UNIVERSITÄT BONN

Decision Trees and Random Forests

Elena Trunz

September 19, 2023

Uni Bonn

Overview

Introduction

Decision Trees

Bagging

Random Forests

Introduction

Is your date evil?

Our experience with evil and good

	mask	cape	tie	ears	smokes	height	class
Batman	+	+	-	+	-	180	good
Robin	+	+	-	-	-	176	good
Alfred	-	-	+	-	-	185	good
Penguin	-	-	+	-	+	140	evil
Catwoman	+	-	-	+	-	170	evil
Joker	-	-	-	-	-	179	evil
Batgirl	+	+	-	+	-	165	?
Riddler	+	-	-	-	-	182	?
Your date	-	+	+	+	+	181	?

The idea for this dataset was taken from the Cornell course CS4780 SP17.

Tree-based method

Solutions to machine learning tasks are called hypotheses

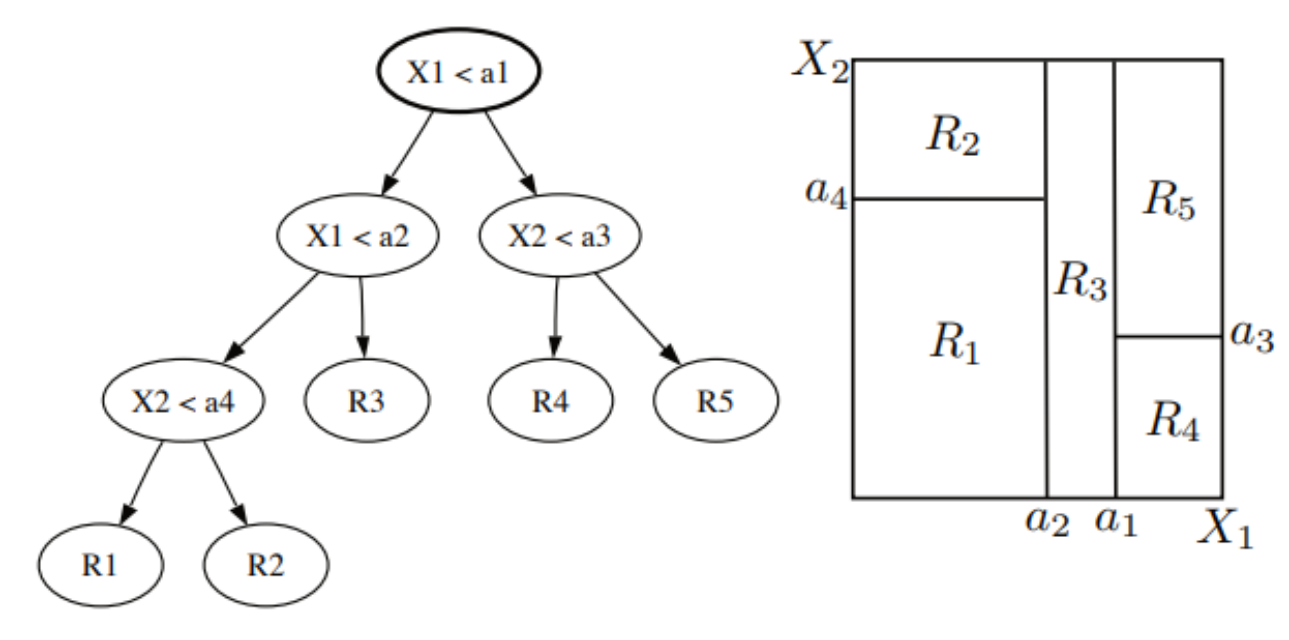
Hypotheses have to be represented in some representation scheme

- Last time: hyperplane representation
- Today: tree representation

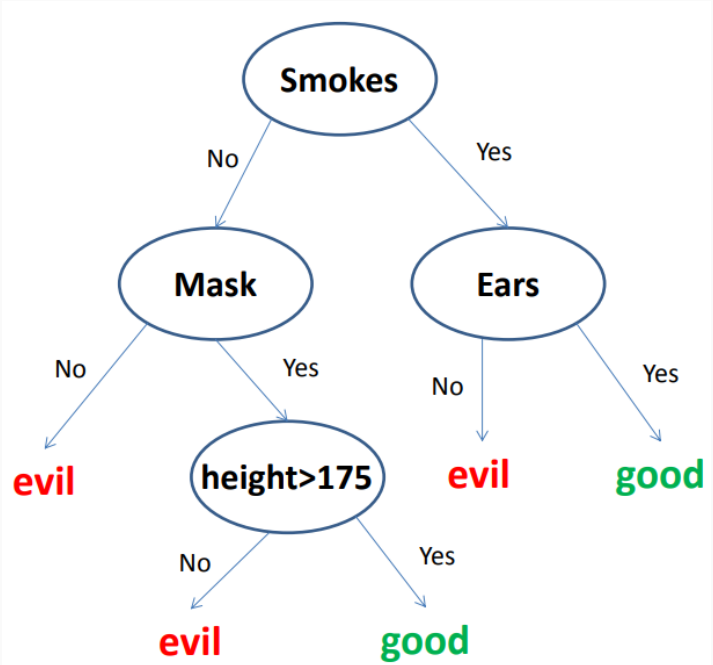
Decision Trees

Decision tree

- Tree representation of a partition of the feature space
- Each interior node corresponds to a question related to the features.
  - Numerical question: $X \leq a$, for a feature X and some threshold a
  - Categorical question: $X \in \{blue, white, red\}$
- Each leaf is labeled with a target label y



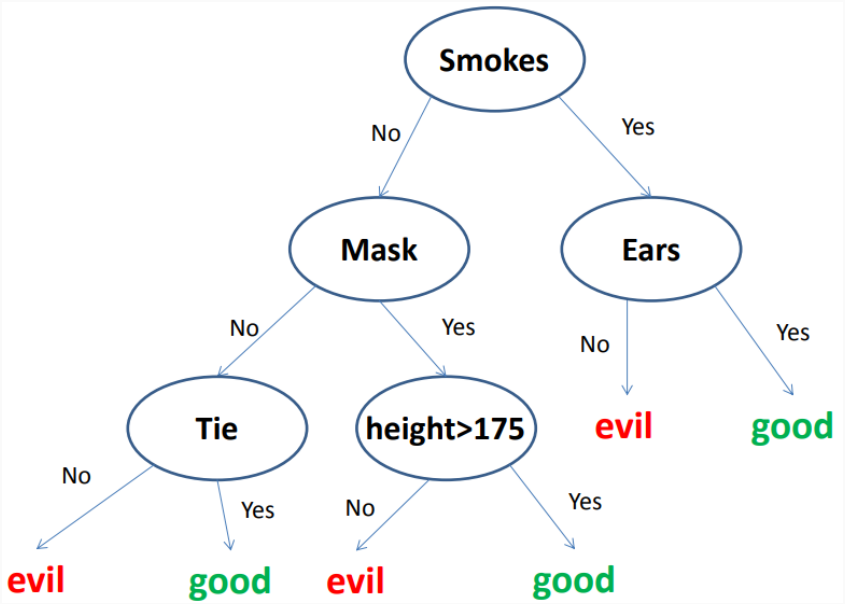
Is this decision tree consistent?



	mask	cape	tie	ears	smokes	height	class
Batman	+	+	-	+	-	180	good
Robin	+	+	-	-	-	176	good
Alfred	-	-	+	-	-	185	good
Penguin	-	-	+	-	+	140	evil
Catwoman	+	-	-	+	-	170	evil
Joker	-	-	-	-	-	179	evil
Batgirl	+	+	-	+	-	165	?
Riddler	+	-	-	-	-	182	?
Your date	-	+	+	+	+	181	?

What can we do to make the tree consistent?

Splitting makes a decision tree consistent



	mask	cape	tie	ears	smokes	height	class
Batman	+	+	-	+	-	180	good
Robin	+	+	-	-	-	176	good
Alfred	-	-	+	-	-	185	good
Penguin	-	-	+	-	+	140	evil
Catwoman	+	-	-	+	-	170	evil
Joker	-	-	-	-	-	179	evil
Batgirl	+	+	-	+	-	165	?
Riddler	+	-	-	-	-	182	?
Your date	-	+	+	+	+	181	?

