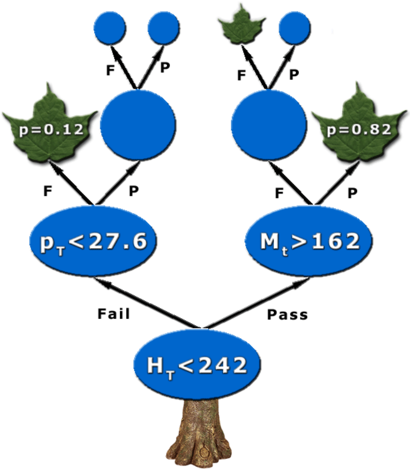
Arbres de décision boostés
Boosted decision trees

Yann Coadou

CPPM Marseille

School of Statistics SOS2010, Autrans
20 May 2010

Outline



Before we go on...

!!! VERY IMPORTANT !!!

Understand your inputs well
before you start playing with multivariate techniques

Introduction

Decision tree origin

- Machine-learning technique, widely used in social sciences.
  Originally data mining/pattern recognition, then medical diagnostic, 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 (or we would not be attending SOS2010...)
  - 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

Tree building algorithm

Start with all events = 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 is reached
- Splitting stops: terminal node = leaf

Algorithm example

- 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:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5} \leq H_T^{b_3} \leq \cdots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
    - $M_t^{b_6} \leq M_t^{s_8} \leq \cdots \leq M_t^{s_{12}} \leq M_t^{b_9}$

Algorithm example

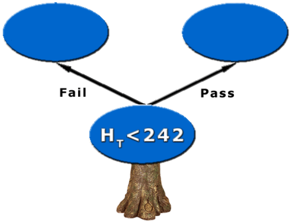
- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T's of all objects in the event H_T
  - sort all events by each variable:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5}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

Algorithm example

- 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:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5} \leq H_T^{b_3} \leq \cdots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
    - $M_t^{b_6} \leq M_t^{s_8} \leq \cdots \leq M_t^{s_{12}} \leq M_t^{b_9}$
  - best split (arbitrary unit):
    - $p_T < 56$ GeV, separation $= 3$
    - $H_T < 242$ GeV, separation $= 5$
    - $M_t < 105$ GeV, separation $= 0.7$

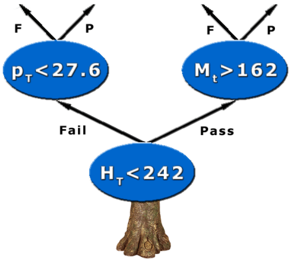
Algorithm example

- 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:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5} \leq H_T^{b_3} \leq \cdots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
    - $M_t^{b_6} \leq M_t^{s_8} \leq \cdots \leq M_t^{s_{12}} \leq M_t^{b_9}$
  - 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



Algorithm example

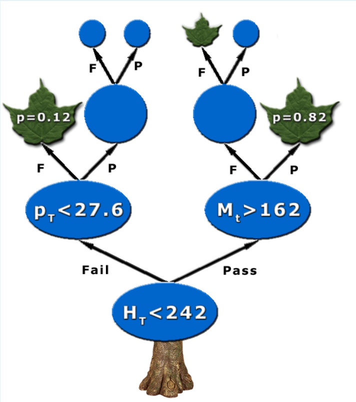
- Consider signal (s_i) and background (b_j) events described by 3 variables: p_T of leading jet, top mass M_t and scalar sum of p_T's of all objects in the event H_T
  - sort all events by each variable:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5}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

Algorithm example

- 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:
    - $p_T^{s_1} \leq p_T^{b_{34}} \leq \cdots \leq p_T^{b_2} \leq p_T^{s_{12}}$
    - $H_T^{b_5} \leq H_T^{b_3} \leq \cdots \leq H_T^{s_{67}} \leq H_T^{s_{43}}$
    - $M_t^{b_6} \leq M_t^{s_8} \leq \cdots \leq M_t^{s_{12}} \leq M_t^{b_9}$
  - 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 ($p = 0.82$)

Decision tree output

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 $+1$

Tree construction parameters

Normalization of signal and background before training

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

Selection of splits

- list of questions ($variable_i < cut_i$?, "Is the sky blue or overcast?")
- 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

