Decision trees (and ensembles of)

Alberto TONDA, Senior Researcher (DR)

UMR 518 MIA-PS (Applied Mathematics and Computer Science)
INRAE, AgroParisTech, Université Paris-Saclay
Institut des Systèmes Complexes, Paris-Ile-de-France

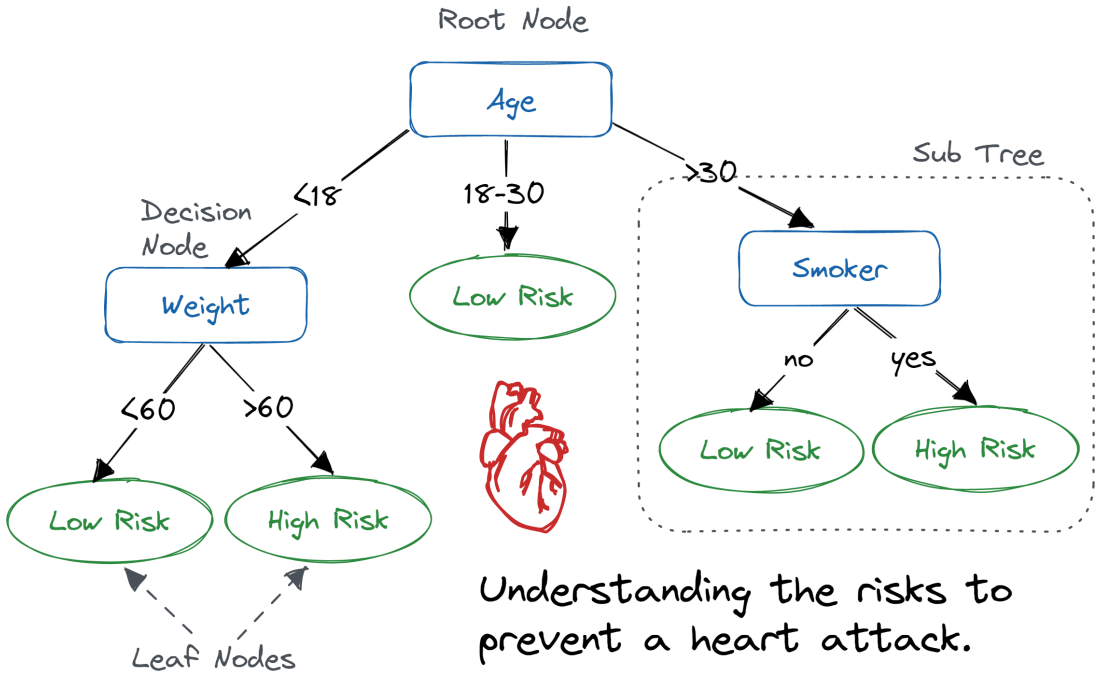
Outline

- Decision trees
- Classification and regression trees
- Optimization algorithm
- Ensembles
- Random forest
- Boosting

Decision trees

- Decision trees are one of the simplest kinds of AI algorithms
  - *Easy to read** for humans, we can follow the decision process
  - Can be hand-crafted! Encode expertise
  - Or created from data (ML)