To split or not to split

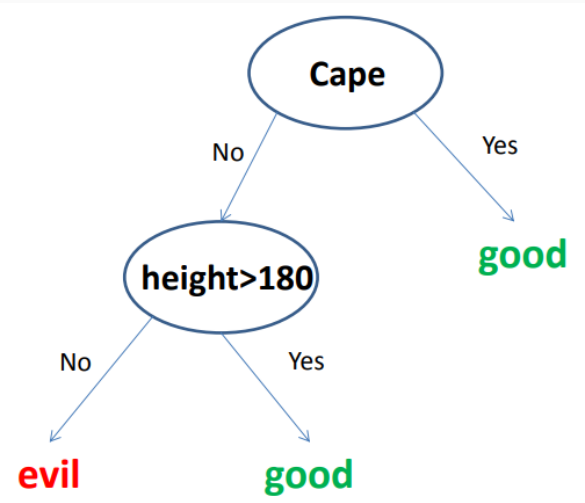
- Is it always a good idea to split?
- What is bias and variance of a decision tree as a function of depth?
  - $\rightarrow$ Low depth - high bias, high depth - high variance!
- What we ideally want to have, is the smallest tree that fits the data.
- Problem of finding the smallest decision tree with the smallest error is NP-hard. ☹

What is the smallest tree here?

	mask	cape	tie	ears	smokes	height	class
Batman	+	+	-	+	-	180	good
Robin	+	+	-	-	-	176	good
Alfred	-	-	+	-	-	185	good
Penguin	-	-	+	-	+	140	evil
Catwoman	+	-	-	+	-	170	evil
Joker	-	-	-	-	-	179	evil
Batgirl	+	+	-	+	-	165	?
Riddler	+	-	-	-	-	182	?
Your date	-	+	+	+	+	181	?

What is the smallest tree here?

	mask	cape	tie	ears	smokes	height	class
Batman	+	+	-	+	-	180	good
Robin	+	+	-	-	-	176	good
Alfred	-	-	+	-	-	185	good
Penguin	-	-	+	-	+	140	evil
Catwoman	+	-	-	+	-	170	evil
Joker	-	-	-	-	-	179	evil
Batgirl	+	+	-	+	-	165	?
Riddler	+	-	-	-	-	182	?
Your date	-	+	+	+	+	181	?



Greedy decision trees (ID3)

Greedy idea: optimize each step, step by step

Greedy idea for decision tree learning:

1. Decide, which loss function will describe the effect of a split
2. Start with the root node
3. Try all features and all possible splits
4. Pick the split that minimizes the current loss
5. Repeat from step 3

Effect of a split

- We want our leafs to be pure: every point in a leaf should have the same label.

$\rightarrow$ We need to measure the impurity of a set of points:

  - Gini impurity
  - Entropy

Gini impurity

Let

- $S = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)\}$ be a set of data points in a leaf
- $S_k = \{(\mathbf{x}, y) \in S | y = k\}$
- The probability to pick a point with a label k is $p_k = \frac{|S_k|}{|S|}$

The Gini impurity of S is defined as

$$G(S) = \sum_{k=1}^{K} p_k(1 - p_k)$$

Entropy

The information content of an event E is defined by

$$I(E) = -log_2(p(E))$$

In our case the event E is picking a label from the set S.

Entropy measures the expected amount of information conveyed by identifying the outcome of a random trial

$$F(S) = -\sum_{k=1}^{K} p_k log_2(p_k)$$

We want to minimize the entropy, because we don't want to be surprised, we want to be certain. We want our leafs to have low information content.

Information Gain

We can compute the entropy of a node, what is the reduction in entropy of a split?

The reduction in entropy of a split (the decrease in impurity by the split) is called *Information Gain* and is given by

$$G(S, q) = F(S) - \sum_{r \in Replies(q)} \frac{|S_r|}{|S|} F(S_r)$$

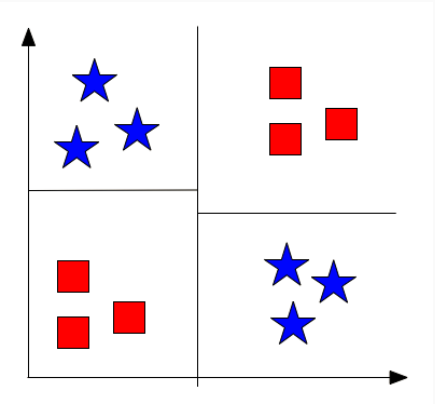
where, q is the question, S_r is the subset of S, with the answer r to the question q.

