Introduction Boosted Decision Trees (BDTs)

EASY-ML 2025

Speakers

Dr Luca Cadamuro

Dr Luca is a researcher at the Irène Joliot-Curie Laboratory (IJCLab) of CNRS in Orsay, France.

Dr Charles Ndung'u Ndegwa

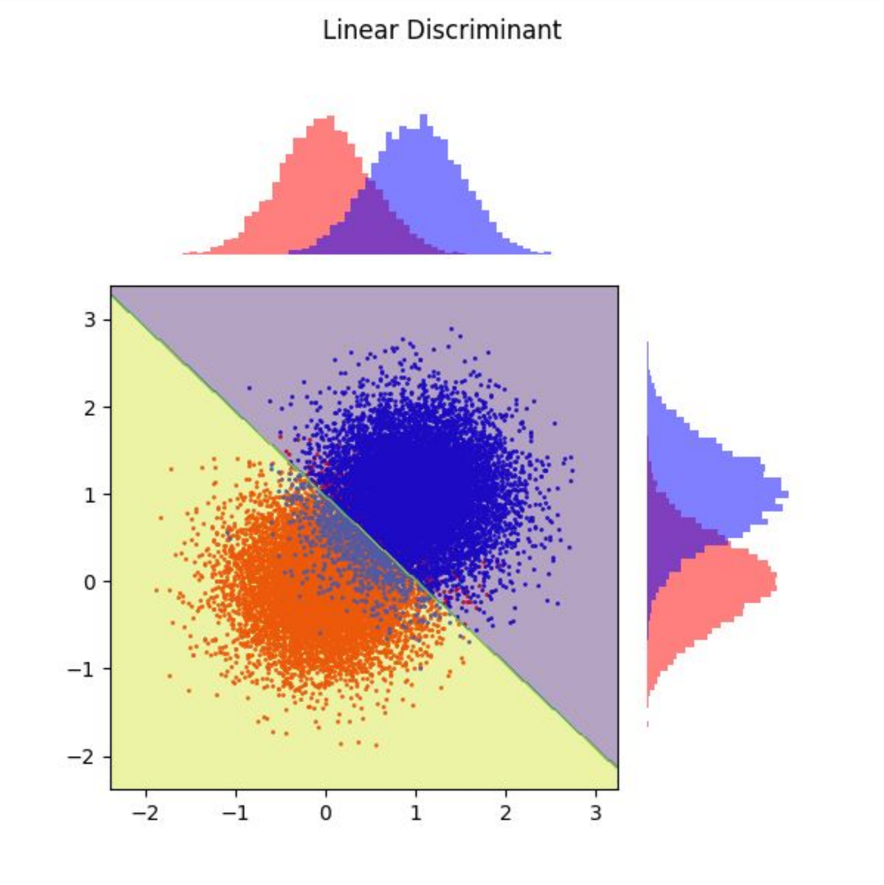
Postdoctoral researcher at Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), CNRS, Université Paris-Saclay in France.

Contents

Decision trees

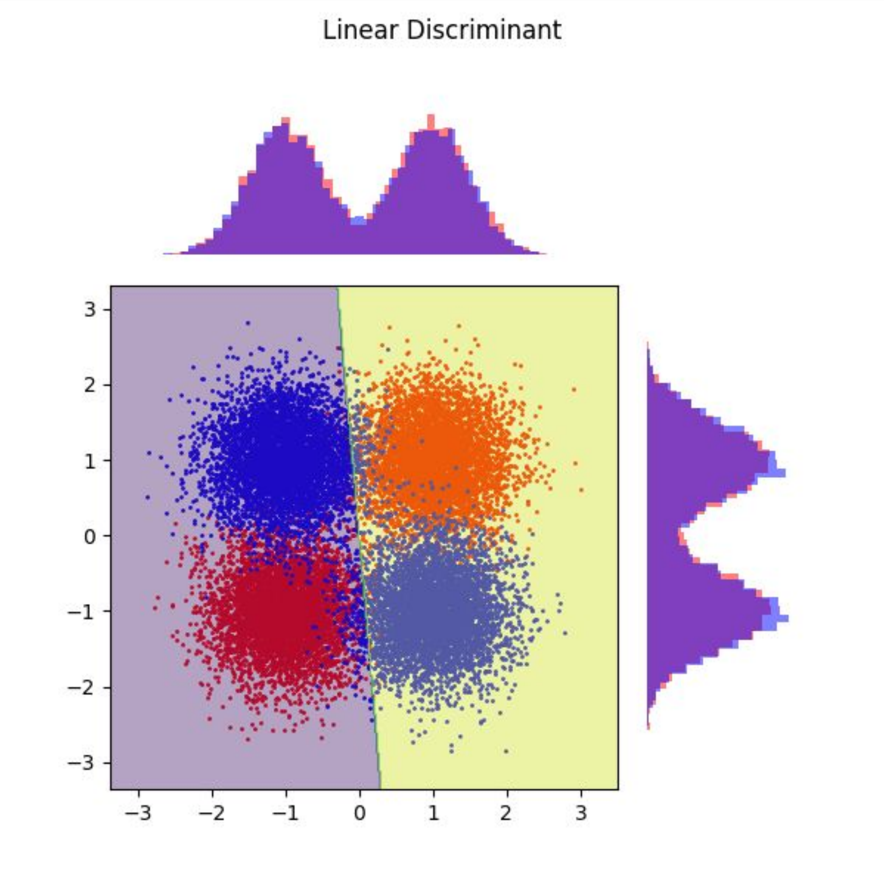
and their limits

Why decision trees?



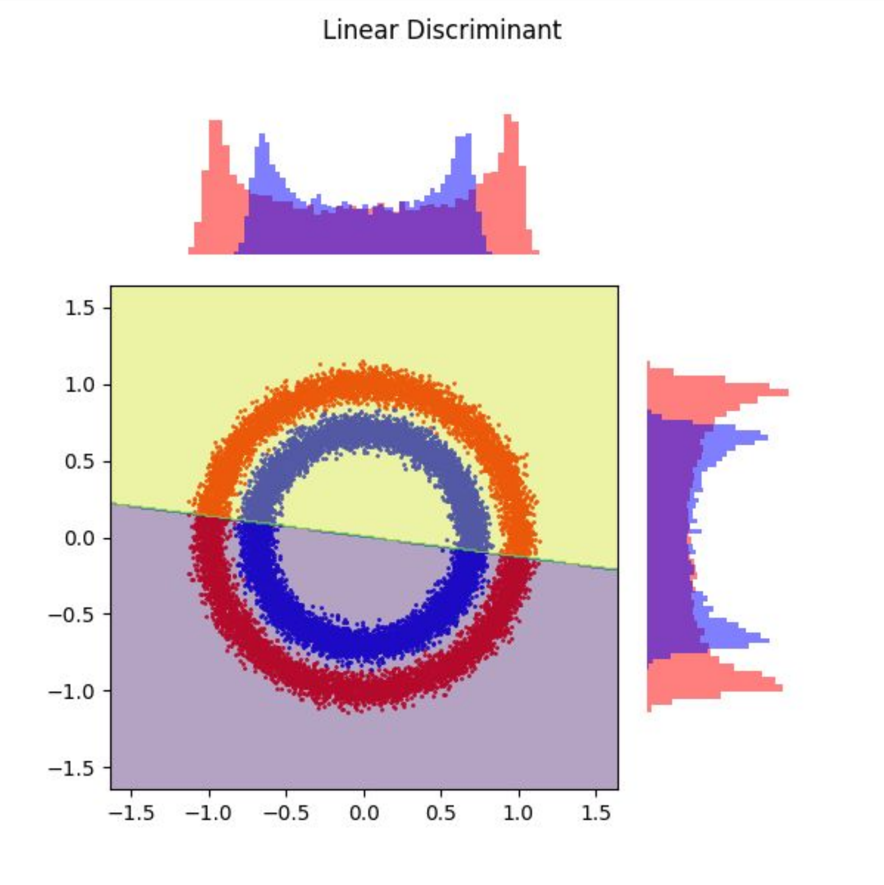
- Whenever data are described by simple correlations, linear discriminants do just fine in classification problems

Why decision trees?



- But they fail when data have non-trivial correlations

Why decision trees?

