

Boosted decision trees

Introduction to machine learning tools

Yann Coadou

CPPM Marseille

SOS2026

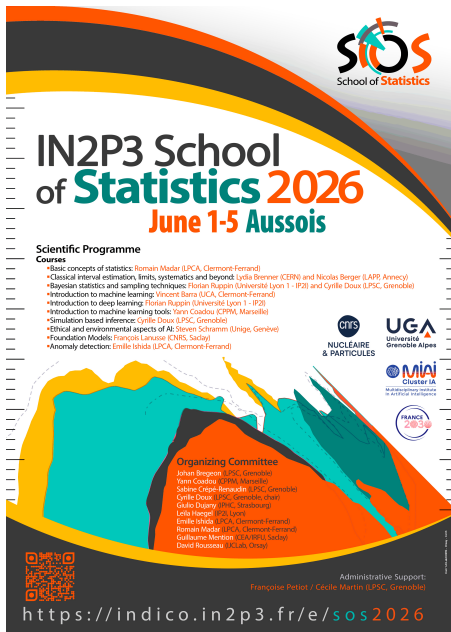
Aussois, 3 June 2026




NUCLÉAIRE
& PARTICULES







School of **Statistics**




IN2P3 School of Statistics 2026
June 1-5 Aussois

Scientific Programme Courses

- Basic concepts of statistics: Romain Madar (LPCA, Clermont-Ferrand)
- Classical interval estimation, limits, systematics and beyond: Lyda Brenner (CERN) and Nicolas Berger (LAPP, Annecy)
- Bayesian statistics and sampling techniques: Florian Ruggin (Université Lyon 1 - IP2I) and Cyrille Doux (LPSG, Grenoble)
- Introduction to machine learning: Vincent Barra (UCA, Clermont-Ferrand)
- Introduction to deep learning: Florian Ruggin (Université Lyon 1 - IP2I)
- Introduction to machine learning tools: Yann Coadou (CPPM, Marseille)
- Simulation based inference: Cyrille Doux (LPSG, Grenoble)
- Ethical and environmental aspects of AI: Steven Schramm (Unige, Genève)
- Foundation Models: François Larusse (CNRS, Saclay)
- Anomaly detection: Emille Ishida (LPCA, Clermont-Ferrand)

 **NUCLEAIRE & PARTICULES**
 **Université Grenoble Alpes**
 **MIA Cluster IA**
Multi-disciplinary Institute on Artificial Intelligence
 **FRANCE 2030**

Organizing Committee

Johan Bregoux (LPSG, Grenoble)
 Yann Coadou (CPPM, Marseille)
 Sabine Crepe-Renouin (LPSG, Grenoble)
 Cyrille Doux (LPSG, Grenoble, chair)
 Guilo Dupuy (IPHC, Strasbourg)
 Léila Huegel (IP2I, Lyon)
 Emille Ishida (LPCA, Clermont-Ferrand)
 Romain Madar (LPCA, Clermont-Ferrand)
 Guillaume Menton (CEA-IRFU, Saclay)
 David Rousseau (ICLabs, Orsay)

Administrative Support:
 Françoise Petiot / Cécile Martin (LPSG, Grenoble)

<https://indico.in2p3.fr/e/sos2026>

- 1 Decision trees
- 2 Limitations
- 3 Boosted decision trees
- 4 Software
- 5 Conclusion
- 6 References

▶ [Jupyter notebook](#)

Decision tree origin

- Machine-learning technique, originally used in data mining/pattern recognition, then medical diagnosis, insurance/loan screening, etc.



L. Breiman *et al.*, “Classification and Regression Trees” (1984)

Basic principle

- Extend cut-based selection
 - many (most?) events do not have *all* characteristics of signal or background
 - try not to rule out events failing a particular criterion
- Keep events rejected by one criterion and see whether other criteria could help classify them properly

Binary trees

- Trees can be built with branches splitting into many sub-branches
- In this lecture: mostly binary trees

Start with all events (signal and background) = first (root) node

- sort all events by each variable
- for each variable, find splitting value with best separation between two children
 - mostly signal in one child
 - mostly background in the other
- select variable and splitting value with best separation, produce two branches (nodes)
 - events failing criterion on one side
 - events passing it on the other

Keep splitting

- Now have two new nodes. Repeat algorithm recursively on each node
- Can reuse the same variable
- Iterate until stopping criterion reached
- Splitting stops: terminal node = leaf

- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T



- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T
 - sort all events by each variable:
 - $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$
 - $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
 - $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$



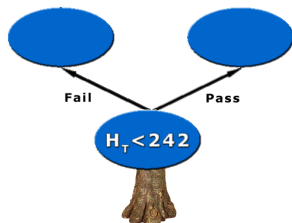
- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T
 - sort all events by each variable:
 - $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$
 - $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
 - $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$
 - best split (arbitrary unit):
 - $p_T < 56$ GeV, separation = 3
 - $H_T < 242$ GeV, separation = 5
 - $M_t < 105$ GeV, separation = 0.7



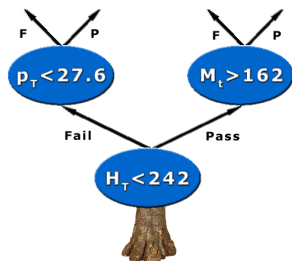
- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T
 - sort all events by each variable:
 - $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$
 - $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
 - $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$
 - best split (arbitrary unit):
 - $p_T < 56$ GeV, separation = 3
 - $H_T < 242$ GeV, separation = 5
 - $M_t < 105$ GeV, separation = 0.7



- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T
 - sort all events by each variable:
 - $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$
 - $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
 - $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$
 - best split (arbitrary unit):
 - $p_T < 56$ GeV, separation = 3
 - $H_T < 242$ GeV, separation = 5
 - $M_t < 105$ GeV, separation = 0.7
 - split events in two branches: pass or fail $H_T < 242$ GeV



- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T
 - sort all events by each variable:
 - $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$
 - $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
 - $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$
 - best split (arbitrary unit):
 - $p_T < 56$ GeV, separation = 3
 - $H_T < 242$ GeV, separation = 5
 - $M_t < 105$ GeV, separation = 0.7
 - split events in two branches: pass or fail $H_T < 242$ GeV
- Repeat recursively on each node



- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T 's of all objects in the event H_T

- sort all events by each variable:

- $p_T^{s_1} \leq p_T^{b_{34}} \leq \dots \leq p_T^{b_2} \leq p_T^{s_{12}}$

- $H_T^{b_5} \leq H_T^{b_3} \leq \dots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$

- $M_t^{b_6} \leq M_t^{s_8} \leq \dots \leq M_t^{s_{12}} \leq M_t^{b_9}$

- best split (arbitrary unit):

- $p_T < 56$ GeV, separation = 3

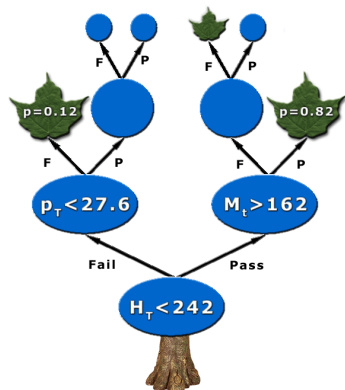
- $H_T < 242$ GeV, separation = 5

- $M_t < 105$ GeV, separation = 0.7

- split events in two branches: pass or fail $H_T < 242$ GeV

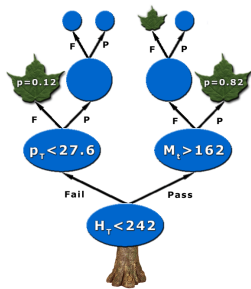
- Repeat recursively on each node

- Splitting stops: e.g. events with $H_T < 242$ GeV and $M_t > 162$ GeV are signal like ($p = 0.82$)



Run event through tree

- Start from root node
- Apply first best cut
- Go to left or right child node
- Apply best cut for this node
- ...Keep going until...
- Event ends up in leaf



DT Output

- Purity ($\frac{s}{s+b}$, with weighted events) of leaf, close to 1 for signal and 0 for background
- or binary answer (discriminant function +1 for signal, -1 or 0 for background) based on purity above/below specified value (e.g. $\frac{1}{2}$) in leaf
- E.g. events with $H_T < 242$ GeV and $M_t > 162$ GeV have a DT output of 0.82 or +1

Normalization of signal and background before training

- Balanced classes: same total weight for signal and background events ($p = 0.5$, maximal mixing)

Selection of splits

- list of questions ($variable_i < cut_i?$, “Is jet b -tagged?”)
- goodness of split (separation measure)

Decision to stop splitting (declare a node terminal)

