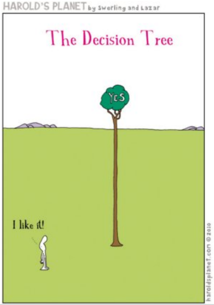
Classification using Boosted Decision Trees

KSETA Tutorial

Fabio Colombo, Raphael Friese, Manuel Kambeitz | 16-18 October, 2013

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)



Overview

- Manuel:
  - Introduction to multivariate analysis
  - Application in physics
- Fabio:
  - Decision trees
  - Training, boosting, overtraining
- Raphael:
  - TMVA: Toolkit for Multivariate Data Analysis with ROOT
  - Hands-on session

Feel free to ask questions during the talk.

What do we mean by classification?

Imagine car insurance company

- Wants to make individual insurance policy for each insured person
- Needs to know probability for an accident
- Has knowledge about each insured person:

ID	age	make	garage	km/year	claim 2013	claim 2014
1	20	Ford	yes	15 000	no	?
2	50	Opel	no	10 000	yes	?
3	38	Seat	no	25 000	no	?
4	35	Porsche	no	15 000	yes	?
5	70	Porsche	yes	7 000	no	?
...						
10 000	25	VW	yes	10 000	no	?

Technical terms

ID	age	make	garage	km/year	claim 2013	claim 2014
1	20	Ford	yes	15 000	no	?
2	50	Opel	no	10 000	yes	?
3	38	Seat	no	25 000	no	?
4	35	Porsche	no	15 000	yes	?
5	70	Porsche	yes	7 000	no	?
...						
10 000	25	VW	yes	10 000	no	?

Technical terms

- Columns in table (except ”claims 2014”) = ”Features” or ”Variables”
- Lines in table = ”Observations”
- Column ”claims 2014” = ”Target”
- Observations with Target equal ”yes” = ”Signal”
- Observations with Target equal ”no” = ”Background”

Technical terms

ID	age	make	garage	km/year	claim 2013	claim 2014
1	20	Ford	yes	15 000	no	?
2	50	Opel	no	10 000	yes	?
3	38	Seat	no	25 000	no	?
4	35	Porsche	no	15 000	yes	?
5	70	Porsche	yes	7 000	no	?
...						
10 000	25	VW	yes	10 000	no	?

Multivariate classification

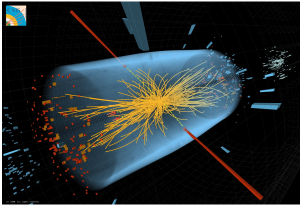
- Find out from features if target will be Signal or Background
- Mathematically seen: Function $\mathbb{R}^n \to \mathbb{R}$

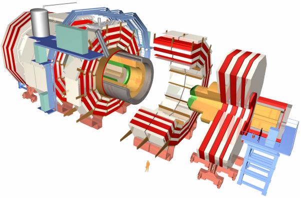
Multivariate classification

- Training needs data where Target is known in advance (labeled data)
- In this example, one could use ”claim 2013” as a label

Colliders and detectors

1. Particle collision
2. Produces intermediate particles
3. Decay to final state particles:
   $\pi, K, p^+, e^-, \mu^-, \gamma, invisible$
4. Signals in detector components
5. Read out electronically
6. From these data, final state particles are *reconstructed*
7. We know about each final state particle (with uncertainty): Momentum, energy, trajectory, particle identification info, ...





This typically happens before an **analysis** at EKP starts.

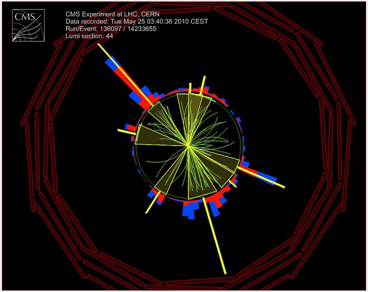
Data analysis

Example for HEP analysis at CMS: Reaction $H \rightarrow \tau\tau \rightarrow \mu\mu(4\nu)$

Looking at the final state particles in each particle collision and find out if the reaction happened in this collision *event*.