Splitting a node

Impurity measure $i(t)$

- maximal for equal mix of signal and background
- symmetric in p_{signal} and $p_{background}$
- minimal for node with either signal only or background only
- strictly concave $\Rightarrow$ reward purer nodes (favours end cuts with one smaller node and one larger node)

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:
  $$\Delta i(s^*,t) = \max_{s \in \{\text{splits}\}} \Delta i(s,t)$$

Stopping condition

- See previous slide
- When not enough improvement ($\Delta i(s^*,t) < \beta$)
- Careful with early-stopping conditions

- Maximising $\Delta i(s,t) \equiv$ minimizing overall tree impurity

Splitting a node: examples

Node purity

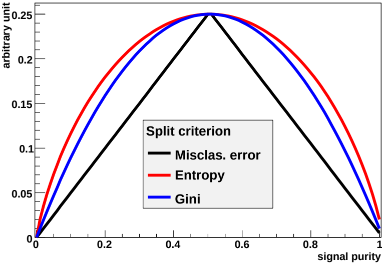
- Signal (background) event i with weight w_s^i (w_b^i)

$$p = \frac{\sum_{i \in signal} w_s^i}{\sum_{i \in signal} w_s^i + \sum_{j \in bkg} w_b^j}$$

- Signal purity (= purity)
  $p_s = p = \frac{s}{s+b}$
- Background purity
  $p_b = \frac{b}{s+b} = 1 - p_s = 1 - p$

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

Splitting a node: Gini index of diversity

Defined for many classes

- Gini $= \sum_{i,j\in\{\text{classes}\}}^{i\neq j} p_i p_j$

Statistical interpretation

- Assign random object to class i with probability p_i.
- Probability that it is actually in class j is p_j
- $\Rightarrow$ Gini $=$ probability of misclassification

For two classes (signal and background)

- $i = s, b$ and $p_s = p = 1 - p_b$
- $\Rightarrow$ Gini $= 1 - \sum_{i=s,b} p_i^2 = 2p(1-p) = \frac{2sb}{(s+b)^2}$
- Most popular in DT implementations
- Usually similar performance to e.g. entropy

Variable selection I

Reminder

- Need model giving good description of data

Variable selection I

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
- Insensitive to duplicate variables (give same ordering $\Rightarrow$ same DT)
- Variable order does not matter: all variables treated equal
- Order of training events is irrelevant
- Irrelevant variables:
  - no discriminative power (e.g. age of analyst) $\Rightarrow$ not used
  - only costs a little CPU time, no added noise
- Can use continuous and discrete variables, simultaneously

Variable selection II

Transforming input variables

- Completely insensitive to the replacement of any subset of input variables by (possibly different) arbitrary strictly monotone functions of them:
    - let $f : x_i \rightarrow f(x_i)$ be strictly monotone
    - if $x > y$ then $f(x) > f(y)$
    - ordering of events by x_i is the same as by $f(x_i)$
    - $\Rightarrow$ produces the same DT
- Examples:
    - 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

Variable selection III

Linear combinations of input variables

- Until now, answering questions like "is $x_i < c_i$?"
- Instead, take set of coefficients $a = (a_1, .., a_n)$, $||a||^2 = \sum_i a_i^2 = 1$
- Question: "is $\sum_i a_i x_i < c_i$?"
- Choose optimal split $s^*(a^*)$ and set of linear coefficients a^* that maximises $\Delta i(s^*(a), t)$
- Tricky to implement, very CPU intensive
- Only useful with strong linear correlations $\Rightarrow$ better to decorrelate first. DT will find them anyway, but inefficiently

Variable ranking

- Ranking of variable x_i: add up decrease of impurity at each node where x_i is used
- Largest decrease of impurity = best variable

Variable selection IV

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
  $\Rightarrow$ careful with interpreting ranking

Solution: surrogate split

- Compare which events are sent left or right by optimal split and by any other split
- Give higher score to split that mimics better the optimal split
- Highest score = surrogate split
- Can be included in variable ranking
- Helps in case of missing data: replace optimal split by surrogate

Tree instability

Training sample composition

- Small changes in sample can lead to very different tree structures
- Performance on testing events may be as good, or not
- Not optimal to understand data from DT rules
- Doesn't 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)

Pruning a tree I

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 is probably > 0, overtraining
- Pruning: eliminate subtrees (branches) that seem too specific to training sample:
  - a node and all its descendants turn into a leaf

Pruning a tree II

Pre-pruning

- Stop tree growth during building phase
- Already seen: minimum leaf size, minimum separation improvement, etc.