- minimum leaf size (for statistical significance, e.g. 100 events)
- insufficient improvement from further splitting
- perfect classification (all events in leaf belong to same class)
- maximal tree depth (like-size trees choice or computing concerns)

Assignment of terminal node to a class

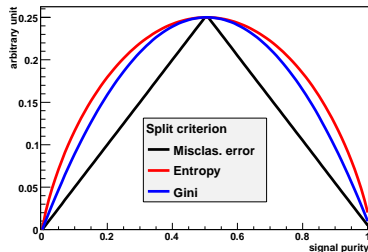
- signal leaf if purity > 0.5 , background otherwise

Optimal split: figure of merit

- Decrease of impurity for split s of node t into children t_P and t_F (goodness of split): $\Delta i(s, t) = i(t) - p_P \cdot i(t_P) - p_F \cdot i(t_F)$
- Aim: find split s^* such that $\Delta i(s^*, t) = \max_{s \in \{\text{splits}\}} \Delta i(s, t)$
- Maximising $\Delta i(s, t) \equiv$ minimising overall tree impurity

Common impurity functions

- misclassification error
 $= 1 - \max(p, 1 - p)$
- (cross) entropy
 $= -\sum_{i=s,b} p_i \log p_i$
- Gini index



- Also cross section significance $(-\frac{s^2}{s+b})$ and excess significance $(-\frac{s^2}{b})$

Reminder

- Need model giving good description of data

Reminder

- Need model giving good description of data

Playing with variables

- Number of variables:
 - not affected too much by “curse of dimensionality”
 - CPU consumption scales as $nN \log N$ with n variables and N training events
- Variable order does not matter: all variables treated equal
- Order of training events irrelevant (batch training)
- Irrelevant variables:
 - no discriminative power \Rightarrow not used
 - only cost: extra CPU time, no added noise
- Can use continuous and discrete variables, simultaneously

Transforming input variables

- Completely insensitive to replacement of any subset of input variables by (possibly different) arbitrary strictly monotone functions of them (same order \Rightarrow same DT):
 - convert MeV \rightarrow GeV
 - no need to make all variables fit in the same range
 - no need to regularise variables (e.g. taking the log)

\Rightarrow Some immunity against outliers

Transforming input variables

- Completely insensitive to replacement of any subset of input variables by (possibly different) arbitrary strictly monotone functions of them (same order \Rightarrow same DT):
 - convert MeV \rightarrow GeV
 - no need to make all variables fit in the same range
 - no need to regularise variables (e.g. taking the log)

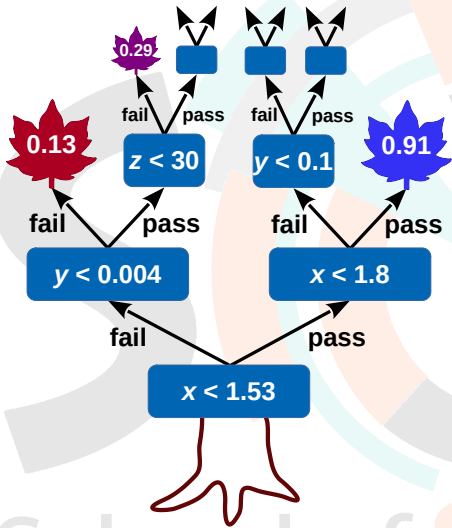
\Rightarrow Some immunity against outliers

Note about actual implementation

- Strictly true only if testing all possible cut values
- May not work anymore with computational optimisation (e.g., check only 20 possible cuts on each variable)

Choosing variables

- Usually try to have as few variables as possible
- But difficult: correlations, possibly large number to consider, large phase space with different properties in different regions
- **Brute force:** with n variables train all n , $n - 1$, etc. combinations, pick best (first *define* “best”)
- **Backward elimination:** train with n variables, then train all $n - 1$ variables trees and pick best one; now train all $n - 2$ variables trees starting from the $n - 1$ variable list; etc. Pick optimal cost-complexity tree.
- **Forward greedy selection:** start with $k = 1$ variable, then train all $k + 1$ variables trees and pick the best; move to $k + 2$ variables; etc.



2 Limitations

- Training sample composition
- Pruning a tree
- Ensemble learning

- Small changes in sample can lead to very different tree structures (high variance)
- Not optimal to understand data from DT rules
- Does not give confidence in result:
 - DT output distribution discrete by nature
 - granularity related to tree complexity
 - tendency to have spikes at certain purity values (or just two delta functions at ± 1 if not using purity)

Why prune a tree?

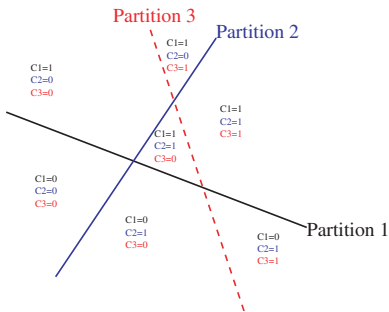
- Possible to get a perfect classifier on training events
- Mathematically misclassification error can be made as little as wanted
- E.g. tree with one class only per leaf (down to 1 event per leaf if necessary)
- Training error is zero
- But run new independent events through tree (testing or validation sample): misclassification probably > 0 , overtraining
- Pruning: eliminate subtrees (branches) that seem too specific to training sample:
 - node and all its descendants turned into single leaf

Pruning algorithms

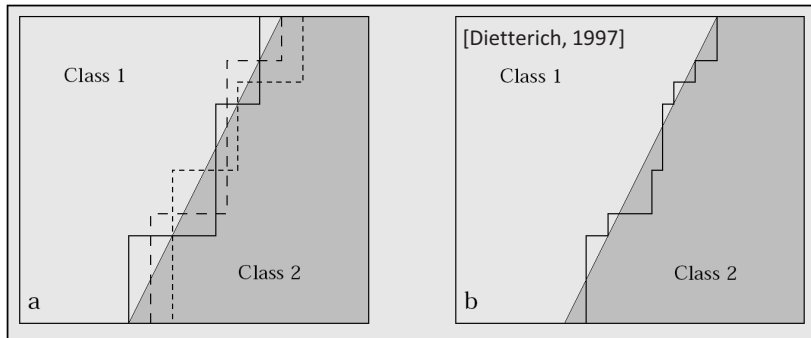
[▶ details in backup](#)

- Pre-pruning (early stopping condition like min leaf size, max depth)
- Expected error pruning (based on statistical error estimate)
- Cost-complexity pruning (penalise “complex” trees with many nodes/leaves)

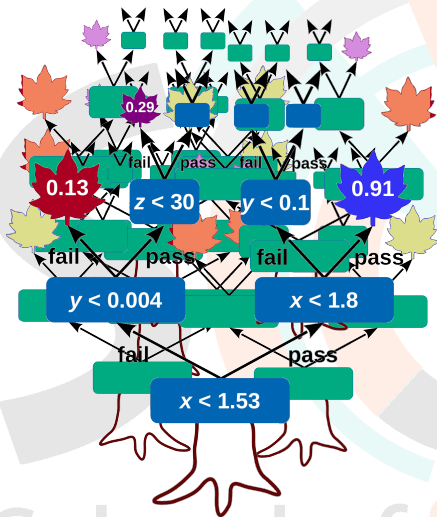
- One tree:
 - one information about event (one leaf)
 - cannot really generalise to variations not covered in training set (at most as many leaves as input size)
- Many trees:
 - **distributed representation**: number of intersections of leaves exponential in number of trees
 - many leaves contain the event \Rightarrow richer description of input pattern



- Build several trees and average outputs



- K-fold cross-validation (good for small samples)
 - divide training sample \mathcal{L} in K subsets of equal size: $\mathcal{L} = \bigcup_{k=1..K} \mathcal{L}_k$
 - Train tree T_k on $\mathcal{L} - \mathcal{L}_k$, test on \mathcal{L}_k
 - DT output = $\frac{1}{K} \sum_{k=1..K} T_k$
- Bagging, boosting, random forests: **ensemble learning**



3 Boosted decision trees

- Introduction
- AdaBoost
- Clues to boosting performance
- Gradient boosting
- Performance examples
- BDTs in real physics cases

First provable algorithm [Schapire 1990]

- Train classifier T_1 on N events
- Train T_2 on new N -sample, half of which misclassified by T_1
- Build T_3 on events where T_1 and T_2 disagree
- Boosted classifier: $\text{MajorityVote}(T_1, T_2, T_3)$

First provable algorithm [Schapire 1990]

- Train classifier T_1 on N events
- Train T_2 on new N -sample, half of which misclassified by T_1
- Build T_3 on events where T_1 and T_2 disagree
- Boosted classifier: MajorityVote(T_1, T_2, T_3)

Then

- Variation [Freund 1995]: boost by majority (combining many learners with fixed error rate)
- Freund&Schapire joined forces: 1st functional model **AdaBoost** (1996)

First provable algorithm [Schapire 1990]

- Train classifier T_1 on N events
- Train T_2 on new N -sample, half of which misclassified by T_1
- Build T_3 on events where T_1 and T_2 disagree
- Boosted classifier: MajorityVote(T_1, T_2, T_3)

Then

- Variation [Freund 1995]: boost by majority (combining many learners with fixed error rate)
- Freund&Schapire joined forces: 1st functional model **AdaBoost** (1996)

When it really picked up in HEP

- MiniBooNe compared performance of different boosting algorithms and neural networks for particle ID [MiniBooNe 2005]
- D0 claimed first evidence for single top quark production [D0 2006]
- CDF copied 😊 (2008). Both used BDT for single top observation

What is boosting?