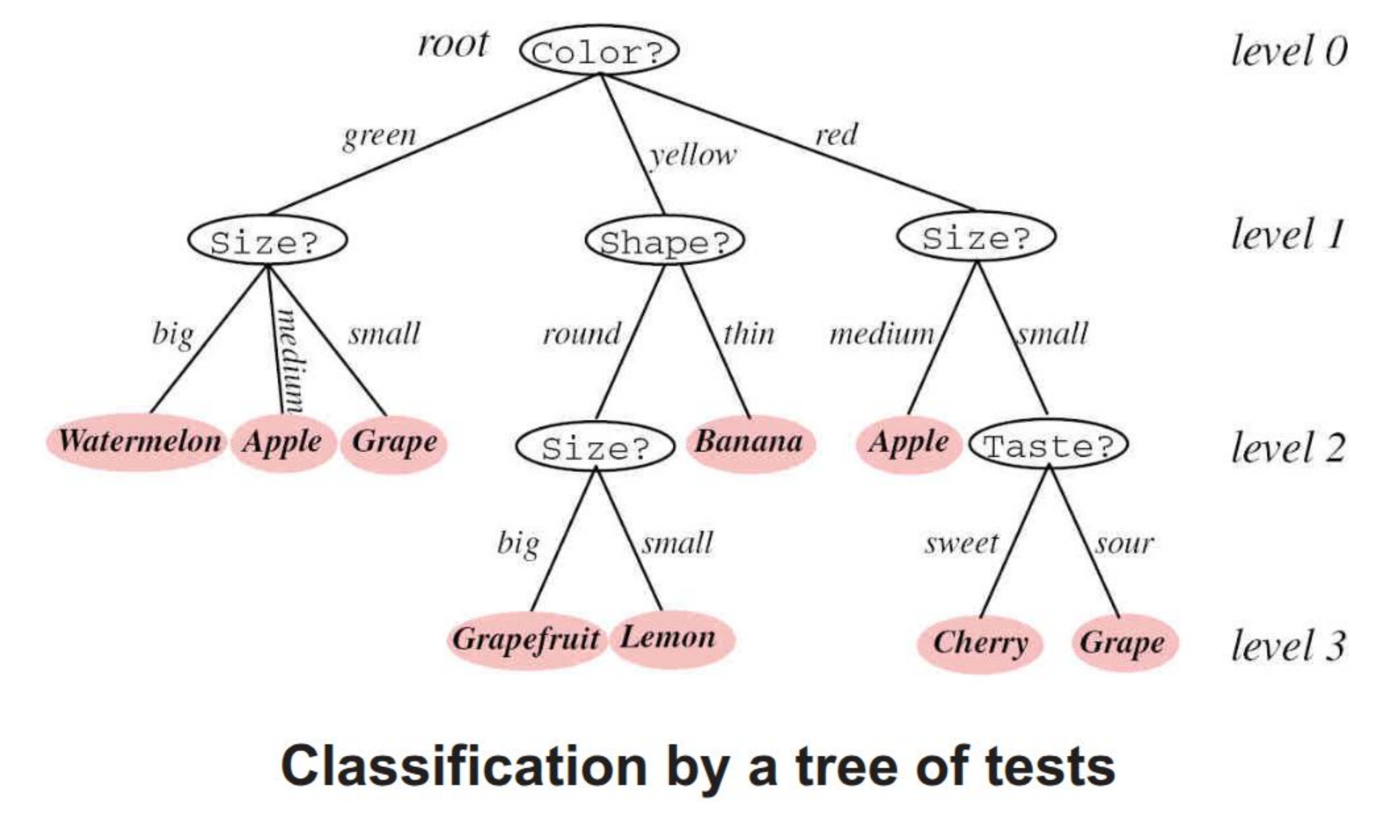


- But they fail when data have non-trivial correlations

We need a way to identify complex, non-linear features in our data

- Here, we would like to draw several, small linear segments to separate our data in a way that a simple linear surface cannot do ⇒ decision trees

What is a Decision Tree?



Source: Lucas Masuch & Vincent Lepetit

Decision Trees

The idea of decision trees is to partition the input space into regions and solving each region with a simpler model.

We focus on **Classification and Regression Trees** (CART; Breiman et al., 1984), but there are additional variants like ID3, C4.5, ...

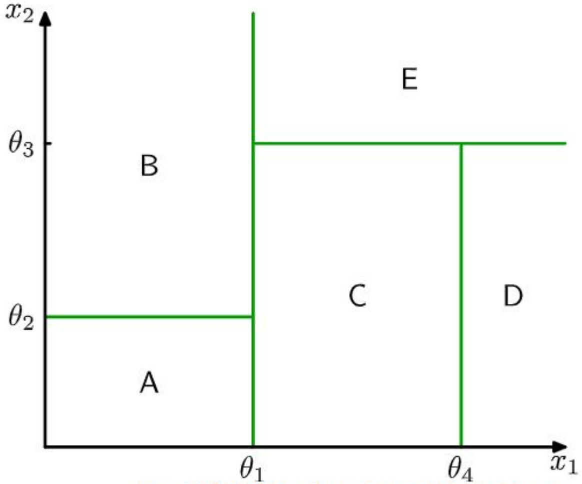


Figure 14.6 of Pattern Recognition and Machine Learning.

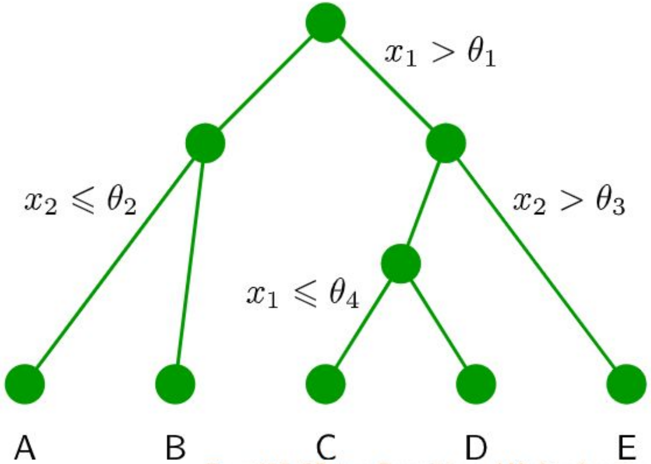
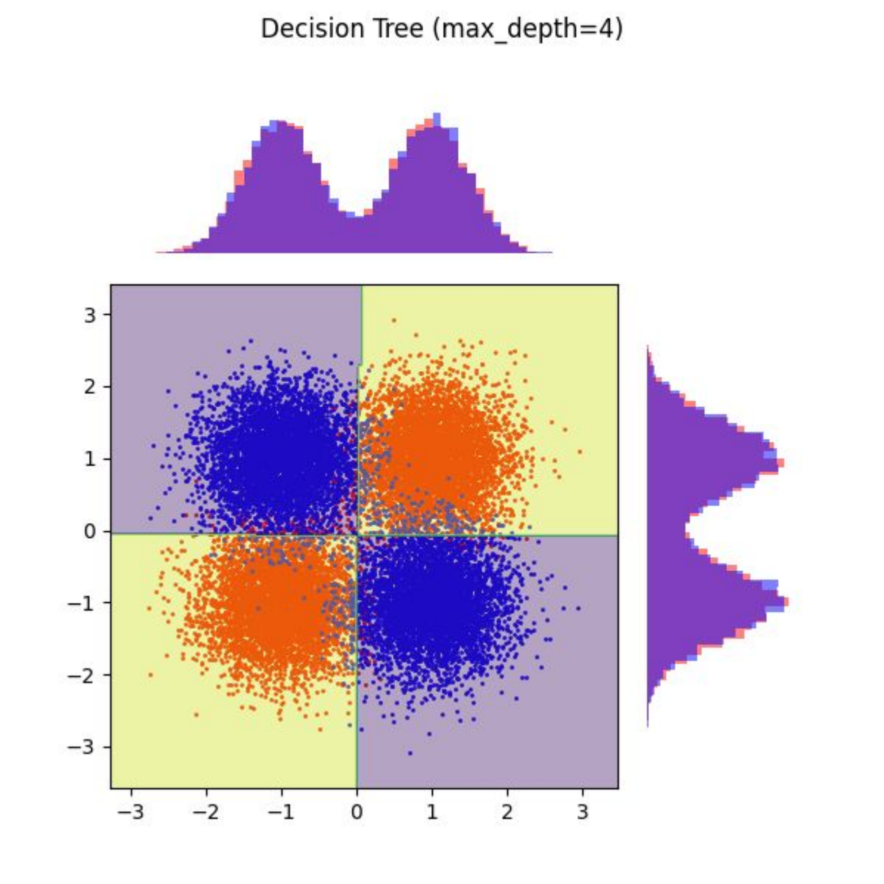


Figure 14.5 of Pattern Recognition and Machine Learning.

Source: Lucas Masuch & Vincent Lepetit

Back to our examples



- A simple Decision Tree with at most 4 levels of splitting can easily identify the four corners of our problem
- Let's see what is behind and how we train this Decision Tree

Decision Trees

- Decisions trees: splitting each variable sequentially, creating rectangular regions.
- Making fitting/prediction locally at each hyper-rectangular region.
- It is intuitive and easy to implement, may have good interpretation.
- Generally of lower prediction accuracy (weak learners).
- "The Boosting problem" (Kearns & Valiant): *Can a set of weak learners create a single strong learner?*
- Bagging, random forests and boosting ... make fitting/prediction based on a number of trees.
- Bagging and Boosting are general methodologies, not just limited to trees.

Inference and Training

Inference

- Just follow the branching rules until you reach a leaf.
- Output a prediction (real value/distribution/predicted class) based on the leaf.

https://medium.com/analytics-vidhya/decision-trees-explained-in-simple-steps-39ee1a6b00a2

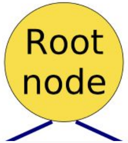
Training

- Training data is stored in tree leaves -- the leaf prediction is based on what is data items are in the leaf.
- At the beginning the tree is a single leaf node.
- Adding a node = leaf $\rightarrow$ decision node + 2 leafs
- The goal of training = finding the most consistent leafs for the prediction

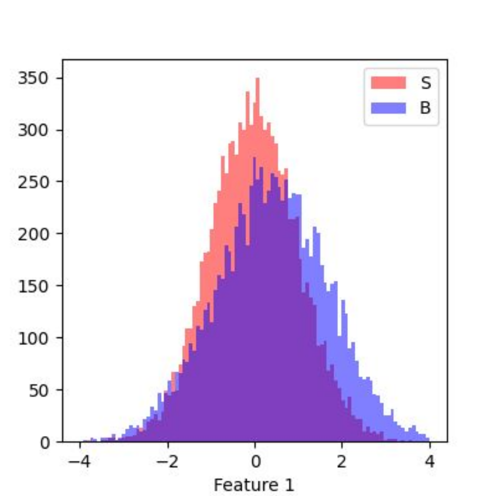
Later, we will show that the consistency measures follow from the loss function, we are optimizing.

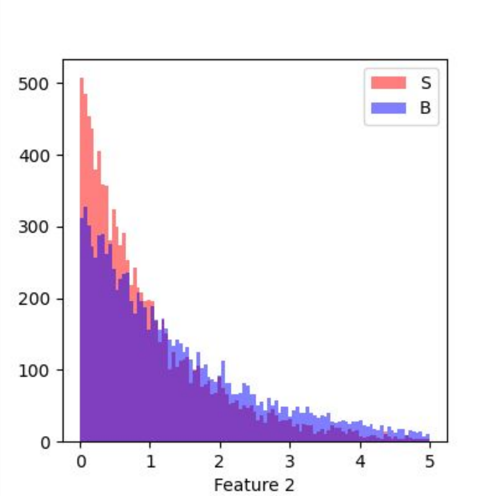
How do you train a decision tree ?

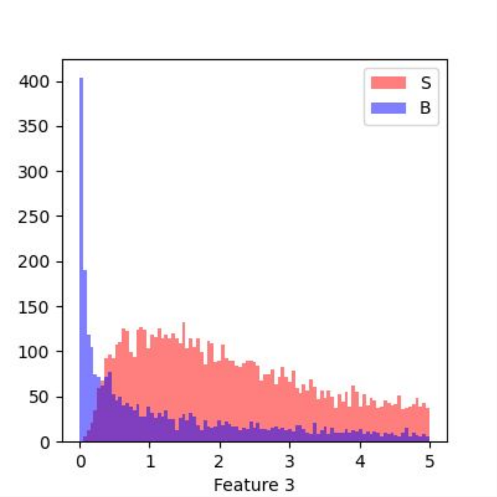
Take all features



Generic example of samples S and B with 3 features



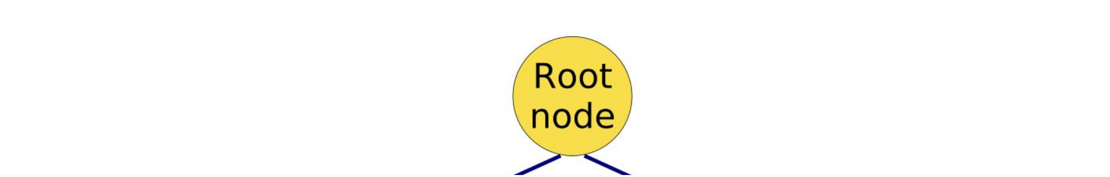




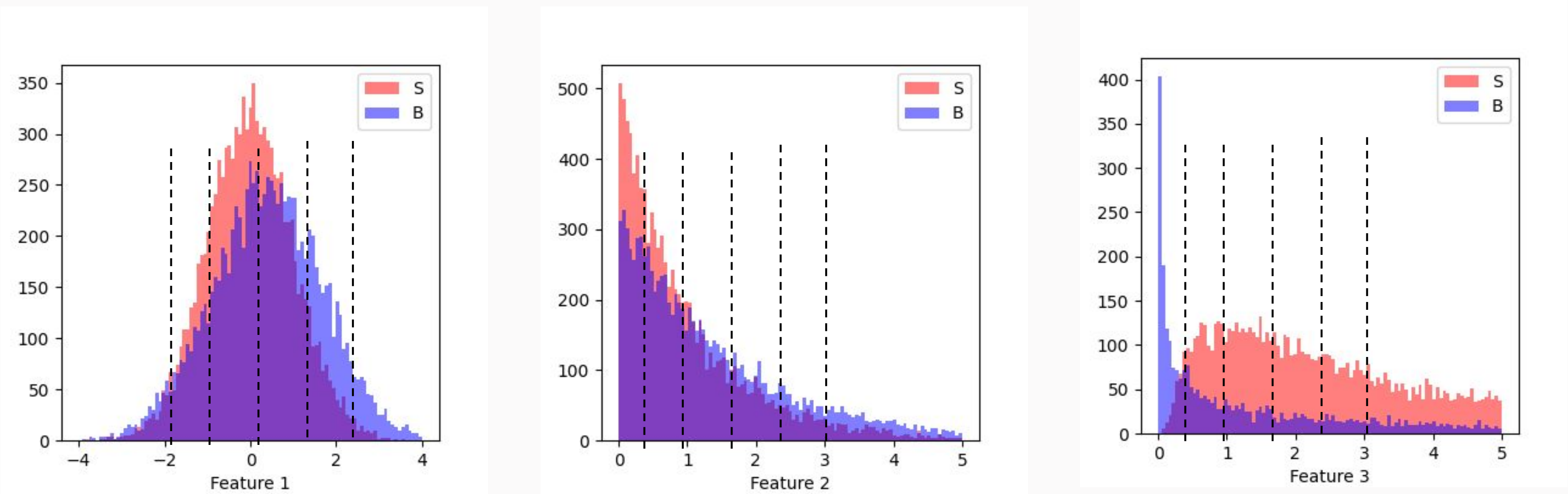
How do you train a decision tree ?

Take all features

Compare their separating power



Loop over all features "f_i"
For each value, check separation at a given cut "f_i > "c_ij"
Find the variable and cut giving the best separation



How do you train a decision tree ?

