Decision Trees and Random Forests

Machine Learning 2021
UML book chapter 18
Slides P. Zanuttigh
A decision tree is a predictor that predicts the label associated with a sample $\mathbf{x}$ by traveling along a tree from the root to a leaf.

- We’ll focus on binary trees.

At each internal node we made a decision based on features of $\mathbf{x}$:

- It corresponds to splitting the input space.
- Simplest idea, split on the basis of a threshold on one of the features, i.e., $x_i < \theta$ or $x_i \geq \theta$.

Each leaf is associated to a label.
Example: Decision Tree

\[ x = [x_1 \ x_2] \]

Decision Tree Diagram:
- Feature: \( \text{feature}_x1 \)
  - \( \leq 3 \):
    - Feature: \( \text{feature}_x2 \)
      - \( \leq 1 \):
        - Class 0
      - \( > 1 \):
        - Class 1
  - \( > 3 \):
    - Class 0
Grow a Decision Tree

Consider a *binary* classification setting and assume to have a gain (performances) measure:

**Start**
- A single leaf assigning the most common of the two labels (i.e., the one of the majority of the samples)

**At each iteration**
- Analyze the effect of splitting a leaf
- Among all possible splits select the one leading to a larger gain and split that leaf (or choose not to split)
Iterative Dichotomizer 3 (ID3)

INPUT: training set $S$, feature subset $A \subseteq [d]$

- if all examples in $S$ are labeled by 1, return a leaf 1
- if all examples in $S$ are labeled by 0, return a leaf 0
- if $A = \emptyset$, return a leaf whose value = majority of labels in $S$
  
else:

Let $j = \arg\max_{i \in A} \text{Gain}(S, i)$

- if all examples in $S$ have the same label
  
  Return a leaf whose value = majority of labels in $S$
  
- Let $T_1$ be the tree returned by $\text{ID3}((x, y) \in S : x_j = 1, A \setminus \{j\})$.
- Let $T_2$ be the tree returned by $\text{ID3}((x, y) \in S : x_j = 0, A \setminus \{j\})$.
  
Return the tree:

$x_j = 1$?

- $T_1$
- $T_2$

Assume binary features, i.e., $\mathcal{X} = \{0,1\}^d$

Find which split (i.e. splitting over which feature) leads to the maximum gain

$x_j$: selected feature for the split

No more features to use

Split on $x_j$ and recursively call the algorithm considering the remaining features*

* Split on a feature only once: they are binary

If real valued features: need to find threshold, can split on same feature with different thresholds
Gain Measure

Train error
- Define the gain as the decrease in the training error

Gini Index \( G = \sum_{i=1}^{C} p(i)(1 - p(i)) \)
- \( i = 1, \ldots, C \) : classes
- \( p(i) \) : probability of class \( i \)

- It is a measure of statistical dispersion in a frequency distribution
  - For binary case: \( G=0 \) if all in the same class, \( G=1/2 \) if 50% split
- Measure of variance: higher variance more misclassifications
- It measures how “pure” is the distribution after the split
- Gini Index is a smooth and concave upper bound of the train error

Threshold based splitting rules for real valued features
- Extend by creating a set of thresholds and testing all the various combination of features and thresholds
Example
Issue of ID3: The tree is typically very large with high risk of overfitting
Prune the tree to reduce its size without affecting too much the performances

Generic Tree Pruning Procedure

input:
function $f(T, m)$ (bound/estimate for the generalization error of a decision tree $T$, based on a sample of size $m$),
tree $T$.

foreach node $j$ in a bottom-up walk on $T$ (from leaves to root):
find $T'$ which minimizes $f(T', m)$, where $T'$ is any of the following:
the current tree after replacing node $j$ with a leaf 1.
the current tree after replacing node $j$ with a leaf 0.
the current tree after replacing node $j$ with its left subtree.
the current tree after replacing node $j$ with its right subtree.
the current tree.
let $T := T'$. 
Random Forests (RF)

- Introduced by Leo Breiman in 2001
- Instead of using a single large tree, construct an ensemble of simpler trees
- A Random Forest (RF) is a classifier consisting of a collection of decision trees
- The prediction is obtained by a majority voting over the prediction of the single trees

Tally: Six 1s and Three 0s
**Prediction: 1**
Random Forest: Example

Random Forest

Tree-1 -> Class-A

Tree-2 -> Class-B

Tree-n -> Class-B

Majority-Voting

Final-Class
Random Sampling with Replacement

Idea: randomly sample from a training dataset with replacement

- Assume a training set $S$ of size $m$: we can build new training sets by taking at random $m$ samples from $S$ with replacement (i.e., the same sample can be selected multiple times)
  - For example, if our training data is $[1, 2, 3, 4, 5, 6]$ then we might sample sets like $[1, 2, 2, 3, 6, 6]$, $[1, 2, 4, 4, 5, 6]$, $[1 \ 1 \ 1 \ 1 \ 1 \ 1]$, etc.....
  - i.e., all lists have a length of six but some values can be repeated in the random selection
- Notice that we are not subsetting the training data into smaller chunks
Bagging (Bootstrap Aggregation):

- Decisions trees are very sensitive to the data they are trained on: small changes to the training set can result in significantly different tree structures
- **Random forest** takes advantage of this by allowing each individual tree to randomly sample with replacement from the dataset, resulting in different training sets producing different trees
- This process is known as *bagging*
Bagging: Example

A

Training set

... sample 1

... isample N

Tree 1

... Tree N

B

Initial node (root)

Split node

Terminal node

Tree 1

Tree 2

Tree N

Majority Vote

Random Forest's decision
Randomization: Feature Randomness

- In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the largest gain.
- In contrast, each tree in a random forest can pick only from a random subset of features (Feature Randomness).
- I.e., node splitting in a random forest model is based on a random subset of features for each tree.
- This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification.