrow$ two photons
from ν_μ interaction

Comparison of various boosting algorithms



Authors found AdaBoost to have the best performance for Mini-BooNE experiment.

It might be different for other experiments :)

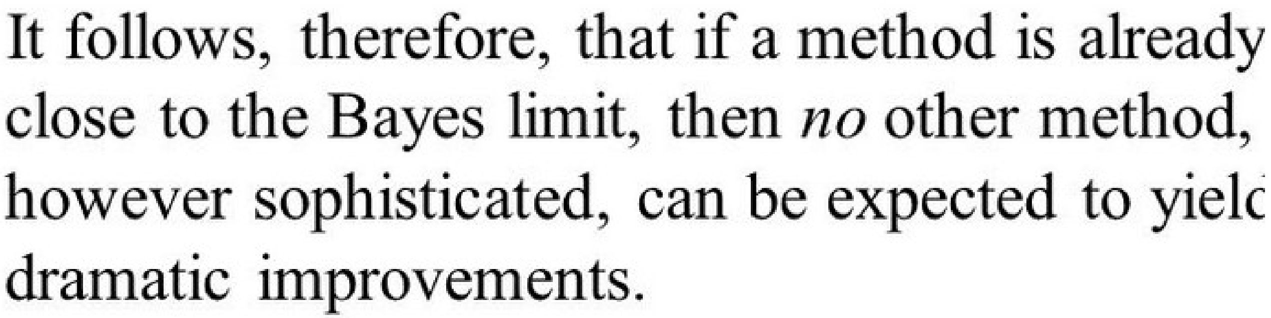
arXiv:physics/0508045

Summary

- Boosting, bagging – in a magic way we can build a strong classifier out of weak ones.
- Commonly used for decision trees, because they are simple and fast.
- Gives good results:

  - „the best out-of-box classification algorithm”.
- Very popular in high energy physics
- Or maybe just fancy?...

Should we use BDT only:



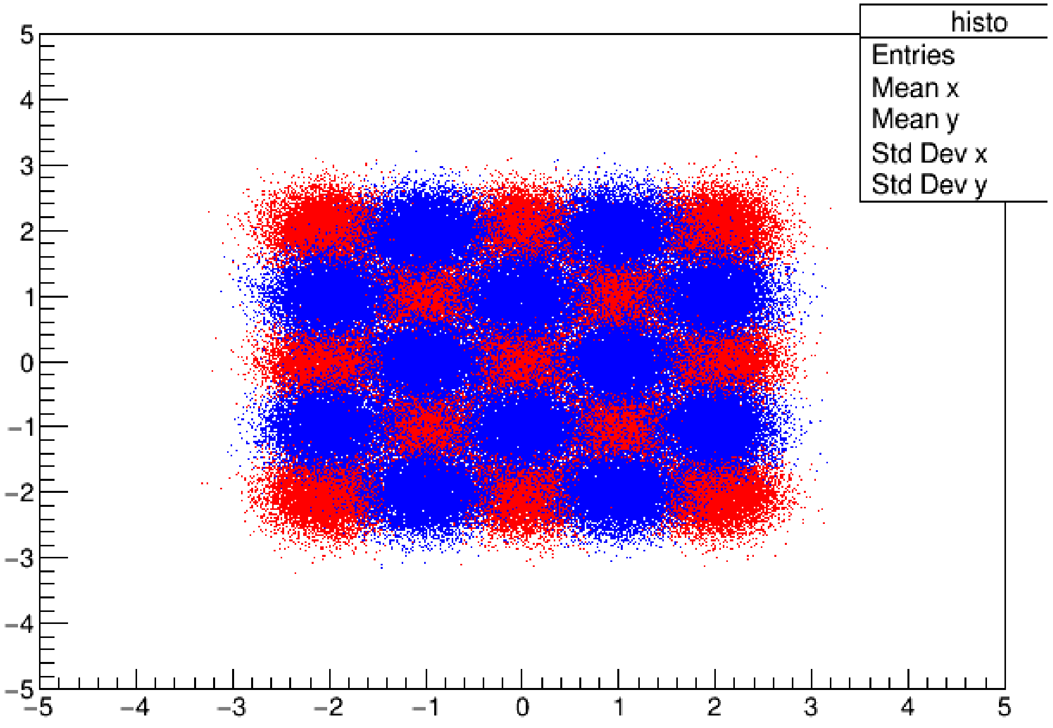
It follows, therefore, that if a method is already close to the Bayes limit, then *no* other method, however sophisticated, can be expected to yield dramatic improvements.