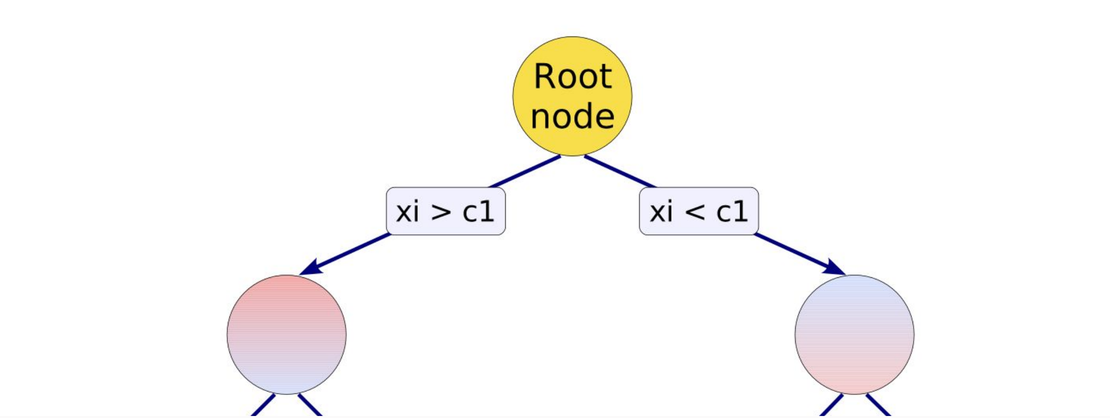
Take all features

↓

Compare their separating power

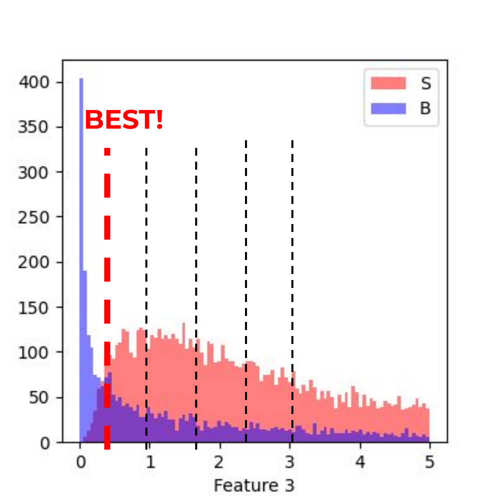
↓

Find best splitting



- *Gini Index* [default], defined by $p \cdot (1-p)$;
- *Cross entropy*, defined by $-p \cdot \ln(p) - (1-p) \cdot \ln(1-p)$;
- *Misclassification error*, defined by $1 - \max(p, 1-p)$;
- *Statistical significance*, defined by $S/\sqrt{S+B}$;

Several separation criteria are usually possible, mostly equivalent. Here p = purity (0.5 : fully mixed samples, 0 : samples composed of one class only)



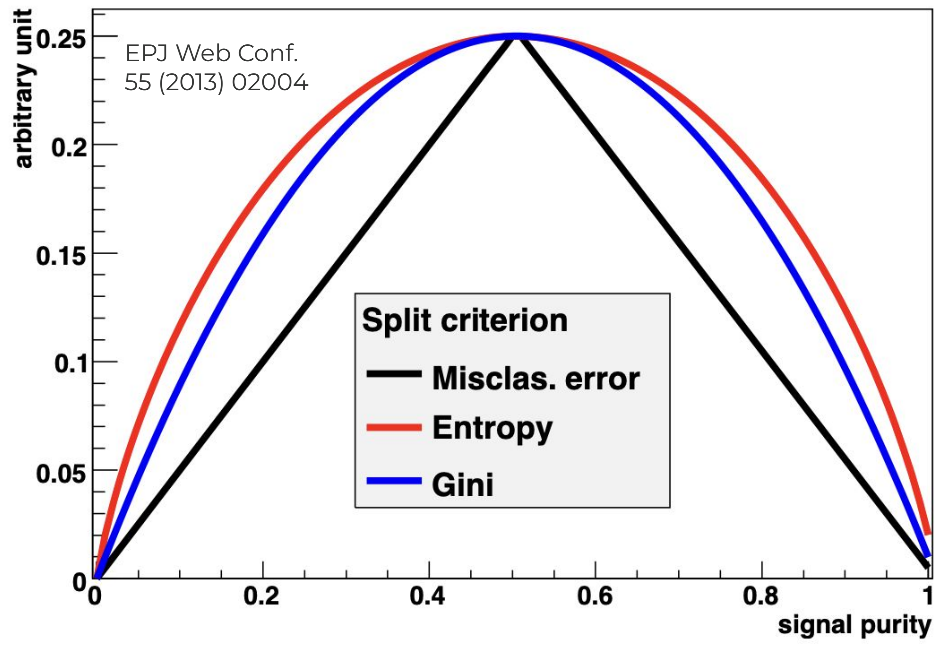
How do you train a decision tree ?

Take all features

Compare their separating power

Find best splitting

- *Gini Index* [default], defined by $p \cdot (1-p)$;
- *Cross entropy*, defined by $-p \cdot \ln(p) - (1-p) \cdot \ln(1-p)$;
- *Misclassification error*, defined by $1 - \max(p, 1-p)$;
- *Statistical significance*, defined by $S/\sqrt{S+B}$;

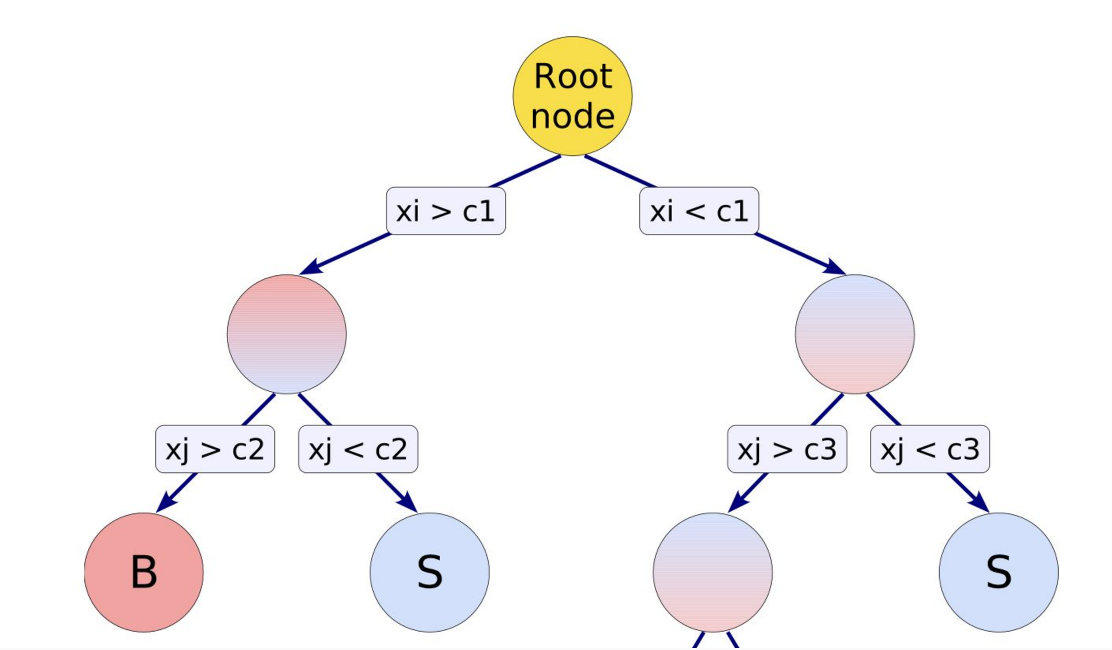


All these values are minimal if a node is dominated by either signal or background

How do you train a decision tree ?

... repeat ...

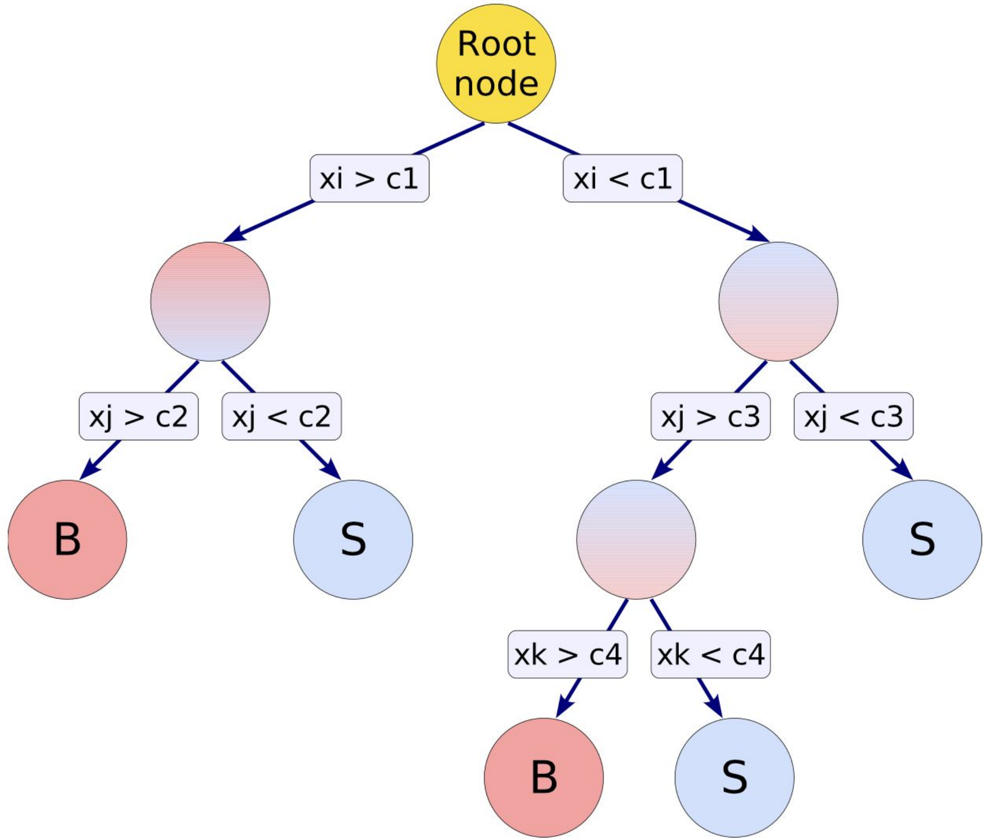
... repeat ...