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"

Pruning a tree III

Cost-complexity pruning

- 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}:

$$\text{cost complexity } R_\alpha(T) = R(T) + \alpha N_T$$

  $\alpha =$ 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

Cost-complexity pruning

- 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

Tree (in)stability solution

Averaging

- Build several trees and average the output
- V-fold cross-validation (good for small samples)
  - divide training sample $\mathcal{L}$ in V subsets of equal size: $\mathcal{L} = \bigcup_{v=1..V} \mathcal{L}_v$
  - Train tree T_v on $\mathcal{L} - \mathcal{L}_v$, test on $\mathcal{L}_v$
  - DT output $= \frac{1}{V} \sum_{v=1..V} T_v$
- Bagging, boosting, random forests, etc.

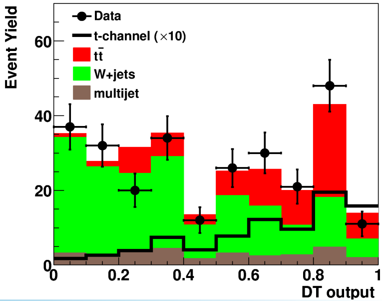
Decision tree score card

- ✔ Training is fast
- ✔ Human readable (not a black box, can interpret tree as selection rules or physics)

- ✔ Deals with continuous and discrete variables simultaneously
- ✔ No need to transform inputs
- ✔ Resistant to irrelevant variables
- ✔ Works well with many variables
- ✘ Good variables can be masked

- ✔ Very few parameters
- ✔ For some time still "original" in HEP

- ✘ Unstable tree structure
- ✘ Piecewise nature of output



A brief history of boosting

First provable algorithm by 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)

A brief history of boosting

First provable algorithm by 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 by Freund (1995): boost by majority (combining many learners with fixed error rate)
- Freund&Schapire joined forces: 1^{st} functional model AdaBoost (1996)

A brief history of boosting

First provable algorithm by 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 by Freund (1995): boost by majority (combining many learners with fixed error rate)
- Freund&Schapire joined forces: 1$^{\text{st}}$ functional model AdaBoost (1996)

"Recently" in HEP

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

Principles of boosting

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

Algorithm

- Training sample $\mathbb{T}_k$ of N events. For i^{th} event:
  - weight w_i^k
  - vector of discriminative variables x_i
  - class label $y_i = +1$ for signal, -1 for background
- Pseudocode:

  Initialise $\mathbb{T}_1$

  for k in 1..N_{tree}

    train classifier T_k on $\mathbb{T}_k$

    assign weight α_k to T_k

    modify $\mathbb{T}_k$ into $\mathbb{T}_{k+1}$
- Boosted output: $F(T_1, .., T_{N_{tree}})$

AdaBoost

What is AdaBoost?

- Introduced by Freund&Schapire in 1996
- Stands for *adaptive boosting*
- Learning procedure adjusts to training data to classify it better
- Many variations on the same theme for actual implementation
- Most common boosting algorithm around
- Usually leads to better results than without boosting

AdaBoost algorithm

- 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 $\{\pm 1\}$: $\text{isMisclassified}_k(i) = \mathbb{I}\big(y_i \times T_k(x_i) \leq 0\big)$
  - or $\text{isMisclassified}_k(i) = \mathbb{I}\big(y_i \times (T_k(x_i) - 0.5) \leq 0\big)$ in purity convention
  - misclassification rate:

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

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

$$w_i^k \rightarrow w_i^{k+1} = w_i^k \times e^{\alpha_k}$$

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

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

AdaBoost by example

- Assume $\beta = 1$

Not-so-good classifier

- Assume error rate $\epsilon = 40\%$
- Then $\alpha = \ln \frac{1-0.4}{0.4} = 0.4$
- Misclassified events get their weight multiplied by $e^{0.4}$=1.5
- $\Rightarrow$ next tree will have to work a bit harder on these events