- General method, not limited to decision trees
- Hard to make a very good learner, but easy to make simple, error-prone ones (but still better than random guessing)
- Goal: combine such weak classifiers into a new more stable one, with smaller error

AdaBoost

- Introduced by Freund&Schapire in 1996
- Stands for *adaptive boosting*
- Learning procedure adjusts to training data to classify it better
- Many variations for actual implementation
- Usually leads to better results than without boosting

- Check which events of training sample \mathbb{T}_k are misclassified by T_k :
 - $\mathbb{I}(X) = 1$ if X is true, 0 otherwise
 - for DT output in $\{\pm 1\}$: $\text{isMisclassified}_k(i) = \mathbb{I}(y_i \times T_k(x_i) \leq 0)$
 - or $\text{isMisclassified}_k(i) = \mathbb{I}(y_i \times (T_k(x_i) - 0.5) \leq 0)$ in purity convention
 - misclassification rate:

$$R(T_k) = \varepsilon_k = \frac{\sum_{i=1}^N w_i^k \times \text{isMisclassified}_k(i)}{\sum_{i=1}^N w_i^k}$$

- Derive tree weight $\alpha_k = \beta \times \ln((1 - \varepsilon_k)/\varepsilon_k)$
- Increase weight of misclassified events in \mathbb{T}_k to create \mathbb{T}_{k+1} :

$$\text{event } i \text{ well classified by } T_k : \quad w_i^k \rightarrow w_i^{k+1} = w_i^k$$

$$\text{event } i \text{ misclassified by } T_k : \quad w_i^k \rightarrow w_i^{k+1} = w_i^k \times e^{\alpha_k}$$

- Train T_{k+1} on \mathbb{T}_{k+1}
- Boosted result of event i :

$$T(i) = \frac{1}{\sum_{k=1}^{N_{\text{tree}}} \alpha_k} \sum_{k=1}^{N_{\text{tree}}} \alpha_k T_k(i)$$

Misclassification rate ε on training sample

- Can be shown to be bound:

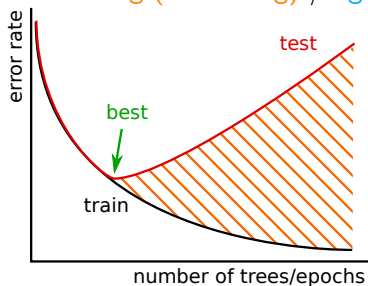
$$\varepsilon \leq \prod_{k=1}^{N_{tree}} 2\sqrt{\varepsilon_k(1 - \varepsilon_k)}$$

- If each tree has $\varepsilon_k \neq 0.5$ (i.e. better than random guessing):
the error rate falls to zero for sufficiently large N_{tree}
- Corollary: training data overfitted by construction

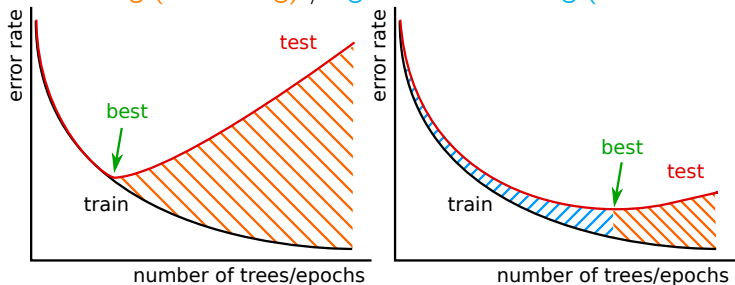
Overtraining?

- Error rate on test sample may reach a minimum and then potentially rise. Stop boosting at the minimum.
- In principle AdaBoost *must* overfit training sample
- In many cases in literature, no loss of performance due to overtraining

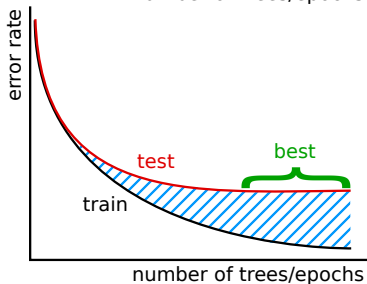
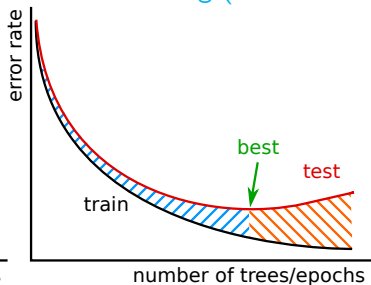
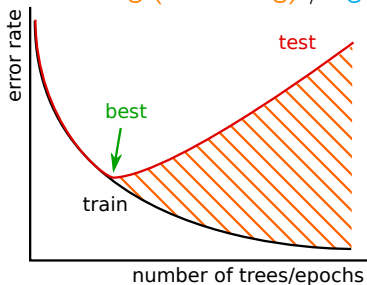
“bad” overtraining (overfitting) / “good” overtraining (still underfitting)



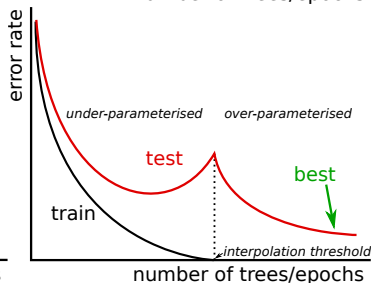
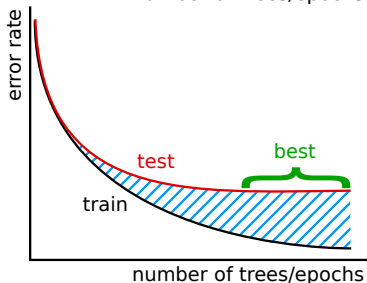
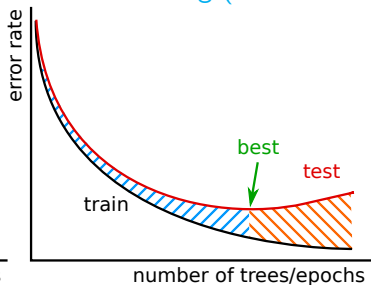
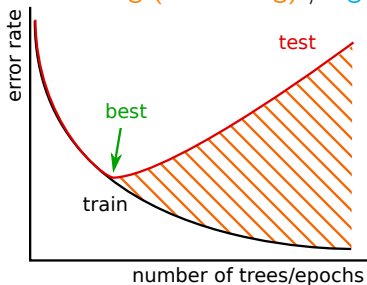
“bad” overtraining (overfitting) / “good” overtraining (still underfitting)



“bad” overtraining (overfitting) / “good” overtraining (still underfitting)



“bad” overtraining (overfitting) / “good” overtraining (still underfitting)



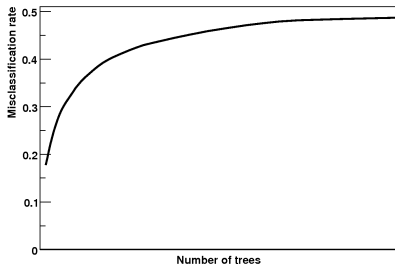
Overfitting implies high variance

- unstable model class
- variance **increases with model complexity**
- variance **decreases with more training data**

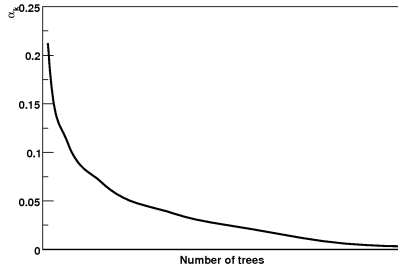
Underfitting implies high bias

- even with no variance, model class has high error
- happens whenever **model complexity is too low**