How do you train a decision tree ?

Until some criterion is reached

- max tree depth
- min nr of entries in a leaf



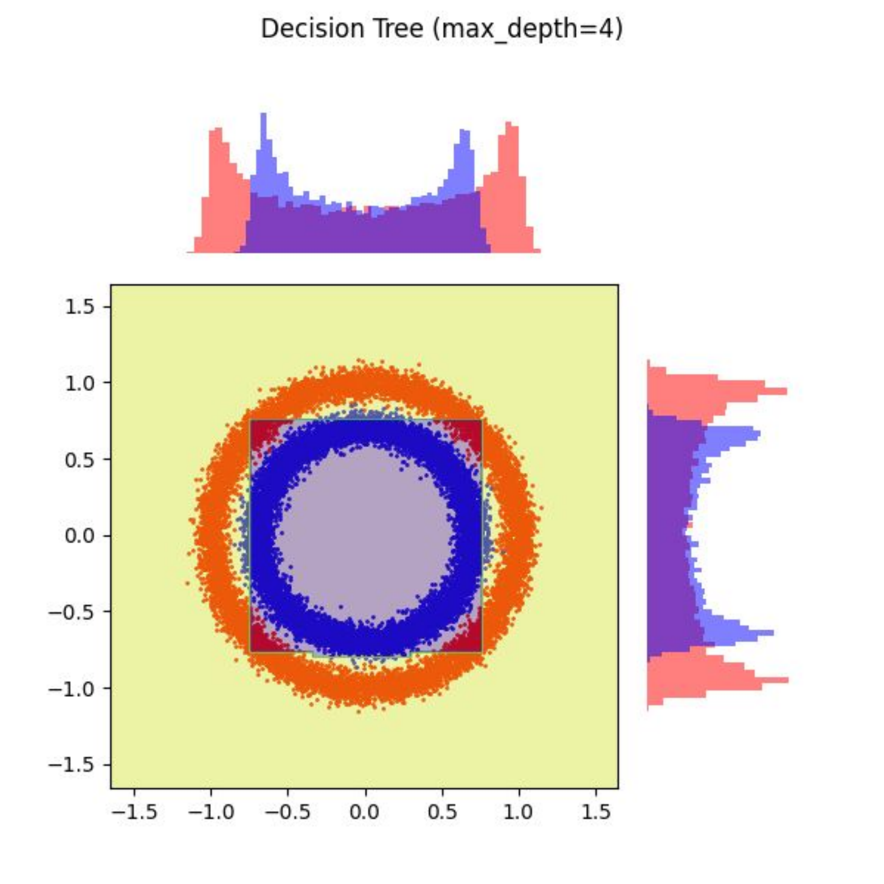
NOTE : this generalizes to multi-class categorization by computing a combined Gini index : $G = 1 - \Sigma (p\_i)^2$

Adantages and limitations of Decision Trees

- Advantages
  - Minimal/no data preprocessing needed
  - Very straightforward interpretation of the decision
  - Non linear : capable of learning complex correlations

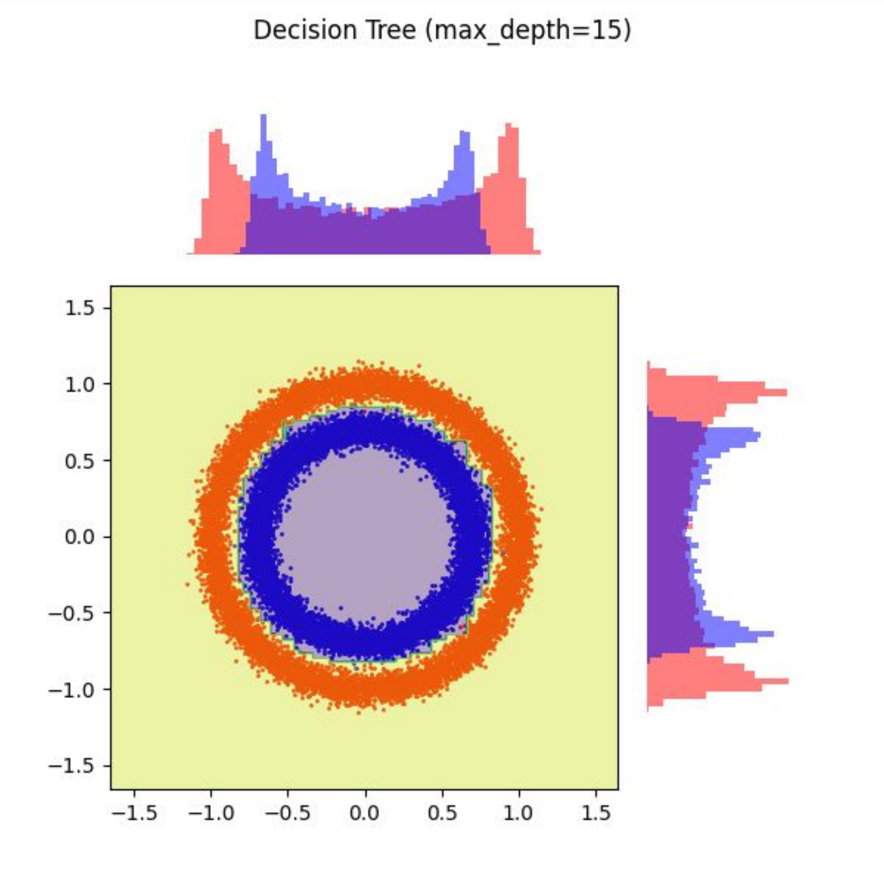
- Limitations
  - small changes in data can induce large variations in structure (different splitting affect deeply the tree)
  - contours are hypercubes (squares) : can struggle to detect linear behaviours
  - some care needed on imbalanced datasets
  - **prone to overfitting!**

Limitations of Decision Trees :overfitting



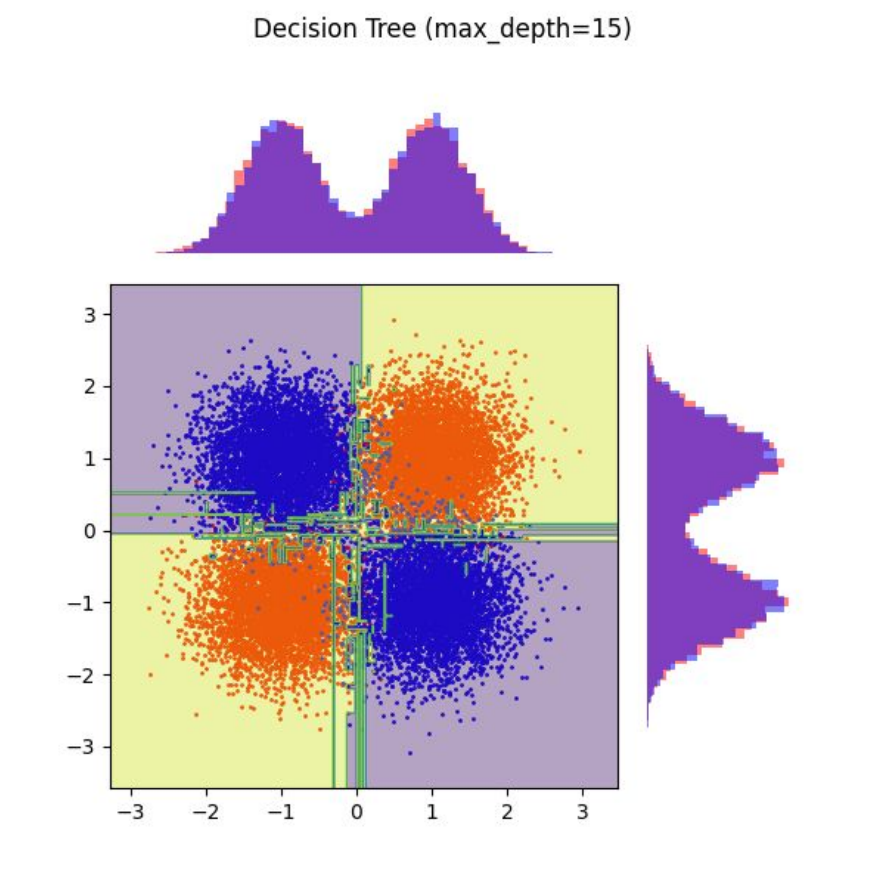
- Shallow trees cannot learn very complex features