Difficulties:

- Very rare processes
- Very few of the muons observed by CMS come from the analyzed reaction
- Most muons originate from a different process
- We do not know which events contain signal or background



Data analysis

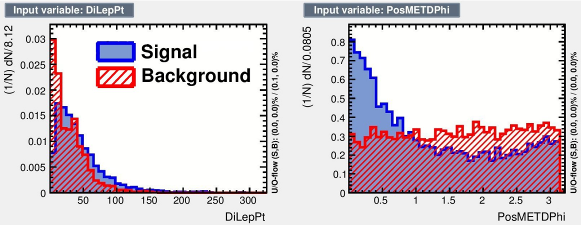
- We form di-muon combinations
- We need to classify di-muon combinations to either signal or background
- We have to do this very effectively
- We use suitable features
- Features are calculated for each combination

- Describe muons with
  - transverse momentum
  - particle identification
  - *impact parameter*
- Describe di-muon combinations with
  - invariant mass
  - angles

Why this works

Example $H \rightarrow \tau\tau \rightarrow \mu\mu(4\nu)$ decays (only one sort of background):

- Transverse momentum of di-muon system is higher for signal than for background
- Angle between μ^+ and $E^T_{\rm miss}$ is random for background and small for signal



- Due to physics laws, signal and background can ”look” different
- Values of features (or correlations) are different for signal and background
- So we have the possibility to distinguish between signal and background

How we know how signal looks like

A multivariate classifier can only be trained using labeled data

Typical approach: (*Monte Carlo*) Simulations

Typical Ingredients:

- Simulation of the production of the investigated state
- Simulation of the decay of the investigated state
- Other reactions happening at the same time
- Detector simulation
- Running the same reconstruction and analysis as for data

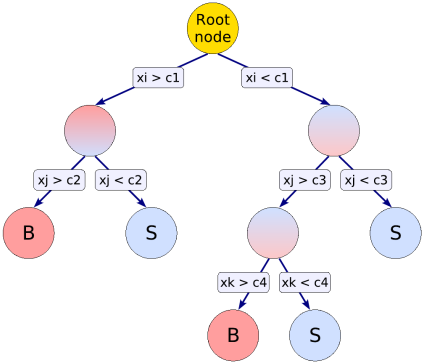
Also for background, often simulations are used.

Typical problem:

Simulations describe data not perfectly

Boosted Decision Trees (BDTs)

simple example of **single** binary decision tree



- simplest example of classifier
- **supervised learning**: training phase on events with known signal/background composition is needed
- repeated yes/no decisions taken at each node on one single variable (the most discriminating one!) per time, until a "stop" criterion is reached
- the event phase space is divided in many regions (hypercubes, or leaves) classified as signal or background-like
- some events in each leaf are "misclassified": background events in signal leaf or signal events in background leaf

Splitting criteria

- many criteria are in principle possible: a common one is the **Gini index**

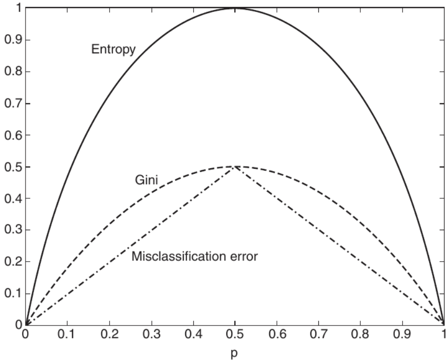
$$p = \frac{S}{S+B} \quad \text{“purity”} \quad (0 \leq p \leq 1) \qquad \Longrightarrow \qquad \mathrm{G} = p(1-p)$$

- can be seen as a measure of the degree of impurity of the node:

  “all signal node” $\Rightarrow \mathrm{G} = 0$
  “all background node” $\Rightarrow \mathrm{G} = 0$
  “perfect mixing” $\Rightarrow \mathrm{G}$ is maximum

- find the variable (and the best cut) that maximizes the difference in Gini index between parent node and the left/right daughter leaves

$$\Delta = \mathrm{G}_P - \mathrm{G}_L - \mathrm{G}_R$$



Single tree: advantages and disadvantages

Karlsruhe Institute of Technology

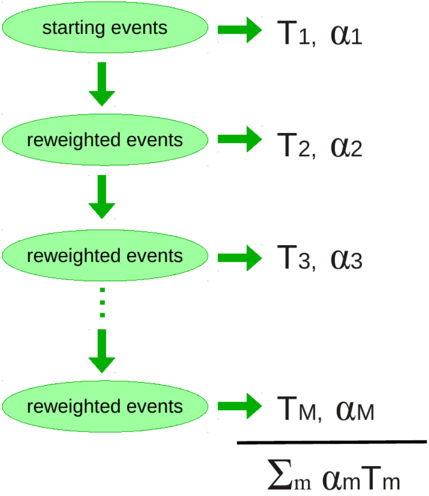
- very straightforward interpretation (almost as simple as a sequence of rectangular cuts)
- insensitive to the addition of poorly discriminating variables (although very sensitive when adding a single - but strongly discriminant! - one)
- no event removed completely: no information loss