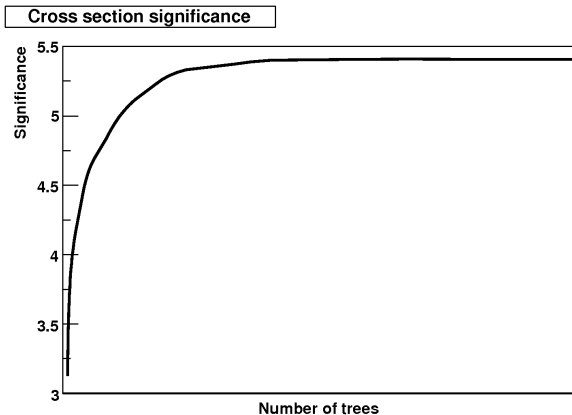
Misclassification rate for each tree



Tree weight α_k



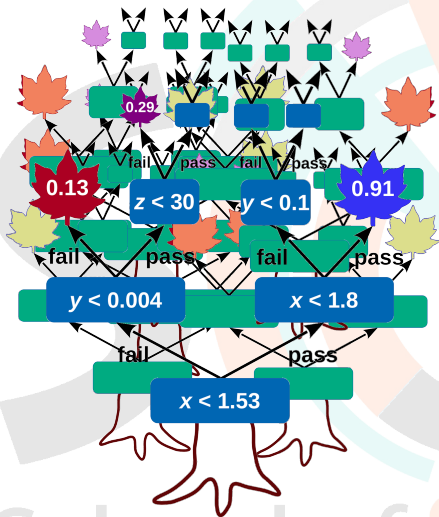
- First tree is best, others are minor corrections
- Specialised trees do not perform well on most events \Rightarrow decreasing tree weight and increasing misclassification rate
- Last tree is not better evolution of first tree, but rather a pretty bad DT that only does a good job on few cases that the other trees could not get right
- But adding trees may increase reliability of prediction: margins explanation [Shapire&Freund 2012]
- Double descent risk curve and interpolation regime [Belkin 2019]



- More relevant than testing error if trying to measure $s - b$ separation
- Reaches plateau (to be checked!)
- Afterwards, boosting does not hurt (just wasted CPU)
- Applicable to any other figure of merit of interest for your use case

- AdaBoost recast in a statistical framework: corresponds to minimising an exponential loss
- Generalisation: formulate boosting as numerical optimisation problem, minimise loss function by adding trees using gradient descent procedure
- Procedure:
 - Build imperfect model F_k at step k (sometimes $F_k(x) \neq y$)
 - Improve model: $F_{k+1}(x) = F_k(x) + h_k(x) = y$, or residual $h_k(x) = y - F_k(x)$
 - Train new classifier on residual
- Example: mean squared error loss function
$$L_{\text{MSE}}(x, y) = \frac{1}{2} (y - F_k(x))^2$$
 - minimising loss $J = \sum_i L_{\text{MSE}}(x_i, y_i)$ leads to $\frac{\partial J}{\partial F_k(x_i)} = F_k(x_i) - y_i$
 \Rightarrow residual as negative gradient: $h_k(x_i) = y_i - F_k(x_i) = -\frac{\partial J}{\partial F_k(x_i)}$
- Generalised to any differentiable loss function

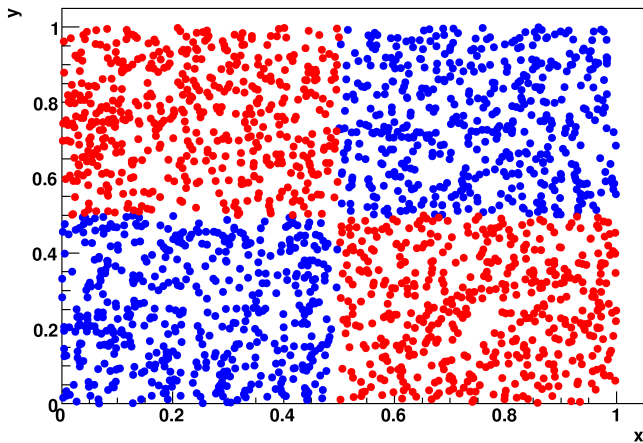
- AdaBoost recast in a statistical framework: corresponds to minimising an exponential loss
- Generalisation: formulate boosting as numerical optimisation problem, minimise loss function by adding trees using gradient descent procedure
- Procedure:
 - Build imperfect model F_k at step k (sometimes $F_k(x) \neq y$)
 - Improve model: $F_{k+1}(x) = F_k(x) + h_k(x) = y$, or residual $h_k(x) = y - F_k(x)$
 - Train new classifier on residual
- Example: mean squared error loss function
$$L_{\text{MSE}}(x, y) = \frac{1}{2} (y - F_k(x))^2$$
 - minimising loss $J = \sum_i L_{\text{MSE}}(x_i, y_i)$ leads to $\frac{\partial J}{\partial F_k(x_i)} = F_k(x_i) - y_i$
 \Rightarrow residual as negative gradient: $h_k(x_i) = y_i - F_k(x_i) = -\frac{\partial J}{\partial F_k(x_i)}$
- Generalised to any differentiable loss function
- Nowadays most used algorithm in BDT implementations [▶ see hands-on](#)

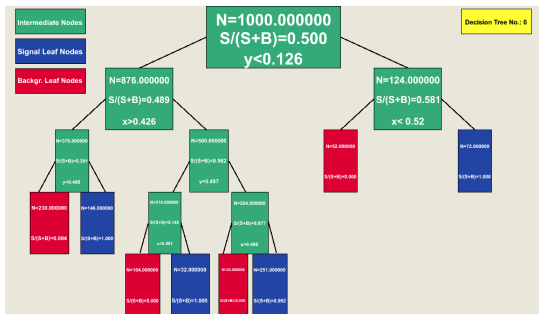
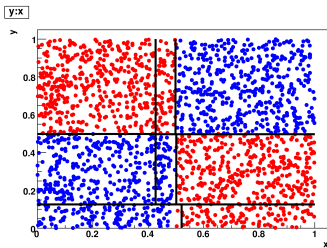


3 Boosted decision trees

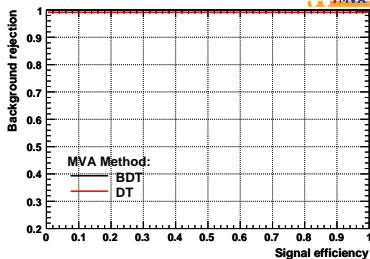
- Performance examples
 - XOR problem
 - Comparing train/test outputs
 - Variable ranking
 - Other averaging techniques

y:x

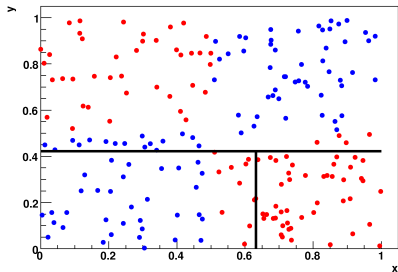




Background rejection versus Signal efficiency



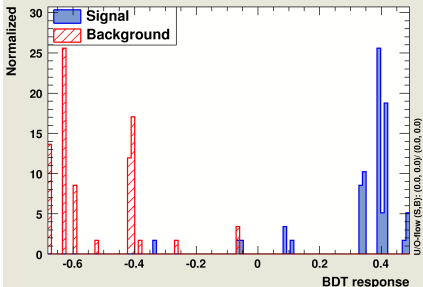
y:x



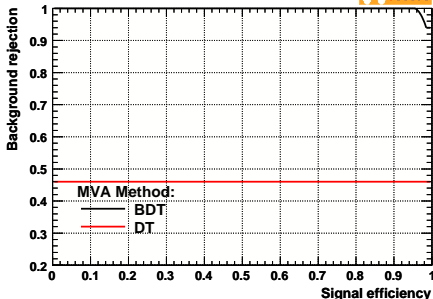
Small statistics

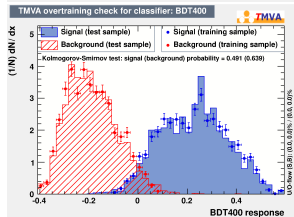
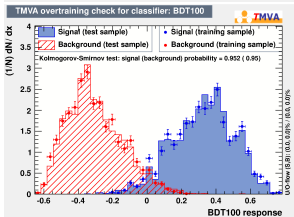
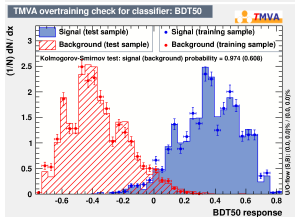
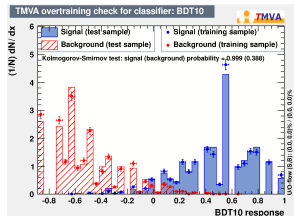
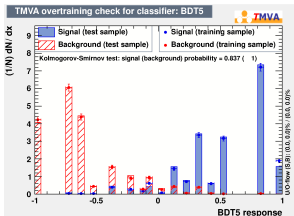
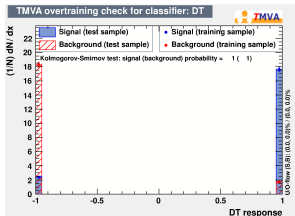
- Single tree not so good
- BDT very good: high performance discriminant from combination of weak classifiers