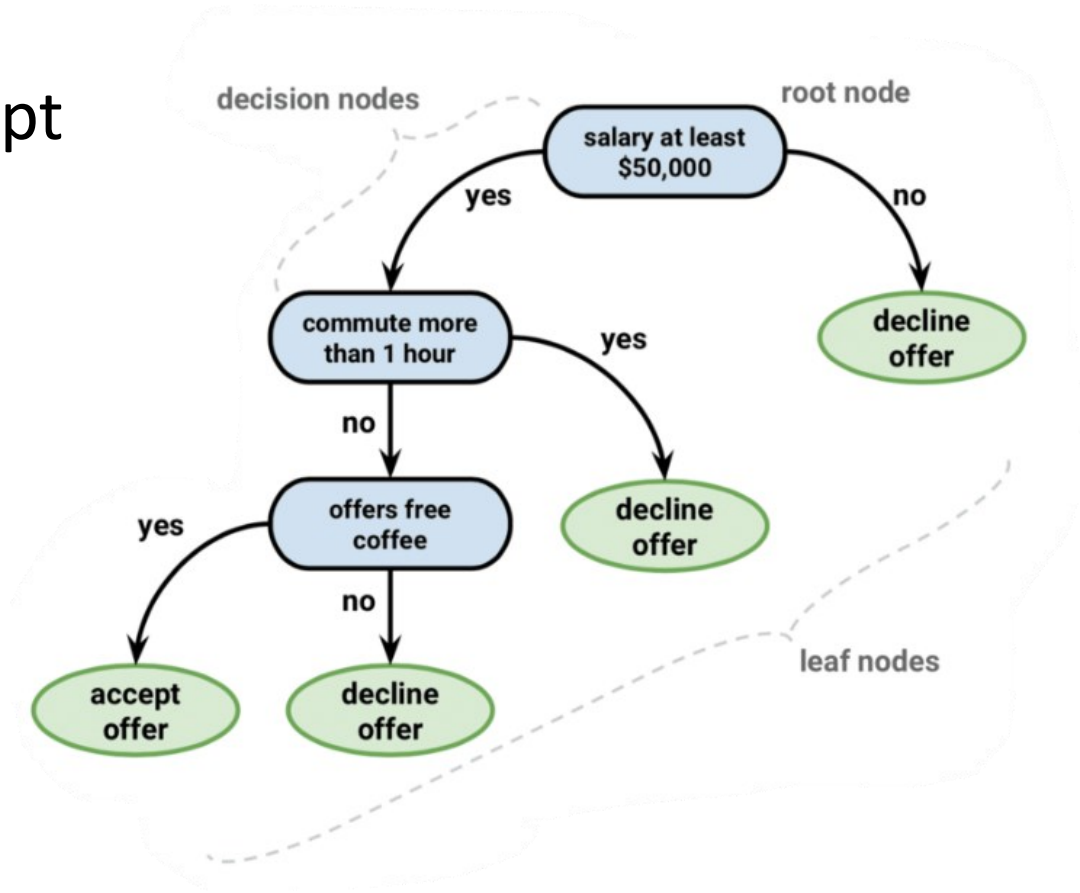
- CARTs
  - Classification and Regression Trees