Good classifier

- Error rate $\epsilon = 5\%$
- Then $\alpha = \ln \frac{1-0.05}{0.05} = 2.9$
- Misclassified events get their weight multiplied by $e^{2.9}$=19 (!!)
- $\Rightarrow$ being failed by a good classifier means a big penalty:
  - must be a difficult case
  - next tree will have to pay much more attention to this event and try to get it right

AdaBoost error rate

Misclassification rate ϵ on training sample

- Can be shown to be bound:
$$\epsilon \leq \prod_{k=1}^{N_{tree}} 2\sqrt{\epsilon_k(1-\epsilon_k)}$$
- If each tree has $\epsilon_k \neq 0.5$ (i.e. better than random guessing):

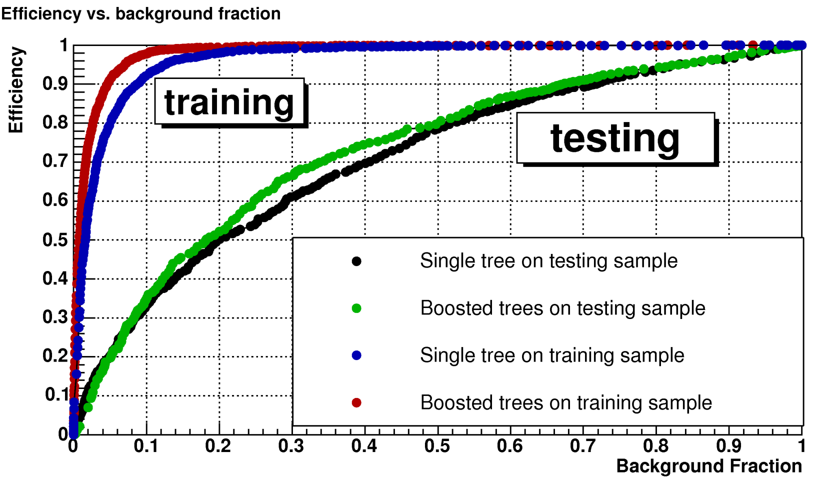
the error rate falls to zero for sufficiently large N_{tree}

- Corollary: training data is over fitted

Overtraining?

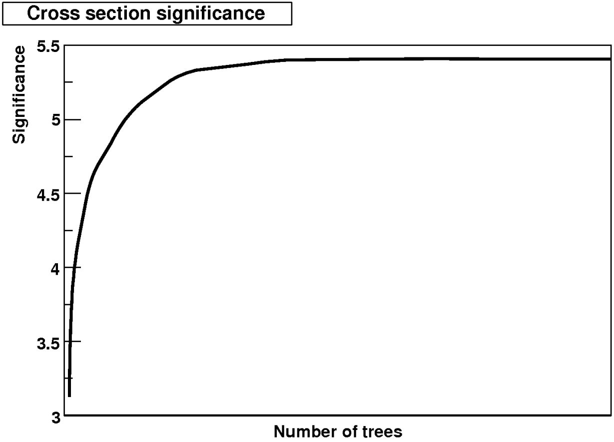
- Error rate on test sample may reach a minimum and then potentially rise. Stop boosting at the minimum.
- In principle AdaBoost *must* overfit training sample
- In many cases in literature, no loss of performance due to overtraining
  - may have to do with fact that successive trees get in general smaller and smaller weights
  - trees that lead to overtraining contribute very little to final DT output on validation sample

Training and generalisation error



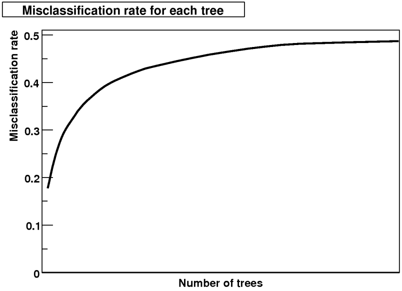
- Clear overtraining, but still better performance after boosting

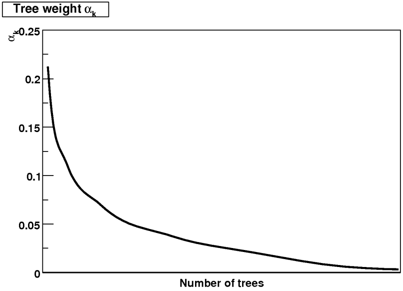
Cross section significance



- More relevant than testing error
- Reaches plateau
- Afterwards, boosting does not hurt (just wasted CPU)