TMVA response for classifier: BDT

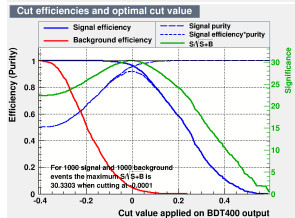
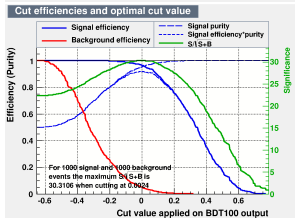
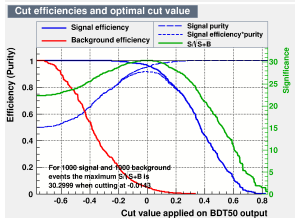
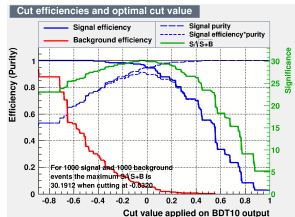
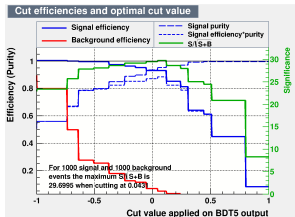
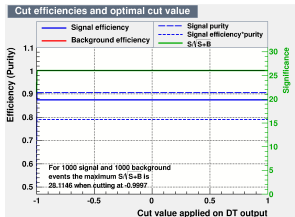


Background rejection versus Signal efficiency

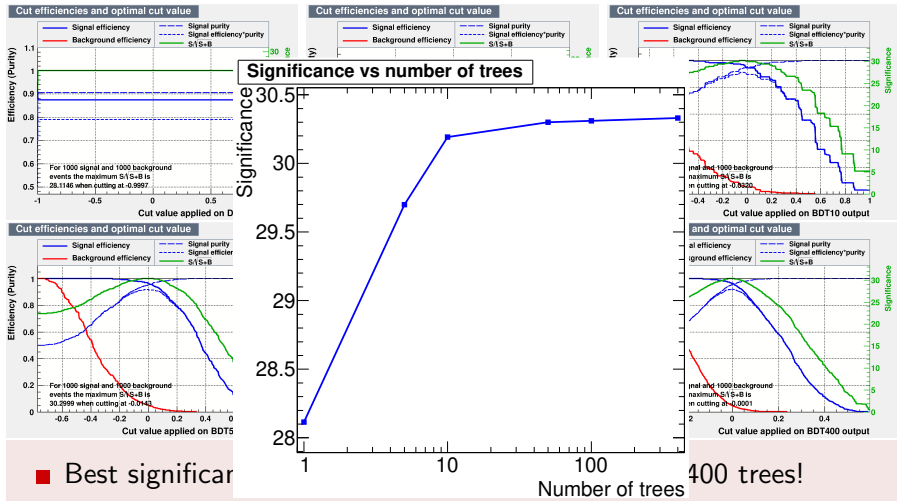




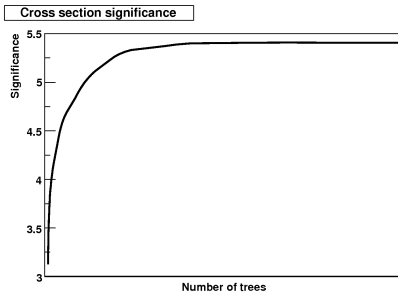
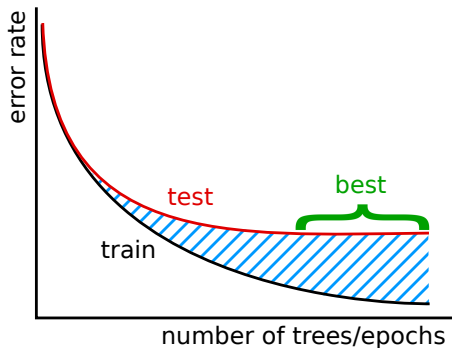
- Better shape with more trees: quasi-continuous
- Overtraining because of disagreement between training and testing?
Let's see. . .



- Best significance actually obtained with last BDT, 400 trees!
- But to be fair, equivalent performance with 10 trees already
- Less “stepped” output desirable? \Rightarrow maybe 50 is reasonable



- Best significant
- But to be fair, equivalent performance with 10 trees already
- Less “stepped” output desirable? ⇒ maybe 50 is reasonable



Variable importance (mean decrease impurity MDI)

- Ranking of x_i : add up decrease of impurity each time x_i is used
- Largest decrease of impurity (*during training*) = best variable

Shortcoming: masking of variables

- x_j may be just a little worse than x_i but will never be picked
- x_j is ranked as irrelevant
- But remove x_i , and x_j becomes very relevant
⇒ **careful with interpreting ranking** (*specific to training*)

Permutation importance (mean decrease accuracy MDA)

- Applicable to any *already trained* classifier
- Randomly shuffle each variable in turn and measure decrease of performance
- Important variable ⇒ big loss of performance
- Can also be performed on validation sample
- Beware of correlations

Bagging (Bootstrap aggregating)

[Breiman 1996]

- Before building tree T_k take random sample of N events from training sample with replacement
- Train T_k on it
- Events not picked form “out of bag” validation sample
- Applicable to other techniques than DT
 - tends to produce more stable and better classifier
- **Reduces variance of weak learners** (while boosting reduces bias)

Bagging (Bootstrap aggregating)

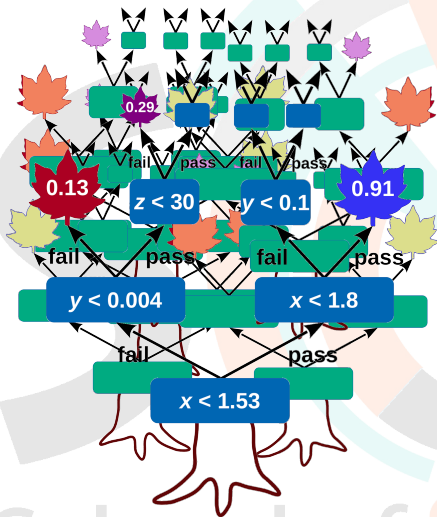
[Breiman 1996]

- Before building tree T_k take random sample of N events from training sample with replacement
- Train T_k on it
- Events not picked form “out of bag” validation sample
- Applicable to other techniques than DT
 - tends to produce more stable and better classifier
- **Reduces variance of weak learners** (while boosting reduces bias)

Random forests

[Breiman 2001]

- Same as bagging
 - In addition, pick random subset of variables to consider for each node split
 - Two levels of randomisation, much more stable output
 - Often as good as boosting
- [▶ see hands-on](#)

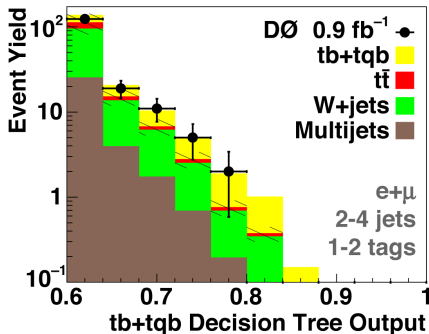
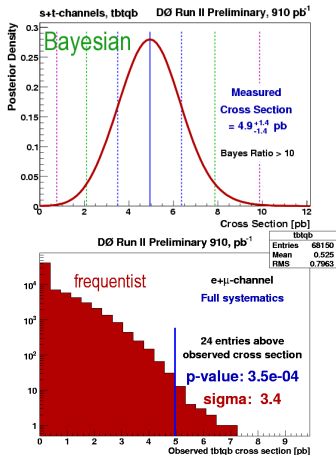


3 Boosted decision trees

- BDTs in real physics cases
 - Single top search at D0
 - LHC examples
 - Type Ia SN photometric classification
 - BDT and systematic uncertainties