- very unstable!! small statistical fluctuations in the training sample might change the variable on which the splitting happens $\Rightarrow$ structure of the tree altered below that particular node

“Boost” the tree, creating many of them (a “forest”)

- stability increased
- performance enhanced

Boosting: general concept



- take the misclassified events on final leaves and **give to them an higher weight**
- renormalize everything: the sum of weights must remain constant
- build another tree with the reweighted events
- give a certain **score** α to the tree
- repeat many times ($\sim$1000)

- **take the average** of all trees, using the tree score α as weights
- the final result represents the output of the BDT

The Adaptive Boosting (AdaBoost) algorithm

Event type: $\mathrm{Y}_i = \begin{cases} +1 & \text{signal} \\ -1 & \text{background} \end{cases}$

Tree response: $\mathrm{T}_i = \pm 1$

the misclassified events are those for which $\mathrm{T}_i \neq \mathrm{Y}_i$

Define, for the tree $\Rightarrow$ $\mathrm{err}_m = \sum_{\mathrm{T}_i \neq \mathrm{Y}_i} w_i \implies \alpha_m = \beta \log\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right)$

Increase the weight for misclassified events: $w_i \rightarrow w_i \cdot \exp(\alpha_m)$

Renormalize the weights (sum of weights remains constant)

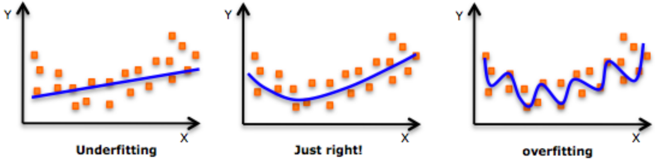
After having optimized and scored M trees on a training sample with known composition, we can feed the real events to the "forest"

$$\mathrm{T}_i(\vec{x}) = \sum_{m=1}^{M} \alpha_m \cdot \mathrm{T}_m(\vec{x})$$

binary response (± 1) for signal/background leaf

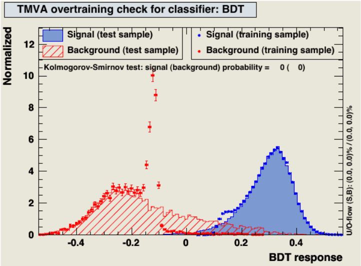
The problem of overtraining

- a classifier can “over-adapt” itself to the training sample, and show a very high efficiency, but on an independent dataset the results would be very different...



- an overtraining would show itself in an inconsistency between the output obtained on the **training sample** and the one obtained on an independent **test sample**

- a common way to avoid overtraining is to remove statistically insignificant nodes, and/or playing with the stopping parameter or the β parameter of the AdaBoost algorithm



Hands-on session: Boosted Decision Trees in TMVA

TMVA - Toolkit for Multivariate Analysis

- Recent Version 4.2.0 is included in ROOT release 5.34/11
- More than a dozen different implementations of multivariate methods
- Easy-to-use with access to all training parameters

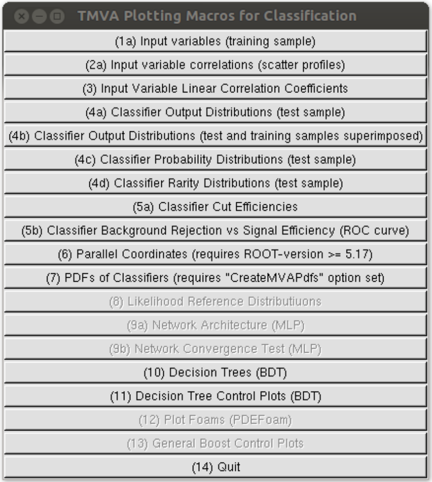
Procedure

- split the available labled data in two parts:
  - training sample
  - testing sample
- perform the training
- evaluate the performance

Hands-on session: Three exercises for the first use of TMVA.