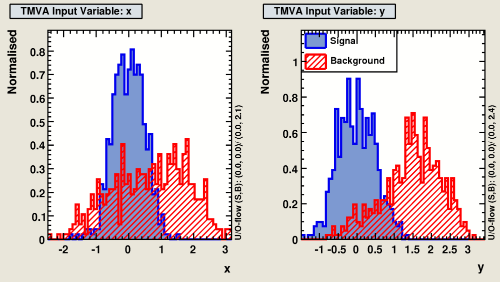
Clues to boosting performance

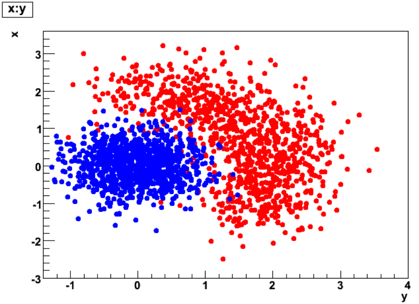




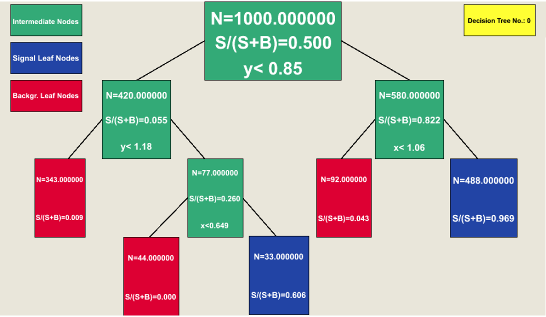
- First tree is best, others are minor corrections
- Specialised trees do not perform well on most events $\Rightarrow$ decreasing tree weight and increasing misclassification rate
- Last tree is not better evolution of first tree

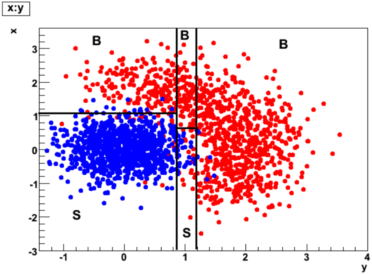
- Using TMVA and some code modified from G. Cowan's CERN academic lectures (June 2008)