- Three multivariate techniques: BDT, Matrix Elements, BNN
- Most sensitive: BDT

$\sigma_{s+t} = 4.9 \pm 1.4 \text{ pb}$
 $p\text{-value} = 0.035\% (3.4\sigma)$
 SM compatibility: 11% (1.3σ)



$\sigma_s = 1.0 \pm 0.9 \text{ pb}$
 $\sigma_t = 4.2^{+1.8}_{-1.4} \text{ pb}$

► Phys. Rev. D78, 012005 (2008)

Object Kinematics

$p_T(\text{jet1})$
 $p_T(\text{jet2})$
 $p_T(\text{jet3})$
 $p_T(\text{jet4})$
 $p_T(\text{best1})$
 $p_T(\text{notbest1})$
 $p_T(\text{notbest2})$
 $p_T(\text{tag1})$
 $p_T(\text{untag1})$
 $p_T(\text{untag2})$

Angular Correlations

$\Delta R(\text{jet1}, \text{jet2})$
 $\cos(\text{best1}, \text{lepton})_{\text{besttop}}$
 $\cos(\text{best1}, \text{notbest1})_{\text{besttop}}$
 $\cos(\text{tag1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{tag1}, \text{lepton})_{\text{btaggedtop}}$
 $\cos(\text{jet1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{jet1}, \text{lepton})_{\text{btaggedtop}}$
 $\cos(\text{jet2}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{jet2}, \text{lepton})_{\text{btaggedtop}}$
 $\cos(\text{lepton}, Q(\text{lepton}) \times z)_{\text{besttop}}$
 $\cos(\text{lepton}_{\text{besttop}}, \text{besttop})_{\text{CMframe}}$
 $\cos(\text{lepton}_{\text{btaggedtop}}, \text{btaggedtop})_{\text{CMframe}}$
 $\cos(\text{notbest}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{notbest}, \text{lepton})_{\text{besttop}}$
 $\cos(\text{untag1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{untag1}, \text{lepton})_{\text{btaggedtop}}$

Event Kinematics

$A_{\text{planarity}}(\text{alljets}, W)$
 $M(W, \text{best1})$ (“best” top mass)
 $M(W, \text{tag1})$ (“b-tagged” top mass)
 $H_T(\text{alljets})$
 $H_T(\text{alljets} - \text{best1})$
 $H_T(\text{alljets} - \text{tag1})$
 $H_T(\text{alljets}, W)$
 $H_T(\text{jet1}, \text{jet2})$
 $H_T(\text{jet1}, \text{jet2}, W)$
 $M(\text{alljets})$
 $M(\text{alljets} - \text{best1})$
 $M(\text{alljets} - \text{tag1})$
 $M(\text{jet1}, \text{jet2})$
 $M(\text{jet1}, \text{jet2}, W)$
 $M_T(\text{jet1}, \text{jet2})$
 $M_T(W)$
 $\text{Missing } E_T$
 $p_T(\text{alljets} - \text{best1})$
 $p_T(\text{alljets} - \text{tag1})$
 $p_T(\text{jet1}, \text{jet2})$
 $Q(\text{lepton}) \times \eta(\text{untag1})$
 \sqrt{s}
 $\text{Sphericity}(\text{alljets}, W)$

- Adding variables did not degrade performance
- Tested shorter lists, lost some sensitivity
- Same list used for all channels

Object Kinematics

$p_T(\text{jet1})$
 $p_T(\text{jet2})$
 $p_T(\text{jet3})$
 $p_T(\text{jet4})$
 $p_T(\text{best1})$
 $p_T(\text{notbest1})$
 $p_T(\text{notbest2})$
 $p_T(\text{tag1})$
 $p_T(\text{untag1})$
 $p_T(\text{untag2})$

Angular Correlations

$\Delta R(\text{jet1}, \text{jet2})$
 $\cos(\text{best1}, \text{lepton})_{\text{besttop}}$
 $\cos(\text{best1}, \text{notbest1})_{\text{besttop}}$
 $\cos(\text{tag1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{tag1}, \text{lepton})_{\text{btagedtop}}$
 $\cos(\text{jet1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{jet1}, \text{lepton})_{\text{btagedtop}}$
 $\cos(\text{jet2}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{jet2}, \text{lepton})_{\text{btagedtop}}$
 $\cos(\text{lepton}, Q(\text{lepton}) \times z)_{\text{besttop}}$
 $\cos(\text{lepton}_{\text{besttop}}, \text{besttop})_{\text{CMframe}}$
 $\cos(\text{lepton}_{\text{btagedtop}}, \text{btagedtop})_{\text{CMframe}}$
 $\cos(\text{notbest}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{notbest}, \text{lepton})_{\text{besttop}}$
 $\cos(\text{untag1}, \text{alljets})_{\text{alljets}}$
 $\cos(\text{untag1}, \text{lepton})_{\text{btagedtop}}$

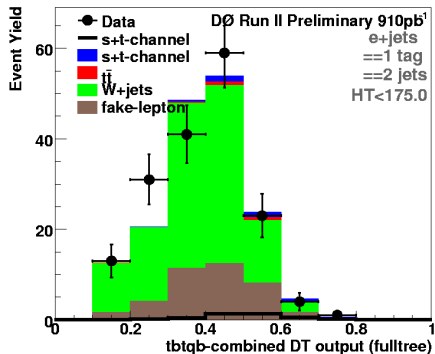
Event Kinematics

$A_{\text{planarity}}(\text{alljets}, W)$
 $M(W, \text{best1})$ (“best” top mass)
 $M(W, \text{tag1})$ (“b-tagged” top mass)
 $H_T(\text{alljets})$
 $H_T(\text{alljets} - \text{best1})$
 $H_T(\text{alljets} - \text{tag1})$
 $H_T(\text{alljets}, W)$
 $H_T(\text{jet1}, \text{jet2})$
 $H_T(\text{jet1}, \text{jet2}, W)$
 $M(\text{alljets})$
 $M(\text{alljets} - \text{best1})$
 $M(\text{alljets} - \text{tag1})$
 $M(\text{jet1}, \text{jet2})$
 $M(\text{jet1}, \text{jet2}, W)$
 $M_T(\text{jet1}, \text{jet2})$
 $M_T(W)$
 $\text{Missing } E_T$
 $p_T(\text{alljets} - \text{best1})$
 $p_T(\text{alljets} - \text{tag1})$
 $p_T(\text{jet1}, \text{jet2})$
 $Q(\text{lepton}) \times \eta(\text{untag1})$
 \sqrt{s}
 $\text{Sphericity}(\text{alljets}, W)$

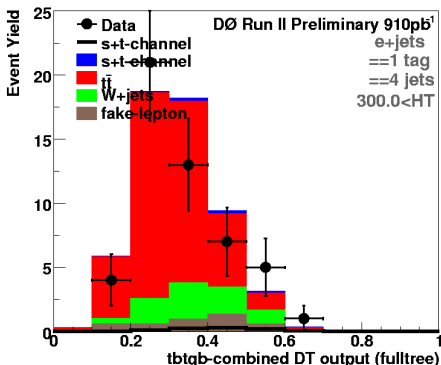
- Adding variables did not degrade performance
- Tested shorter lists, lost some sensitivity
- Same list used for all channels
- Best theoretical variable:
 $H_T(\text{alljets}, W)$.
 But detector not perfect \Rightarrow capture the essence from several variations usually helps “dumb” MVA

- Validate method on data in no-signal region

- **“W+jets”**: = 2 jets,
 $H_T(\text{lepton}, E_T^{\text{miss}}, \text{alljets}) < 175 \text{ GeV}$

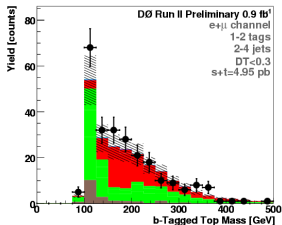


- **“ttbar”**: = 4 jets,
 $H_T(\text{lepton}, E_T^{\text{miss}}, \text{alljets}) > 300 \text{ GeV}$

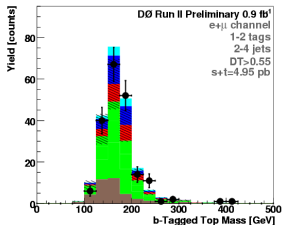


- Good agreement

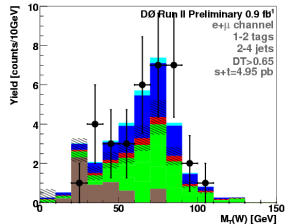
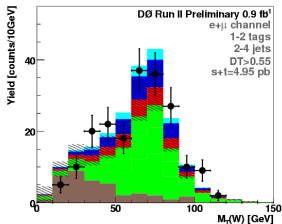
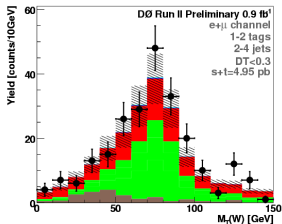
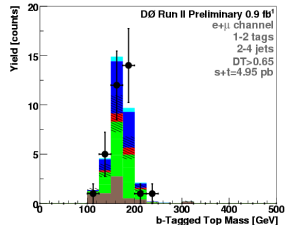
$DT < 0.3$