When do we stop?

- If all labels y in a set are the same: the node becomes a leaf with this label
- If all the inputs $\mathbf{x}$ are the same: the node becomes a leaf with the most frequent label
- In order to reduce the variance we can introduce stopping after a ceratain depth (or number of splits) has been reached

Why don't we stop if no split can improve impurity?

Decision trees cannot look far ahead. Example: XOR



More general decision trees

- Branching factor
  - $= 2$ (binary trees), most commonly used, due their more limited computational cost
  - > 2
- More complex node questions
  - Binary space partition trees: $\sum_{i=1}^{n} = \alpha_i X_i \leq a$
    $\rightarrow$ Partition the input space with convex polyhedral regions
  - Sphere trees: $\|\mathbf{X} - a_0\| \leq a$
    $\rightarrow$ Partition the input space with pieces of spheres

  More complex questions lead to richer partitions

  \+ Richer hypothesis sets

  \- Can cause overfitting if training database is not big enough

  \- Increase computational complexity of prediction and training

Decision trees for regression (CART)

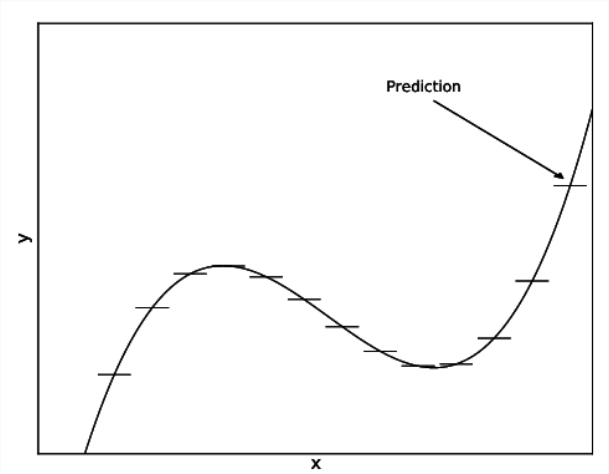
Assume the labels are continuous: $y \in \mathbb{R}$.

To measure the impurity of a set we use the square loss function:

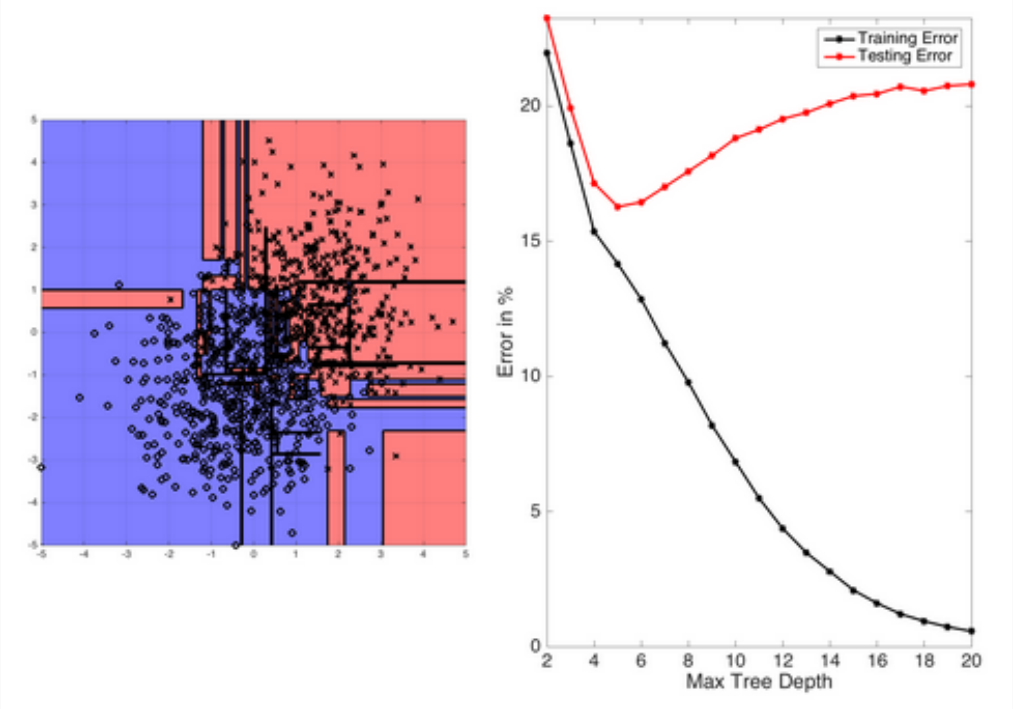
$$L(S) = \frac{1}{|S|} \sum_{(\mathbf{x},y)\in S} (y - \mu)^2$$