Exercise 1: Introduction on basic functionality

- login with name **user** / pw **user**
- Open a terminal (Ctrl + Alt + T)
- run
  sudo mount /dev/sda3 kseta/
  cd kseta
- start the macro with
  ./root/bin/root -l MyTMVA.C
- open the source code with
  gedit MyTMVA.C &



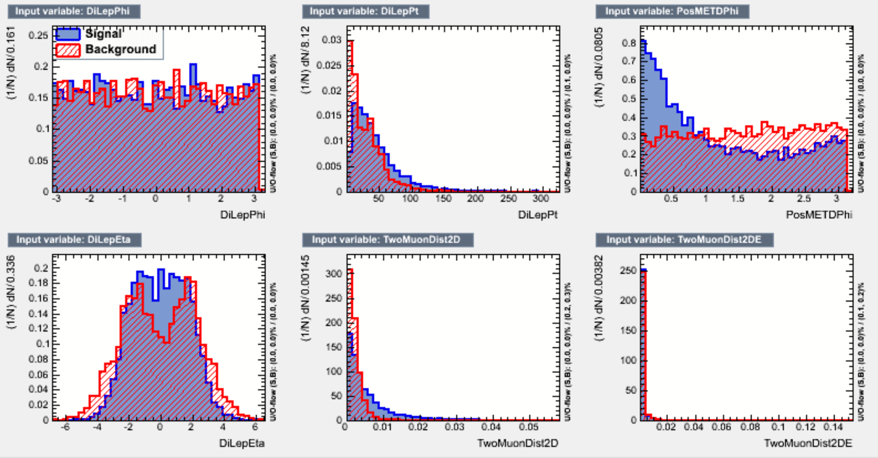
The TMVA Gui

The TMVA Gui is designed to get a quick overview on the training results

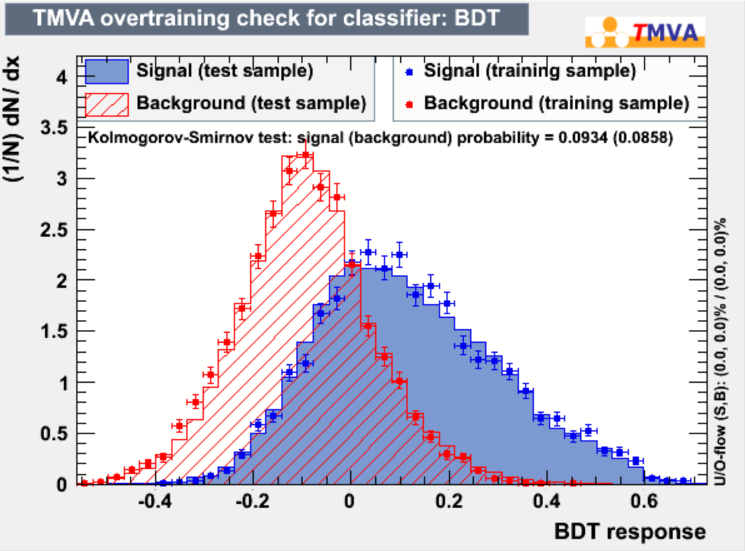
1a) Input Variables - Control Plots

Karlsruhe Institute of Technology

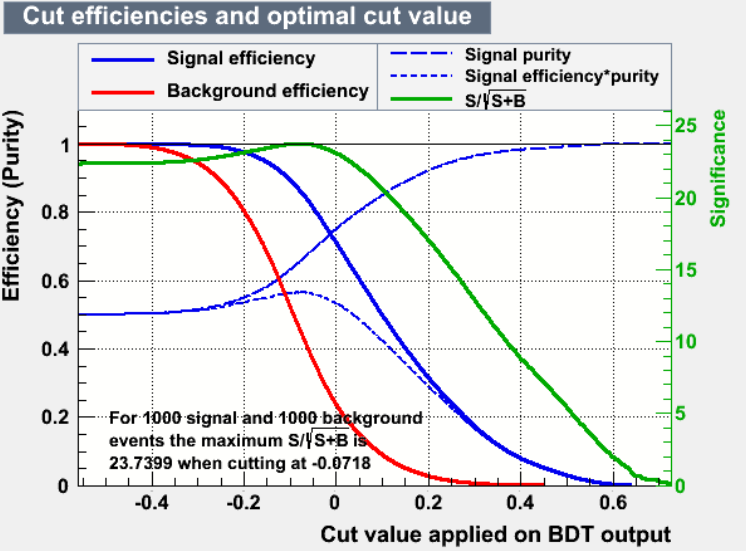
Normalized distributions of input variables to judge their shape