Limitations of Decision Trees :overfitting



- Shallow trees cannot learn very complex features
- We can make deep trees…

Limitations of Decision Trees :overfitting



- Shallow trees cannot learn very complex features
- We can make deep trees…
- … but they become very sensitive to outliers
  - called overfitting : we will see that more in detail later
  - in this example, the decision tree is picking up on individual events, generalization is very poor at the frontier
- So we have conflicting issues
  - weak learner : poor performance on complex data
  - strong learner : prone to overfitting

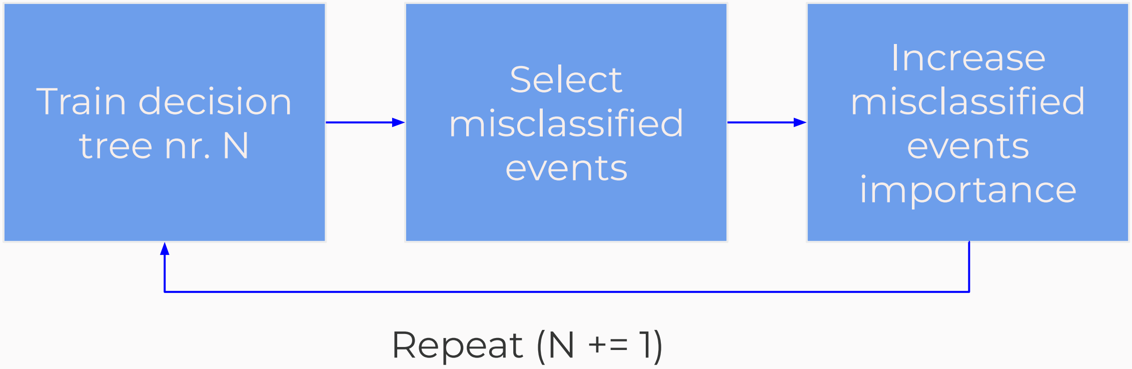
With boosting we combine both!

Boosting

What is it, and why is it needed?

Boosting : the basic idea

- Boosting = combining several weak learners into a powerful one
- Training done with an iterative approach



- Each tree benefits from what the previous one has learned and corrects the classification
- The final score comes from a weighted majority vote
- The strategy to increase the importance and to combine the votes corresponds to various trees

Boosting : the basic idea

Decision trees (weak learners)

Boosted decision trees (strong learners)

AdaBoost

Y. Freund and R.E. Schapire, J. of Computer and System Science 55, 119 (1997)

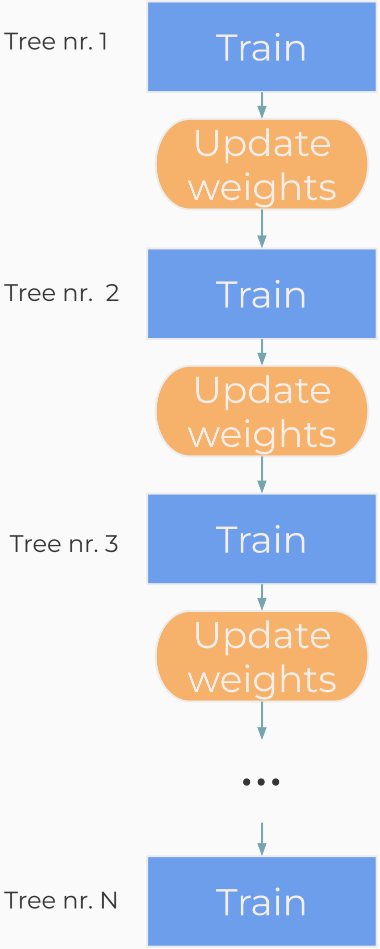
- Simplest boosting strategy : at every iteration n we increase the weight of events wrongly classified
  - compute the the misclassification rate “err” of tree n-1
  - multiply the weight of wrongly classified event by the boost weight α

$$\alpha = \frac{1 - \mathrm{err}}{\mathrm{err}}$$

  - rescale all sample weights to preserve a constant sum(w

- Final BDT decision is given by a weighted sum of each individual classifier decision h_i(x) (where e.g. 0 : bkg, 1 : signal)
  - by construction yBoost is a function between 0 and 1

$$y_{\mathrm{Boost}}(\mathbf{x}) = \frac{1}{N_{\mathrm{collection}}} \cdot \sum_{i}^{N_{\mathrm{collection}}} \ln(\alpha_i) \cdot h_i(\mathbf{x})$$



AdaBoost : learning rate, limitations

- AdaBoost performs better with ensembles of weak classifiers (max depth of 2 or 3) because of the reduced risks of overfitting
- Performance can be improved by increasing the number of classifiers and slowing down the learning rate
- Achieved with a **learning rate parameter** β by replacing $\alpha \rightarrow \alpha^{\beta}$
- The value of β impacts how quickly the weights change for a given error, and how much importance is given to each tree in the final decision
  - $\beta < 1$: slower convergence, better generalization
  - $\beta > 1$: faster convergence, risk of overfitting

- AdaBoost main limitation is the sensitivity to outliers, since misclassified events are given ever (exponentially) increasing weight ⇒ mostly limited to generalize on noisy data

Boosting revisited

- We can generalize the classification problem as a minimization problem
- The target is to **minimize the loss function L** = function that encodes how far is the prediction from the target
- Remember that a BDT decision F is a series of Decision Trees f, each depending on a series of parameters a_m:

$$F(\mathbf{x}; P) = \sum_{m=0}^{M} \beta_m f(x; a_m); \qquad P \in \{\beta_m; a_m\}_0^M$$

- ⇒ At each step of the boosting procedure, when we add a new tree f_m to the series, we can optimize its parameters a_m so that it goes in the direction of minimizing the loss
- How to understand how to optimally grow each tree? → **gradient boosting**

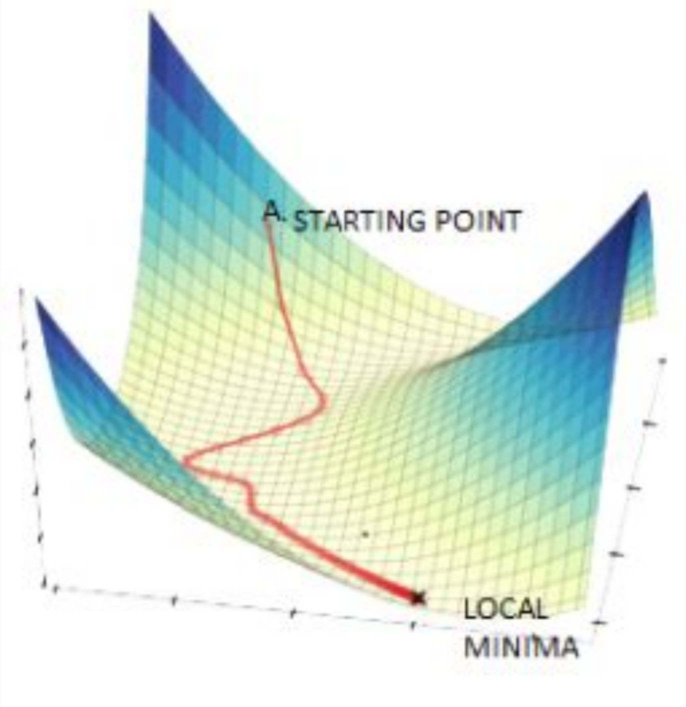
Gradient

- The gradient gives the direction and magnitude of the maximal change in a function F

- If we follow this direction, we get as quickly as possible to the minimum

- ⇒ **efficient minimization criterion** : move by steps prop. to -grad

$$\nabla F = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \cdots, \frac{\partial F}{\partial x_N}\right]$$



Gradient

- The gradient gives the direction and magnitude of the maximal change in a function F

- If we follow this direction, we get as quickly as possible to the minimum

- ⇒ **efficient minimization criterion** : move by steps prop. to -grad

- The choice of the step during the descent (learning rate) is a compromise
  - too small : slow convergence, may get stuck in a local minimum
  - too big : no convergence (jumping around the minimum)