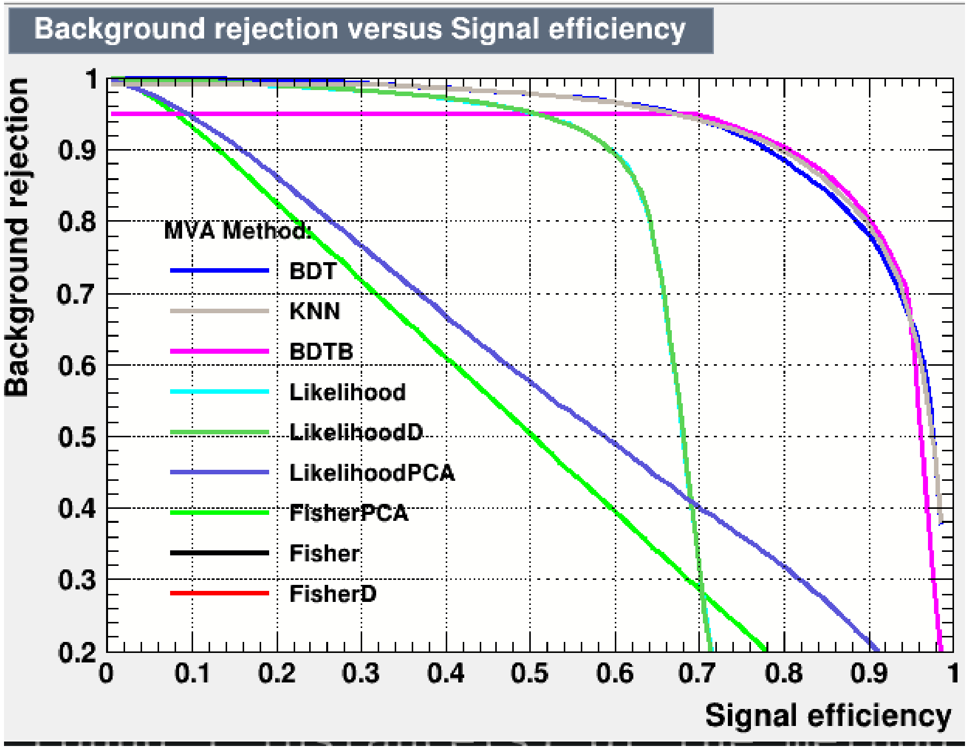
Harrison B. Prosper

Exercise: apply BDT to the "chessboard" data

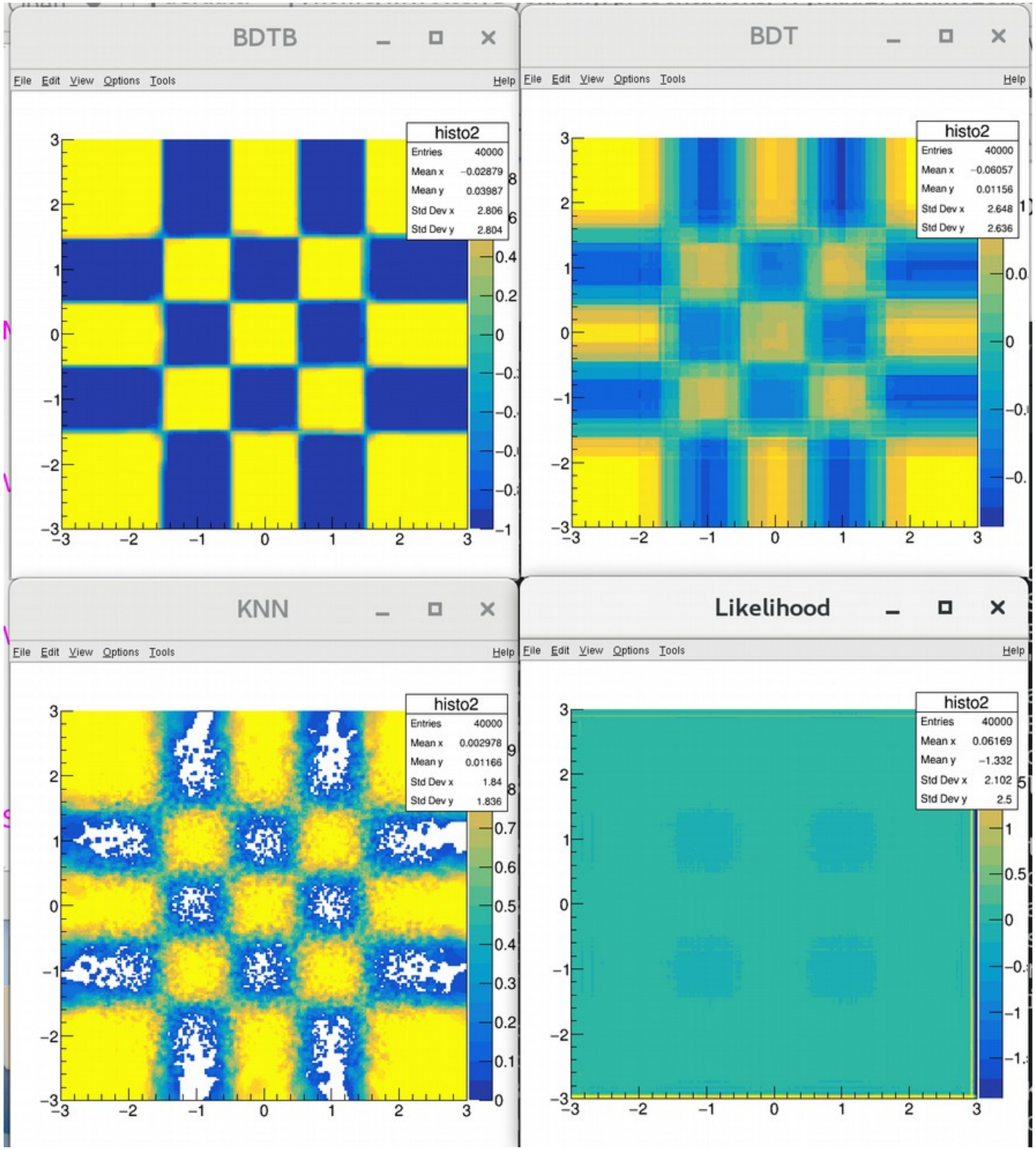
- Get code from: https://indico.ifj.edu.pl/event/232/ exercise_2.tar
- Create ntuple – python TMVAcreateNtuple.py
- Apply: Likelihood (Naive Bayes), kNN, BDT with boosting and bagging
  - python TMVAClassification.py
- Plot ROC etc: root -l

  TMVA::TMVAGui()
- Plot output maps (python TMVAreader.py)





Chessboard analysis


BDTB
File Edit View Options Tools Help
BDT
File Edit View Options Tools Help
KNN
File Edit View Options Tools Help
Likelihood
File Edit View Options Tools Help
3
2
1
0
−1
−2
−3
−3 −2 −1 0 1 2 3
0.4
0.2
0
−1
0.0
0
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
0.5
0
−1

BDTB

histo2	
Entries	40000
Mean x	−0.02879
Mean y	0.03987
Std Dev x	2.806
Std Dev y	2.804

BDT

histo2	
Entries	40000
Mean x	−0.06057
Mean y	0.01156
Std Dev x	2.648
Std Dev y	2.636

KNN

histo2	
Entries	40000
Mean x	0.002978
Mean y	0.01166
Std Dev x	1.84
Std Dev y	1.836

Likelihood

histo2	
Entries	40000
Mean x	0.06169
Mean y	−1.332
Std Dev x	2.102
Std Dev y	2.5