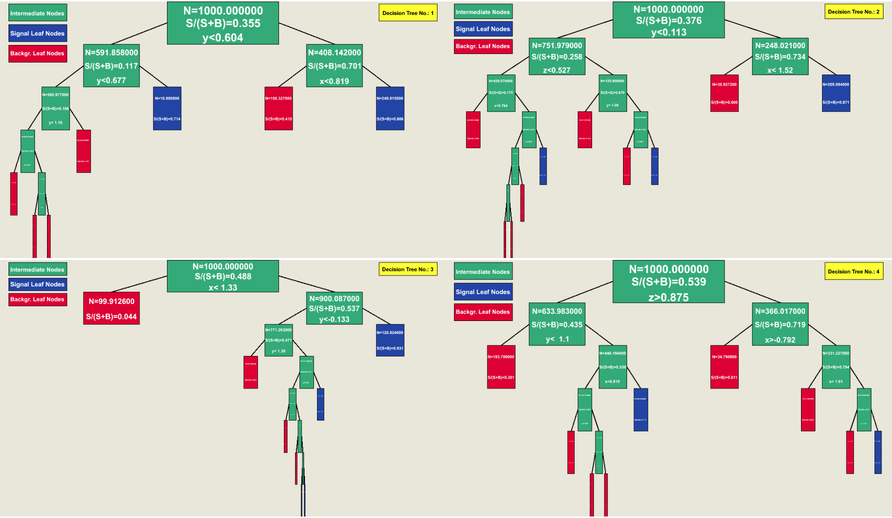


Concrete examples II



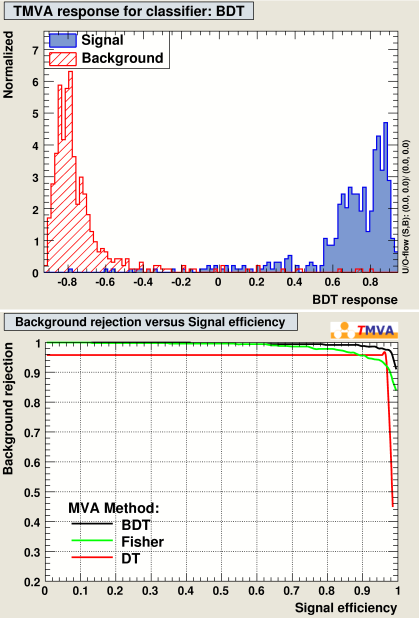


Concrete examples III

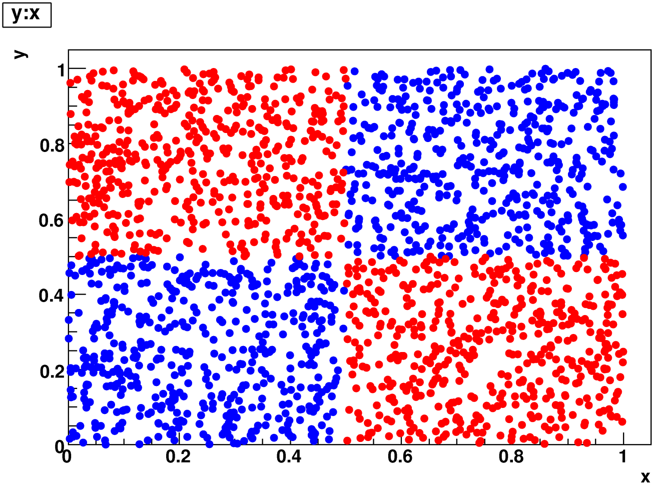


- Specialised trees

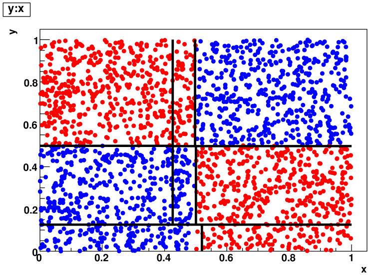
Concrete examples IV

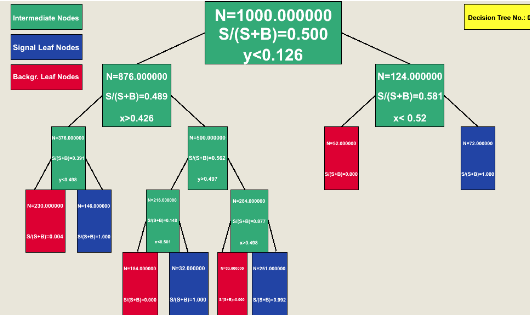


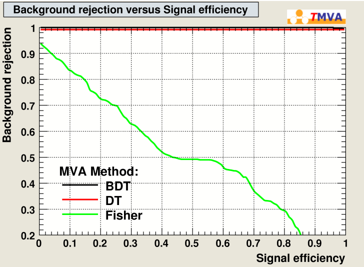
Concrete example: XOR



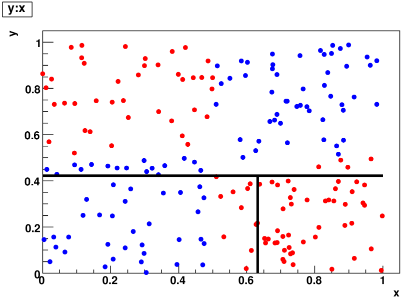
Concrete example: XOR





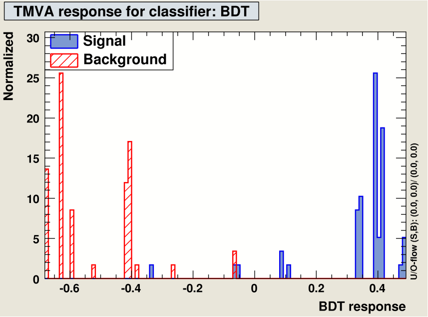


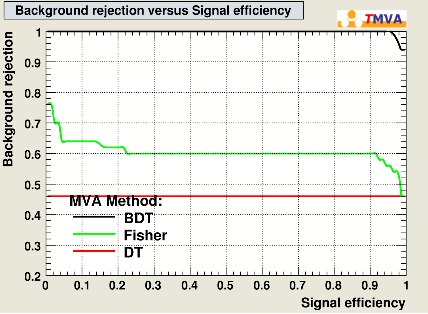
Concrete example: XOR with 100 events



Small statistics

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





Other boosting algorithms

ϵ-Boost (shrinkage)