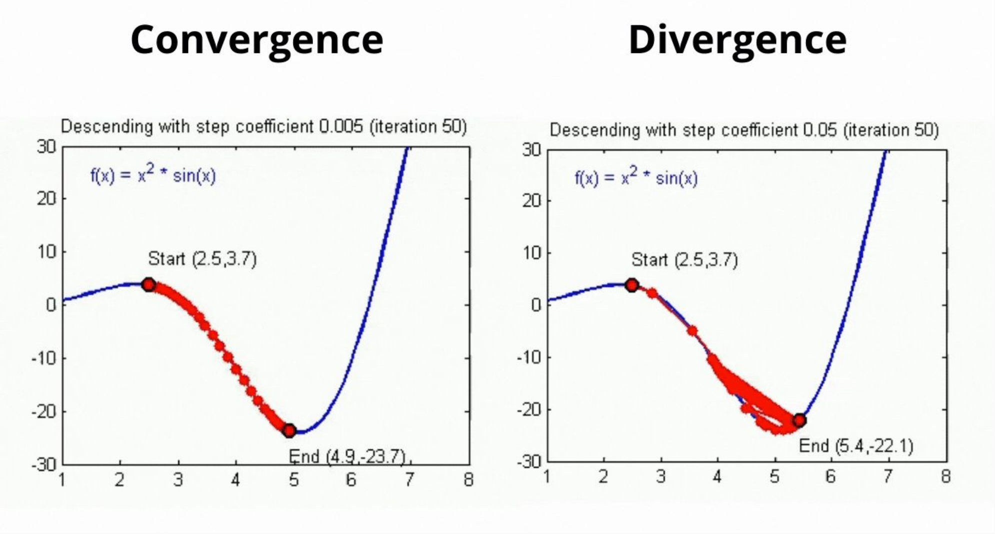


Illustration from this article

Gradient boosting

- If we take L(F, y) = exp(-F(x) y), we fall back to AdaBoost
- But with this method we can choose a loss functions that is more robust against outliers. Standard choices:

$$L(F, y) = (F(\mathbf{x}) - y)^2$$

Mean squared error

$$L(F, y) = \ln\left(1 + e^{-2F(\mathbf{x})y}\right)$$

Binomial log-likelihood loss

- In BDT boosting we take an additive, sequential approach. At each step n
  - fix the tree structure learned until step n-1
  - compute the gradient of the loss (~residuals)
  - train a new tree to learn this residual contribution
  - add the nth tree to the sequence, with a weight given by the learning rate

Gradient boosting

- Calculate the gradient of the loss at the mth iteration:

$$\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)} = g_{im}$$

- Train a tree to learn g_{im}
- Add this tree to the list, with a weight η that is the learning rate. T_m is expected to learn as well as possible g_m

$$f_m = f_{m-1} + T_m = f_{m-1} - \eta_m g_m$$

Simple example of gradient boosting

- Let's take the simple case of mean squared error : $L = ½ \Sigma (y_i - F(x_i))^2$
- In this case the derivative of the loss is :

$$-\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right] = -g_{im} = (y_i - f(x_i))$$

  which is simply the residual of our prediction
- At each iteration the new tree learns the residuals and corrects the previous prediction, converging towards zero residuals

The problem of overfitting

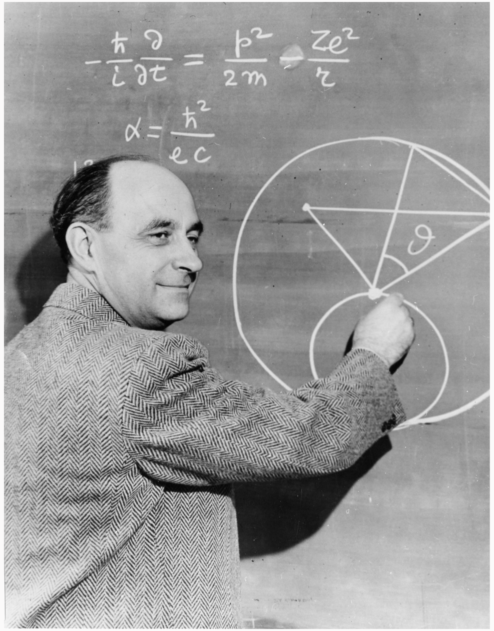
Bagging and random forests

Overfitting

- BDTs are complex functions with many parameters
  - approx. Ntrees x Nnodes : can easily be O(100-1000)
- Results in a lot of freedom to the function that a BDT learns to model the input data

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."

Enrico Fermi in 1953



F. Dyson et al., “A meeting with Enrico Fermi,” Nature, vol. 427, no. 6972, pp. 297–297, 2004

Overfitting

- BDTs are complex functions with many parameters
  - approx. Ntrees x Nnodes : can easily be O(100-1000)
- Results in a lot of freedom to the function that a BDT learns to model the input data

This is actually possible... *See :*

https://en.wikipedia.org/wiki/Von_Neumann%27s_elephant

https://arxiv.org/html/2407.07909v1

J. Mayer, K. Khairy, and J. Howard, “Drawing an elephant with four complex parameters,” American Journal of Physics, vol. 78, no. 6, pp 648–649, Jun. 2010.

Overfitting

- **Overfitting = learning specific statistical fluctuations of the training sample**
  - the method does not generalize well

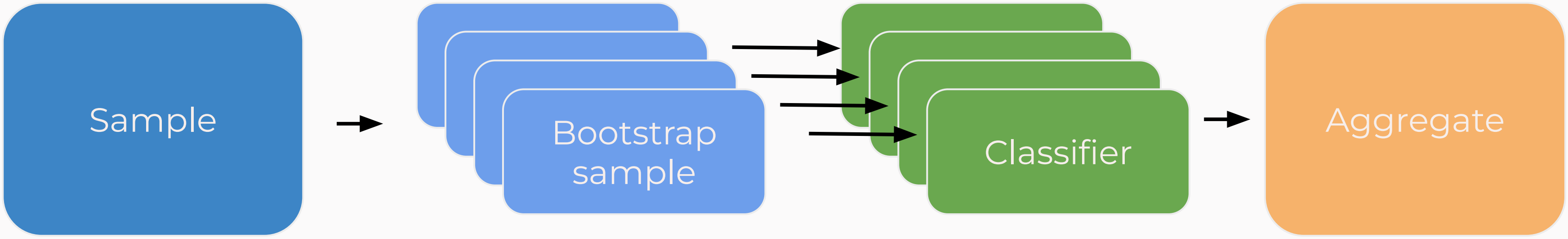
- Major issue for machine learning in general! BDTs are no exception

- Methods exist to control and mitigate overfitting in BDTs

Bagging and Random Forests

L. Breiman, “Bagging Predictors”, Machine Learning, 24 (2), 123–140, 1996
L. Breiman, “Random forests”, Machine Learning, 45 (1), 5–32, 2001.

- With **Bagging** (Bootstrap AGGregatING) we train trees on different bootstrap samples



- Statistical fluctuations are smoothed out, since outliers may not take part to the training of the Nth tree
- The final decision is taken as a majority vote or average from all classifiers

- With **random forests** we take a step ahead and add even more randomization at each node splitting by considering a random subset of the input features

Link to Notebook (Classification)

https://colab.research.google.com/drive/1Kx3rEsHhSBfXSpI3h9bx3af3UrOxFfj0?usp=sharing

Regression

Regression with trees

- Classification = predict a categorical feature (e.g. class : 0,1,2,...)
- Regression = predict a continuous feature

- A regression tree can be built as a decision tree, but replacing the node purity criterion by a standard deviation
  - we try to group together events with similar y

$$1/N \cdot \sum^{N} (y - \hat{y})^2$$

- When the node splitting (based on the regression features x_i) ends, the prediction in a leaf is computed as the averaged of all the target values in that leaf:

$$\hat{y}_{leaf} = \frac{1}{N} \sum y_i$$

Boosting regression trees

- For boosting, we can follow the same ideas as for decision trees, but updating the loss function