$DT > 0.55$

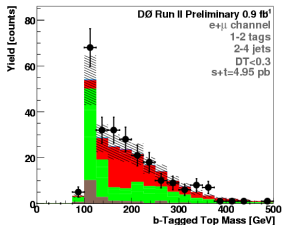


$DT > 0.65$

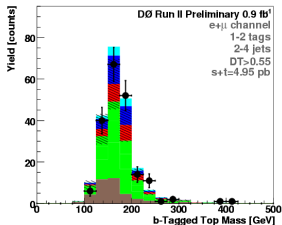


- High BDT region = shows masses of real t and $W \Rightarrow$ expected
- Low BDT region = background-like \Rightarrow expected

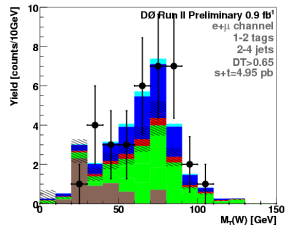
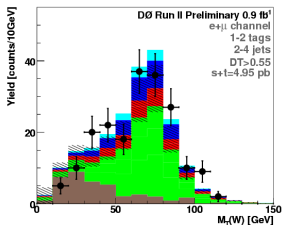
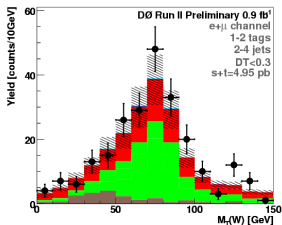
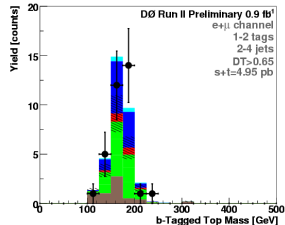
$DT < 0.3$



$DT > 0.55$



$DT > 0.65$

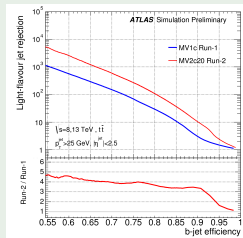


- High BDT region = shows masses of real t and W \Rightarrow expected
- Low BDT region = background-like \Rightarrow expected
- Above does NOT tell analysis is ok, but not seeing this could be a sign of a problem

ATLAS b -tagging in Run 2

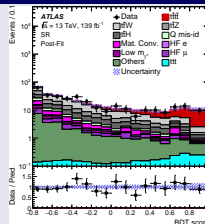
► Eur. Phys. J. C 79 (2019) 970

- Run 1 MV1c: NN trained from output of other taggers
- Run 2 MV2c20: BDT using feature variables of underlying algorithms and p_T , η of jets
- Run 2: introduced IBL (new innermost pixel layer)
 - ⇒ explains part of the performance gain, but not all



ATLAS $t\bar{t}t\bar{t}$ production evidence

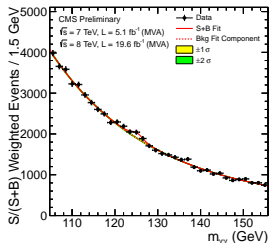
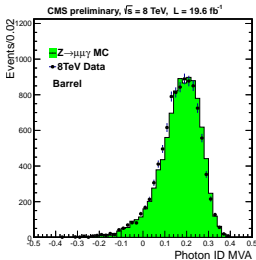
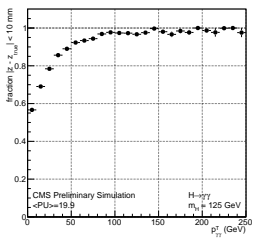
- Eur. Phys. J. C 80 (2020) 1085
- arXiv:2007.14858 [hep-ex]
- BDT output used in final fit to measure cross section
- Constraints on systematic uncertainties from profiling



► CMS-PAS-HIG-13-001

Hard to use more BDT in an analysis:

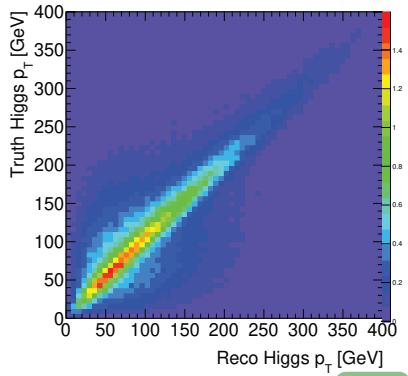
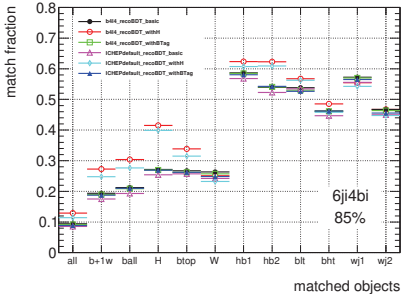
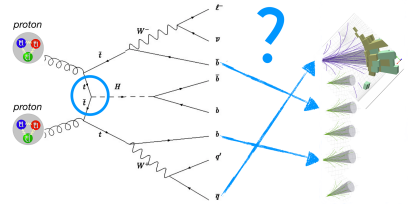
- vertex selected with BDT
- 2nd vertex BDT to estimate probability to be within 1cm of interaction point
- photon ID with BDT
- photon energy corrected with BDT regression
- event-by-event energy uncertainty from another BDT
- several BDT to extract signal in different categories



$t\bar{t}H(b\bar{b})$ reconstruction

- Match jets and partons in high-multiplicity final state
- BDT trained on all combinations
- New inputs to classification BDT
- Access to Higgs p_T , origin of b -jets

► Phys. Rev. D 97, 072016 (2018) ► arXiv:2111.06712 [hep-ex]



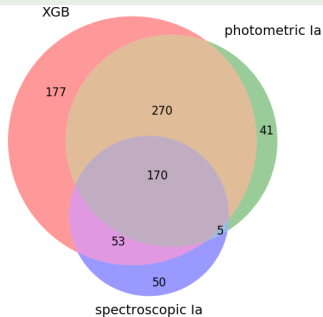
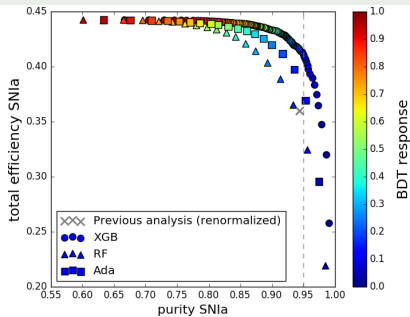
► thesis

► thesis

- Type Ia supernovae photometric classification
- Deriving redshifts from SN light curves on real data
- Tried random forest, AdaBoost and XGBoost

	AdaBoost	Random Forest	XGBoost
% SNIa ($CI = 4$ or 5)	74 ± 3	87 ± 3	96 ± 2
% SNIa* ($CI = 3$)	58 ± 6	73 ± 6	88 ± 4

► JCAP 12 (2016) 008



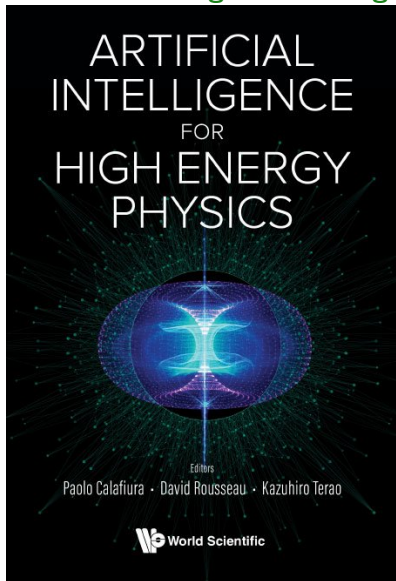
- No particular rule
- BDT output can be considered as any other cut variable (just more powerful). Evaluate systematics by:
 - varying cut value
 - retraining
 - calibrating, etc.
- Most common (and appropriate): propagate other uncertainties (detector, theory, etc.) up to BDT output and check how much the analysis is affected
- More and more common: profiling.
 Watch out:
 - BDT output powerful
 - signal region (high BDT output) probably low statistics
 ⇒ **potential recipe for disaster if modelling is not good**
- May require extra systematics, not so much on technique itself, but because it probes specific corners of phase space and/or wider parameter space (usually loosening pre-BDT selection cuts)