where $\mu = \frac{1}{|S|} \sum_{(\mathbf{x},y)\in S} y$.

The impurity of a set is measured by how close every point in the set is to the mean. $\rightarrow$ Similar to variance!



Bias and variance problems



It is hard to find the sweet spot having only the depth or the number of splits as hyperparameter.

And because those numbers are discrete it might not even be possible.

This is the reason the decision trees usually don't work very well.

Solution to the bias and variance problem

There are two effective ways to deal with these problems:

- Make the tree small $\rightarrow$ we have a bias problem $\rightarrow$ address it with *boosting*
- Make the tree very large $\rightarrow$ we have a variance problem $\rightarrow$ address it with *bagging*

Bagging

Why do large decision trees have a variance problem?

- If we keep splitting, we get every example correct
- But low level splits are very data specific
- With different datasets we get totally different classifiers
- The splits will be made in totally different dimensions

Recall: If the variance is large, that means that the difference from the expected classifier is large.

Reducing the variance

Idea:

- Take M datasets $\mathcal{D} = \{D_1, \ldots, D_M\}$
- Train a decision tree $h_{D_j}(\mathcal{X})$ on each of the dataset $D_j \in \mathcal{D}$
- Take an average as a final hypothesis

$$h(\mathcal{X}) = \frac{1}{M} \sum_{j=1}^{M} h_{D_j}(\mathcal{X})$$

- Then by the *weak low of large numbers* the following holds

$$h(\mathcal{X}) = \frac{1}{M} \sum_{j=1}^{M} h_{D_j}(\mathcal{X}) \underset{M \to \infty}{\longrightarrow} \bar{h}(\mathcal{X})$$

where $\bar{h}_D(\mathcal{X})$ be the expected classifier.

Where do we get M datasets from?

Should we divide our original database in M disjoint parts?

Better idea: bootstrapping!

We create M different datasets by sampling N points with replacement.

That way roughly 60% of data in each dataset will be the same, but the roughly 40% will be different.

Is there a problem?

Which property of the weak law of large numbers is violated?

The new datasets are not independent!

But this is not a big problem, because it is enough to make the variance term decrease.

Random Forests

Idea

1. Draw M datasets from the original dataset D.
2. For each of the new datasets train a decision tree until only pure leafs (or leafs with the same $\mathbf{x}$) remain
    - Instead of trying all splits, before each decision randomly sample only k features to compute the information gain of a split

That way we get very different decision trees, which make different mistakes.

And when we average them, the mistakes will be averaged out.

How to set k and M

- Always set $k = \lceil \sqrt{d} \rceil$, where d is number of features.
- Set M as big as you can afford.

Pros and cons of decision trees

+ Fast to train and evaluate
+ Relatively easy to interpret
- Not as good as other state-of-the-art approaches, but
  - Decision trees have clearly defined bias and variance problems
  - Address the problem of high bias with *boosting*
  - Address the problem of high variance with *bagging*
+ $\rightarrow$ Can be used as weak learners with *boosting* and *bagging* to define effective algorithms

Pros and cons of random forests

+ Can deal with imbalanced data
+ Do not really have hyperparameters, that need to be tuned
+ Can deal with missing values
+ Can deal with unpreprocessed data

$\rightarrow$ At least a second best algorithm for many problems!

Note: Bagging can be used with any machine learning algorithm that has a variance problem!