- AdaBoost is replaced by AdaBoost.R2 . A loss L per event is computed, the quantity

  $$\beta_{(i)} = \langle L \rangle^{(i)} / (1 - \langle L \rangle^{(i)}),$$

  is derived iand the weights are updated by

  $$w^{(i+1)}(k) = w^{(i)}(k) \cdot \beta_{(i)}^{1 - L^{(i)}(k)}$$

- For Gradient Boost, we use the Huber loss function
  - switches from a quadratic to a linear error as error gets bigger : less sensitive to outliers

$$Linear: \quad L(k) = \frac{|y(k) - \hat{y}(k)|}{\max_{\text{events } k'} (|y(k') - \hat{y}(k')|)},$$

$$Square: \quad L(k) = \left[\frac{|y(k) - \hat{y}(k)|}{\max_{\text{events } k'} (|y(k') - \hat{y}(k')|)} \right]^2,$$

$$Exponential: \quad L(k) = 1 - \exp\left[- \frac{|y(k) - \hat{y}(k)|}{\max_{\text{events } k'} (|y(k') - \hat{y}(k')|)} \right].$$

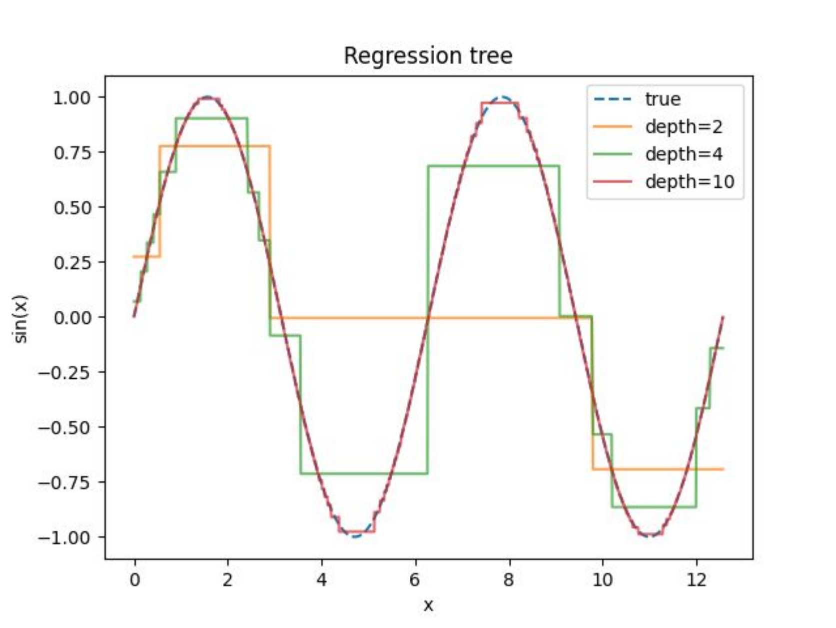
Possible loss functions for AdaBoost.R2

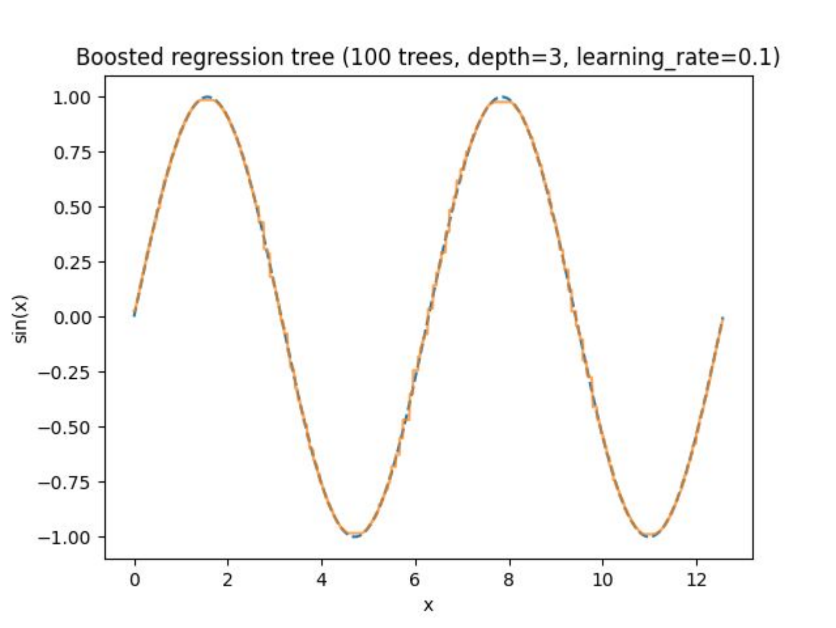
$$L(F, y) = \begin{cases} \frac{1}{2}(y - F(\mathbf{x}))^2 & |y - F| \le \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

Huber loss function

A simple regression example

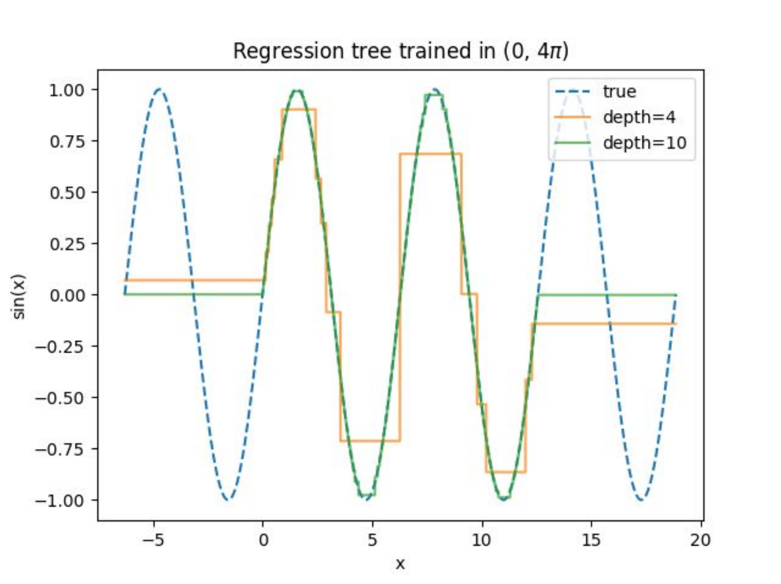
- We learn the function sin(x) using a single feature “x”
- A decision tree needs to be deep to properly model the curve
- A BDT can learn the curve by boosting shallow regressors

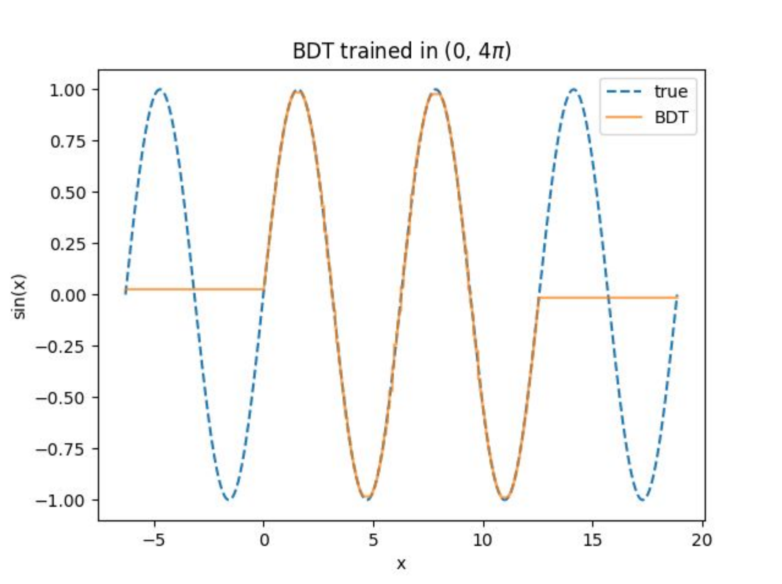




A final word of caution

- Regression trees (and decision trees) **cannot extrapolate**!
- Always make sure that your training and your inference domains are consistent to get reliable predictions





Link to Notebook (Regression)

https://colab.research.google.com/drive/1cWHsJpYF9fIBoj37IxBayB1bKJ_-l4vk?usp=sharing

Practice

Applications to real data sets

The Iris dataset

https://colab.research.google.com/drive/1lY7fyQyFI7_-I03y_XoNFetkqO8OxJNF?usp=sharing

Recap, quiz, resources

Resources

- Overview of BDT and boosting/bagging from CERN ROOT TMVA manual : https://root.cern.ch/download/doc/tmva/TMVAUsersGuide.pdf
- Fermilab introduction talk on BDTs : https://indico.fnal.gov/event/15356/contributions/31377/attachments/19671/24560/DecisionTrees.pdf
- XGBoost paper : https://arxiv.org/pdf/1603.02754
- XGBoost introduction to boosted trees : https://xgboost.readthedocs.io/en/stable/tutorials/model.html
- Boosted Decision Trees : basics and applications : https://inspirehep.net/files/7e6542b5c90f3616cb63675046c9380a