**for small trees*

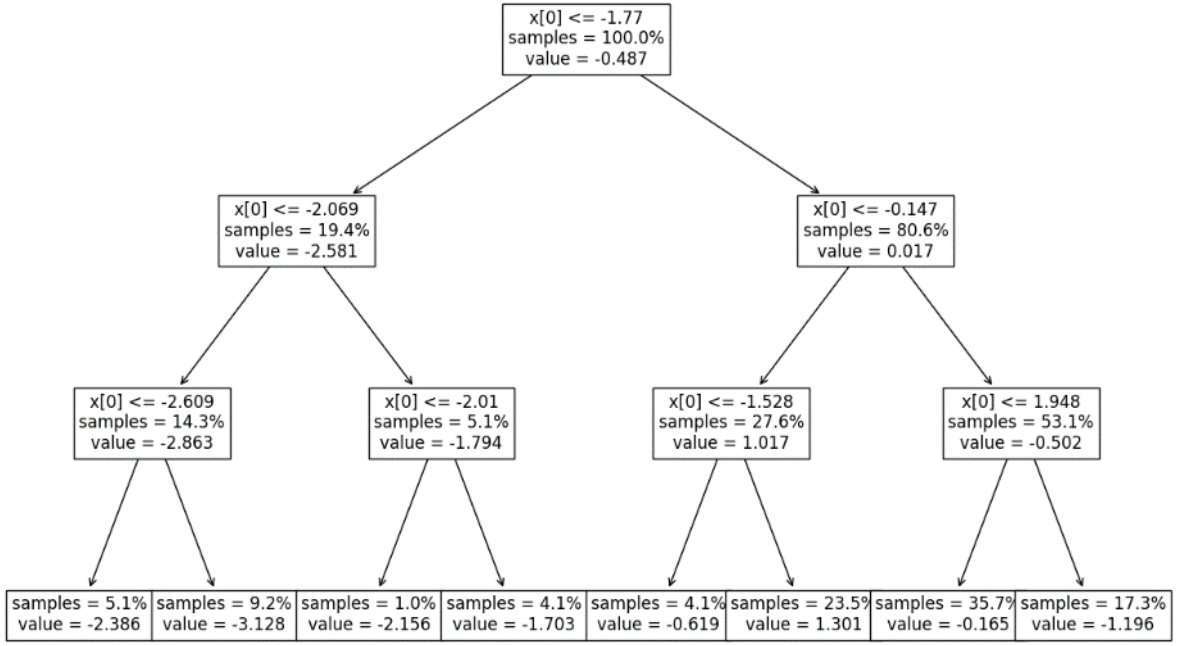
Decision trees

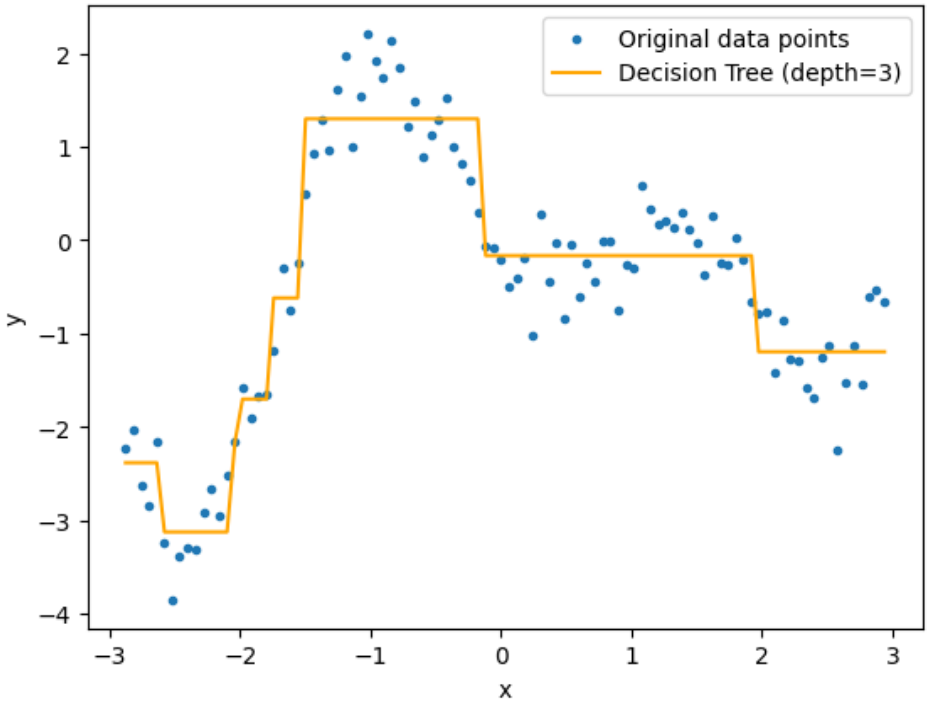
- Classification trees: leaves are CLASSES
  - Assign a sample to a “category”
  - For each job offer, should I accept or decline it?



Decision trees

- Regression trees: leaves are VALUES
  - Hypothesis: function can be approximated by linear segments
  - Each leaf is in the form $f(x) = k$ with k a floating-point value





Decision trees: optimization algorithm

- Can you guess the optimization algorithm used for CARTs?