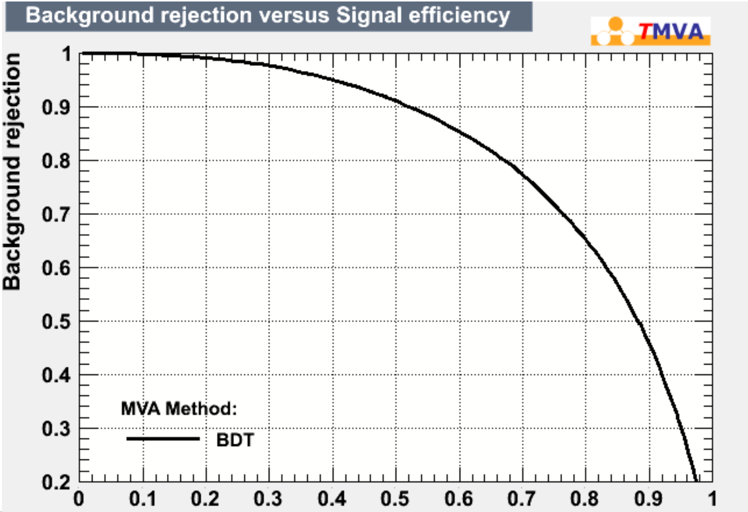
4b) Classifier Outputs superimposed - Control Plots



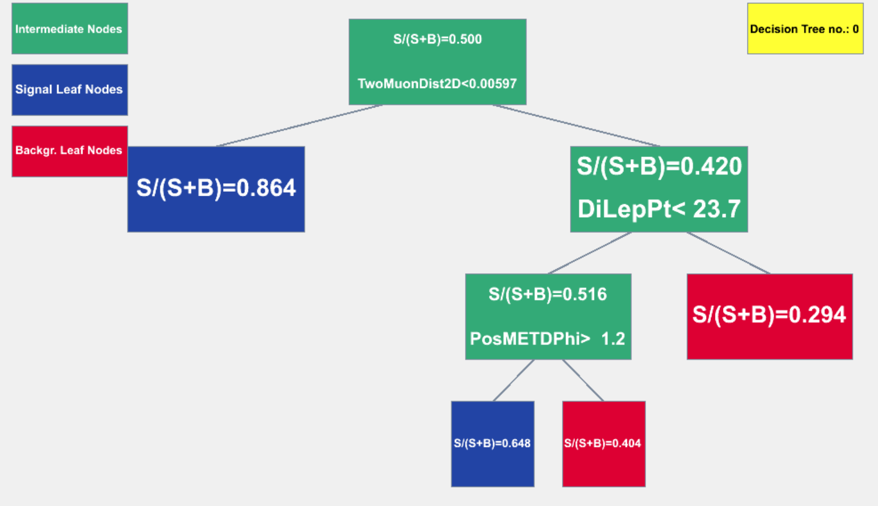
5a) Classifier Cut Efficiencies - Control Plots



5b) Classifier Background Rejection vs. Signal Efficiency (ROC curve) - Control Plots



10) Decision Trees (BDT) - Control Plots



Exercise 2: From a little tree to a huge forest

- Open the TMVA Options Reference: tmva.sourceforge.net/optionRef.html
- Search in for the parameter that sets the number of trees in the forest
- open MyTMVA.C and look for line 73
- Vary parameters and see the changes in the training in the performance parameters explained.
  - Suggested numbers of trees: 1, 2, 100 (default), 1000
- Set the number of trees to a reasonable level (e.g. 100) and vary the stopping parameter for the maximum depth
  - Suggested maximum depth: 1, 3 (default), 20

Checks

Check after each training the control Plots (4b), (5a) and eventually click through (10). Compare the influences of the parameters in terms of separation power and Kolmogorov-Smirnov test.

Exercise 3: Handling small training samples

Small training samples

Small training samples are very prone to statistical fluctuations.

- Set the number of signal- and background training events each to 2000 (line 69). Keep the number of testing events.
- Perform a training. What do you observe?
- Try to get a reasonable training result by the variation of the parameters
  - NTrees
  - MaxDepth
  - AdaBoostBeta
- Which parameters lead to the highest ROC integral while still having the KS-Test above 5%?

Discussion and Feedback

A few points for the discussion...

- What can BDTs and multivariate methods in general be used for?
- What has to be checked when introducing MVAs?

Feedback ...

- Did you get the introduction?
- Do you have the feeling that you know, how BDTs work?
- Did we answer your questions in a satisfying way?
- Was the hands-on part usefull?
- Was the amount of theoretical introduction ok?

Backup Slides