- reweight misclassified events by a fixed $e^{2\epsilon}$ factor
- $T(i) = \sum_{k=1}^{N_{\text{tree}}} \epsilon T_k(i)$

ϵ-LogitBoost

- reweight misclassified events by logistic function $\frac{e^{-y_i T_k(x_i)}}{1+e^{-y_i T_k(x_i)}}$
- $T(i) = \sum_{k=1}^{N_{\text{tree}}} \epsilon T_k(i)$

Real AdaBoost

- DT output is $T_k(i) = 0.5 \times \ln \frac{p_k(i)}{1-p_k(i)}$ where $p_k(i)$ is purity of leaf on which event i falls
- reweight events by $e^{-y_i T_k(i)}$
- $T(i) = \sum_{k=1}^{N_{\text{tree}}} T_k(i)$

- ϵ-HingeBoost, LogitBoost, Gentle AdaBoost, etc.

Other averaging techniques

Bagging (Bootstrap aggregating)

- 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

Other averaging techniques

Bagging (Bootstrap aggregating)

- 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

Random forests

- Same as bagging
- In addition, pick random subset of variables to consider for each node split
- Two levels of randomisation, much more stable output

Other averaging techniques

Bagging (Bootstrap aggregating)

- 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

Random forests

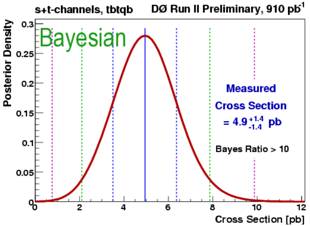
- Same as bagging
- In addition, pick random subset of variables to consider for each node split
- Two levels of randomisation, much more stable output

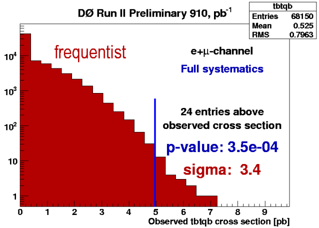
Trimming

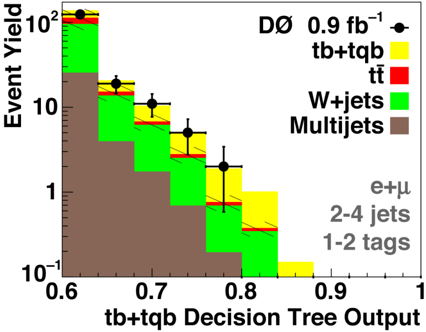
- Not exactly the same. Used to speed up training
- After some boosting, very few high weight events may contribute
- $\Rightarrow$ ignore events with too small a weight

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

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

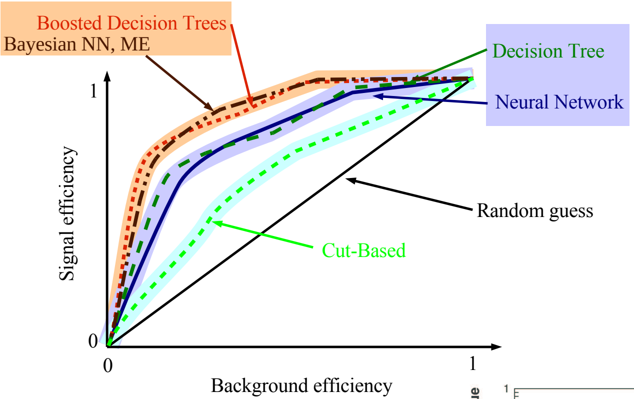




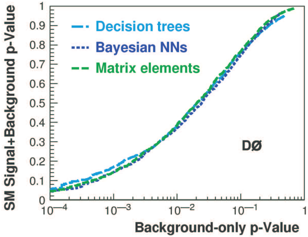


$\sigma_s = 1.0 \pm 0.9$ pb
$\sigma_t = 4.2^{+1.8}_{-1.4}$ pb

Comparison for D0 single top evidence

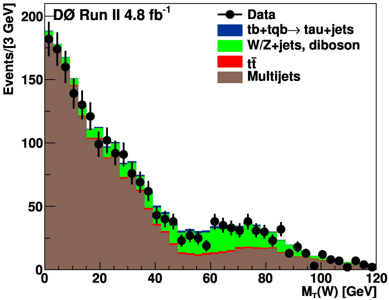


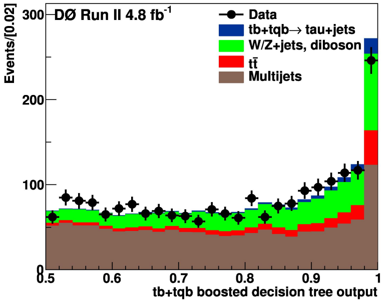
- Cannot know *a priori* which method will work best
- ⇒ Need to experiment with different techniques