Decision trees: optimization algorithm

- It's *greedy*!
  - At each split, **exhaustive search**
  - Over all features and values
  - "Best split" (n values, n+1 splits)
  - Quality is measured by **purity** (samples belonging to the same class) or **mean squared error**
  - Leaves are **majority class** or **mean value**
- Does not guarantee best tree

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

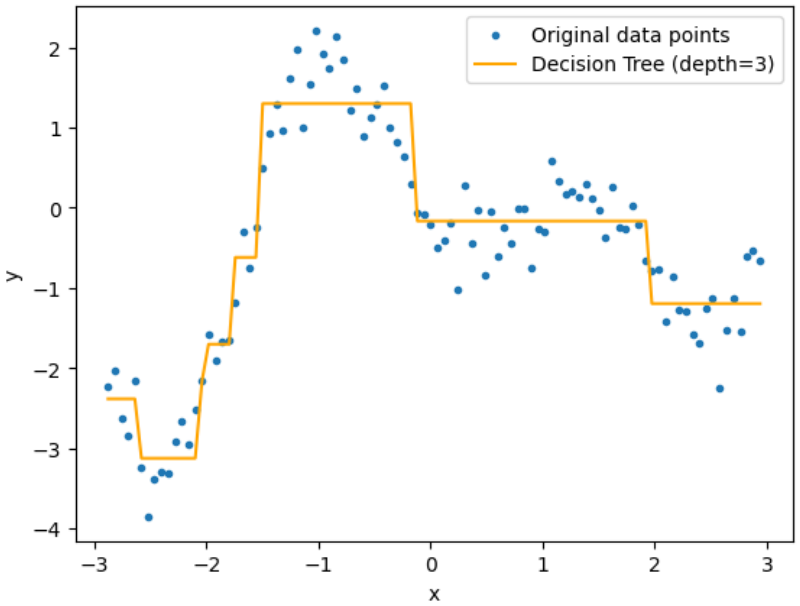
Output: A decision tree.

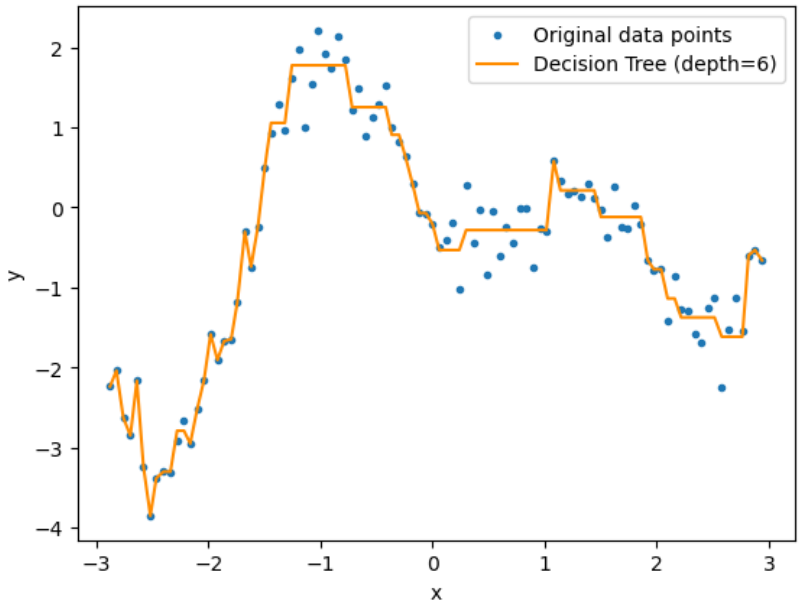
Method:

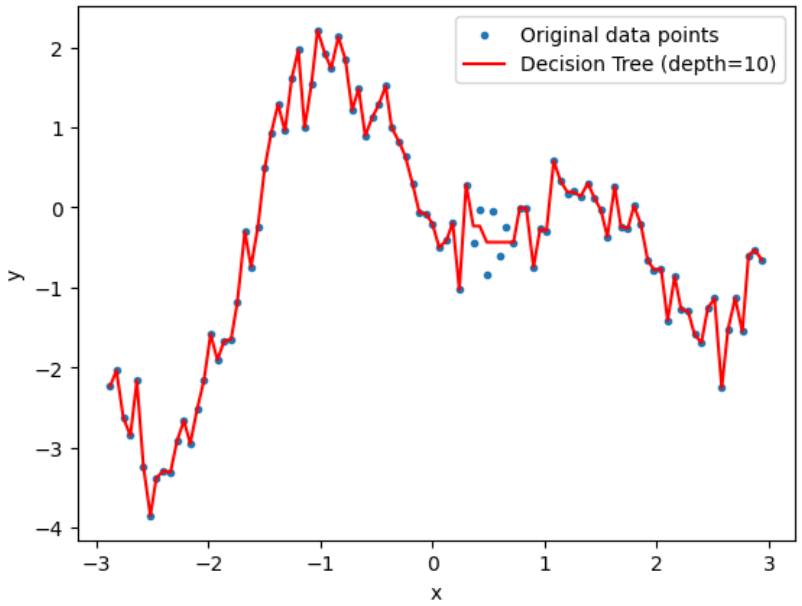
```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Decision trees

- Several hyperparameters
  - Maximum depth of the tree
  - Type of criterion used to evaluate purity of the splits







Decision trees

- For the moment, they seem *fantastic*! **Any drawbacks**?

Decision trees

- For the moment, they seem *fantastic*! **Any drawbacks**?
- Yep, they are pretty terrible at predictions

Ensembles

- A single decision tree is a *weak predictor*
  - "Performs slightly better than random guessing" (Freund & Schapire, 1995)
  - Weak also implies **low capacity** (underfitting)
  - …but then we don't have to worry too much about overfitting!
- What if used an *ensemble* of trees?
  - They can collectively vote for a class (majority vote)
  - Or just average their prediction in the case of regression
  - But algorithm is deterministic, how do we create *different trees*?

Random forest

- Randomly “hide” some features and samples
  - Each tree only sees a part of the training data
  - All trees can be generated in parallel
  - Number of trees is hyperparameter (plus all of CART)
  - The more trees, the more capacity

Boosting

- Family of iterative algorithms for the creation of ensembles
  - Gradient boosting, Adapting Boosting, eXtreme gradient boosting
  - Share similar intuition, different implementation
  - Used mainly with trees (CARTs), but in principle, any predictor

- **Sequentially** create predictors to compensate weaknesses
  - Start with one predictor, evaluate performance
  - New predictors will try to fit samples where ensemble is weak
  - For example, error metric as weighted sum of performance on samples, change weights of samples depending on performance

Ensembles

- In practice, ensembles perform well on **tabular data**
- Win most Kaggle challenges
- Between *a few* and *a lot* of hyperparameters

But…why?

- Decision Trees
  - First appearance, 1936; algorithms during the 1970s-1990s
- Ensembles
  - Random Forest, 2001; Boosting, 1995; Gradient boosting, 2001
- Publication from 2024 (!)

Statistics > Machine Learning

[Submitted on 2 Feb 2024]

Why do Random Forests Work? Understanding Tree Ensembles as Self-Regularizing Adaptive Smoothers

Alicia Curth, Alan Jeffares, Mihaela van der Schaar

Despite their remarkable effectiveness and broad application, the drivers of success underlying ensembles of trees are still not fully understood. In this paper, we highlight how interpreting tree ensembles as adaptive and self-regularizing smoothers can provide new intuition and deeper insight to this topic. We use this perspective to show that, when studied as smoothers, randomized tree ensembles not only make predictions that are quantifiably more smooth than the predictions of the individual trees they consist of, but also further regulate their smoothness at test-time based on the dissimilarity between testing and training inputs. First, we use this insight to revisit, refine and reconcile two recent explanations of forest success by providing a new way of quantifying the

Questions?

Bibliography
- Kochenderfer & Wheeler, *Algorithms for Optimization*, MIT Press, 2019
- Grinsztajn et al., *Why do tree-based models still outperform deep learning on tabular data?*, 2022
- McElfresh et al., *When Do Neural Nets Outperform Boosted Trees on Tabular Data?*, 2023
- Decision Tree and Random Forest - History of Data Science