- Go for a fully integrated solution
 - use different multivariate techniques easily
 - spend your time on understanding your data and model
- Examples:
 - TMVA (Toolkit for MultiVariate Analysis) (C++/python)
Integrated in ROOT, complete manual [▶ https://root.cern/tmva](https://root.cern/tmva)
 - scikit-learn (python) [▶ https://scikit-learn.org](https://scikit-learn.org)
- Dedicated to BDT but transparently integrated with e.g. scikit-learn:
 - XGBoost (popular in HEP) [▶ arXiv:1603.02754](https://arxiv.org/abs/1603.02754) [▶ https://github.com/dmlc/xgboost](https://github.com/dmlc/xgboost)
(note: cannot handle negative weights)
 - LightGBM [Microsoft] [▶ https://lightgbm.readthedocs.io](https://lightgbm.readthedocs.io)
 - CatBoost [Yandex] [▶ https://catboost.ai/](https://catboost.ai/)

- **Decision trees:** natural extension to cut-based analysis
- Greatly improved performance with boosting (and also with bagging, random forests)
- **Boosted decision trees** still very common in HEP results
 - often using TMVA in ROOT or python
 - more and more with XGBoost, LightGBM, etc. ▶ see hands-on
- Possibly soon overpowered by deep learning algorithms, although NN trickier to optimise (*saying this for years, still not true*)
- Whichever technique you use, expect a lot of scepticism (but less and less with time): you will have to convince yourself and others that your advanced technique leads to meaningful and reliable results
 - ⇒ ensemble tests, use several techniques, compare to random grid search, etc. But DO NOT show them useless plots like BDT output on training and testing to measure overtraining, please!
- As with other advanced techniques,
 - no point in using them if data not understood and well modelled**

Artificial Intelligence for High Energy Physics

<https://doi.org/10.1142/12200>



Contents:

- Introduction (*Paolo Calafiura, David Rousseau and Kazuhiro Terao*)
- **Discriminative Models for Signal/Background Boosting:**
 - Boosted Decision Trees (*Yann Coadou*)
 - Deep Learning from Four Vectors (*Pierre Baldi, Peter Sadowski and Daniel Whiteson*)
 - Anomaly Detection for Physics Analysis and Less Than Supervised Learning (*Benjamin Nachman*)
- **Data Quality Monitoring:**
 - Data Quality Monitoring Anomaly Detection (*Adrian Alan Pol, Gianluca Cerminara, Cecile Germain and Maurizio Pierini*)
- **Generative Models:**
 - Generative Models for Fast Simulation (*Michela Paganini, Luke de Oliveira, Benjamin Nachman, Denis Derkach, Fedor Ratnikov, Andrey Ustyuzhanin and Aishik Ghosh*)
 - Generative Networks for LHC Events (*Anja Butter and Tilman Plehn*)
- **Machine Learning Platforms:**
 - Distributed Training and Optimization of Neural Networks (*Jean-Roch Vlimant and Junqi Yin*)
 - Machine Learning for Triggering and Data Acquisition (*Philip Harris and Nhan Tran*)
- **Detector Data Reconstruction:**
 - End-to-End Analyses Using Image Classification (*Adam Aurisano and Leigh H Whitehead*)
 - Clustering (*Kazuhiro Terao*)
 - Graph Neural Networks for Particle Tracking and Reconstruction (*Javier Duarte and Jean-Roch Vlimant*)
- **Jet Classification and Particle Identification from Low Level:**
 - Image-Based Jet Analysis (*Michael Kagan*)
 - Particle Identification in Neutrino Detectors (*Ralitsa Sharankova and Taritree Wongjirad*)
 - Sequence-Based Learning (*Rafael Teixeira de Lima*)
- **Physics Inference:**
 - Simulation-Based Inference Methods for Particle Physics (*Johann Brehmer and Kyle Cranmer*)
 - Dealing with Nuisance Parameters (*T Dorigo and P de Castro Manzano*)
 - Bayesian Neural Networks (*Tom Charnock, Laurence Perreault-Levasseur and François Lanusse*)
 - Parton Distribution Functions (*Stefano Forte and Stefano Carrazza*)
- **Scientific Competitions and Open Datasets:**
 - Machine Learning Scientific Competitions and Datasets (*David Rousseau and Andrey Ustyuzhanin*)



L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*, Wadsworth, Stamford, 1984



R.E. Schapire, "The strength of weak learnability" ▶ *Machine Learning* 5 (1990) 197



Y. Freund, "Boosting a weak learning algorithm by majority"

▶ *Information and computation* 121 (1995) 256



Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm" in *Machine Learning: Proceedings of the Thirteenth International Conference*, edited by L. Saitta (Morgan Kaufmann, San Fransisco, 1996) p. 148



Y. Freund and R.E. Schapire, "A short introduction to boosting"

▶ *Journal of Japanese Society for Artificial Intelligence* 14 (1999) 771











R. E. Schapire and Y. Freund, "Boosting: Foundations and Algorithms", MIT Press, 2012.



Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting" ▶ *Journal of Computer and System Sciences* 55 (1997) 119



J.H. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting" ▶ *Annals of Statistics* 28 (2000) 377

- 
 J. H. Friedman, “Greedy function approximation: A gradient boosting machine”
 ▶ [Annals of Statistics 29 \(2001\) 1189](#)
- 
 T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd edition)” ▶ [Springer Series in Statistics, 2009](#)
- 
 S. Shalev-Shwartz and S. Ben-David, “Understanding Machine Learning: From Theory to Algorithms” ▶ [Cambridge University Press, 2014](#)
- 
 M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off” ▶ [PNAS 116 \(2019\) 15849](#) ,
 ▶ [arXiv:1812.11118 \[stat.ML\]](#)
- 
 L. Breiman, “Bagging Predictors” ▶ [Machine Learning 24 \(1996\) 123](#)
- 
 L. Breiman, “Random forests” ▶ [Machine Learning 45 \(2001\) 5](#)
- 
 B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor
 ▶ [Nucl. Instr. Meth. A 543 \(2005\) 577](#) ; H.-J. Yang, B.P. Roe, and J. Zhu
 ▶ [Nucl. Instr. Meth. A 555 \(2005\) 370](#)
- 
 V. M. Abazov *et al.* [D0 Collaboration], “Evidence for production of single top quarks” ▶ [Phys. Rev. D 78 \(2008\) 012005](#)

Backup

▶ Pruning a decision tree

Pre-pruning

- Stop tree growth during building phase
- Already seen: minimum leaf size, minimum separation improvement, maximum depth, etc.
- Careful: early stopping condition may prevent from discovering further useful splitting

Expected error pruning

- Grow full tree
- When result from children not significantly different from result of parent, prune children
- Can measure statistical error estimate with binomial error $\sqrt{p(1-p)/N}$ for node with purity p and N training events
- No need for testing sample
- Known to be “too aggressive”

- Idea: penalise “complex” trees (many nodes/leaves) and find compromise between good fit to training data (larger tree) and good generalisation properties (smaller tree)
- With misclassification rate $R(T)$ of subtree T (with N_T nodes) of fully grown tree T_{max} :
 - cost complexity $R_\alpha(T) = R(T) + \alpha N_T$
 - $\alpha =$ complexity parameter
- Minimise $R_\alpha(T)$:
 - small α : pick T_{max}
 - large α : keep root node only, T_{max} fully pruned
- First-pass pruning, for terminal nodes t_L, t_R from split of t :
 - by construction $R(t) \geq R(t_L) + R(t_R)$
 - if $R(t) = R(t_L) + R(t_R)$ prune off t_L and t_R

- For node t and subtree T_t :
 - if t non-terminal, $R(t) > R(T_t)$ by construction
 - $R_\alpha(\{t\}) = R_\alpha(t) = R(t) + \alpha$ ($N_T = 1$)
 - if $R_\alpha(T_t) < R_\alpha(t)$ then branch has smaller cost-complexity than single node and should be kept
 - at critical $\alpha = \rho_t$, node is preferable
 - to find ρ_t , solve $R_{\rho_t}(T_t) = R_{\rho_t}(t)$, or:
$$\rho_t = \frac{R(t) - R(T_t)}{N_T - 1}$$
 - node with smallest ρ_t is *weakest link* and gets pruned
 - apply recursively till you get to the root node
 - This generates sequence of decreasing cost-complexity subtrees
 - Compute their true misclassification rate on validation sample:
 - will first decrease with cost-complexity
 - then goes through a minimum and increases again
 - pick this tree at the minimum as the best pruned tree
- Note: best pruned tree may not be optimal in a forest