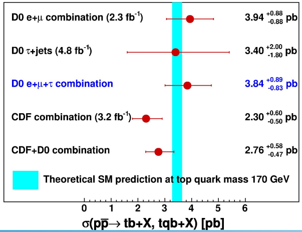


Search for single top in tau+jets at D0 (2010)

- Tau ID BDT and single top search BDT







- 4% sensitivity gain over $e + \mu$ analysis
- In press in PLB

Boosted decision trees in HEP studies

- MiniBooNE (e.g. physics/0408124 NIM A543:577-584, physics/0508045 NIM A555:370-385, hep-ex/0704.1500)
- D0 single top evidence (PRL98:181802,2007, PRD78:012005,2008)
- D0 and CDF single top quark observation (PRL103:092001,2009, PRL103:092002,2009)
- D0 tau ID and single top search (in press in PLB)
- GLAST (same code as D0)
- BaBar (hep-ex/0607112)
- ATLAS: diboson analyses, SUSY analysis (hep-ph/0605106 JHEP060740), single top CSC note, tau ID
- *b*-tagging for LHC (physics/0702041)
- Electron ID in CMS
- More and more underway

Conclusion

- Decision trees have been around for some time in social sciences
- Natural extension to cut-based analysis
- Greatly improved performance with boosting (and also with bagging, random forests)
- Becoming rather fashionable in HEP
- Even so, expect a lot of scepticism: you'll have to convince people that your advanced technique leads to meaningful and reliable results $\Rightarrow$ ensemble tests, use several techniques, compare to random grid search, etc.
- As with other advanced techniques, no point in using them if data are not understood and well modelled
- Even less point optimising MVA to death if you have no data...

Boosted decision tree software

- Historical: CART, ID3, C4.5
- D0 analysis: C++ custom-made code. Can use entropy/Gini, boosting/bagging/random forests
- MiniBoone code at http://www-mhp.physics.lsa.umich.edu/∼roe/

Much better approach

- Go for a fully integrated solution
  - use different multivariate techniques easily
  - spend your time on understanding your data and model
- Examples:
  - Weka. Written in Java, open source, very good published manual. Not written for HEP but very complete http://www.cs.waikato.ac.nz/ml/weka/
  - StatPatternRecognition http://www.hep.caltech.edu/∼narsky/spr.html
  - **TMVA (Toolkit for MultiVariate Analysis). Now integrated in ROOT, complete manual. http://tmva.sourceforge.net**

- L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*, Wadsworth, Stamford, 1984
- J.R. Quinlan, "Induction of decision trees", *Machine Learning*, 1(1):81–106, 1986
- J.R. Quinlan, "Simplifying decision trees", *International Journal of Man-Machine Studies*, 27(3):221–234, 1987
- R.E. Schapire, "The strength of weak learnability", *Machine Learning*, 5(2):197–227,1990
- Y. Freund, "Boosting a weak learning algorithm by majority", *Information and computation*. 121(2):256–285, 1995
- 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(5):771-780 (1999)*

References II

- 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(1):119–139, 1997
- J.H. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", *The Annals of Statistics*, 28(2), 377–386, 2000
- L. Breiman, "Bagging Predictors", *Machine Learning*, 24 (2), 123–140, 1996
- L. Breiman, "Random forests", *Machine Learning*, 45 (1), 5–32, 2001
- B.P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor, Nucl. Instrum. Methods Phys. Res., Sect.A 543, 577 (2005); H.-J. Yang, B.P. Roe, and J. Zhu, Nucl. Instrum.Methods Phys. Res., Sect. A 555, 370 (2005)
- V. M. Abazov *et al.* [D0 Collaboration], "Evidence for production of single top quarks,", Phys. Rev. D**78**, 012005 (2008)