MA2823: Foundations of Machine Learning
Chapter 8: Tree-based Methods

Lecturer: Chloé-Agathe Azencott

Scribe: LIN Laurent
PHAM Olivier
SCHNITZLER Pierre-Louis

In this chapter we will :

— see how to build decision trees (and more precisely how to grow a tree and when to stop growing a tree);

— explain why they are examples of non-metric learning and hierarchical learning;

— combine decision trees (or other weak learners) to make more powerful classifiers.

1 Decision trees

1.1 Hierarchical learning

As single classifiers assign a class to an object x using a single operation and use a single set of features for all classes, they can face difficulties in case of non-metric learning, so when classes have multi-modal distributions or when features are nominal. We are talking about nominal data when the attributes are discrete without any natural notion of similarity/ordering (such as {color, shape, texture, size} for a fruit classification). In order to avoid those problems, decision trees are based on hierarchical classifiers, which make multiple successive tests.

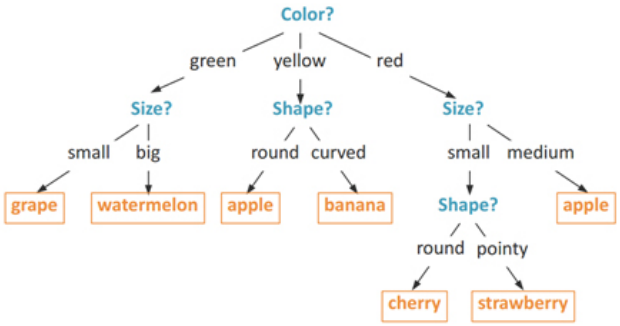

Figure 1: Example of a decision tree

1.2 Multiclass classification

There are strategies for reducing a problem of multiclass classification to multiple binary classification problems. It can be categorized into One-versus-one (Build K (K-1) / 2 classifiers and make them vote) and One-versus-all (Build K classifiers and make them vote). To solve those problems, we have to use an algorithm that naturally handles multiple classes. It is the case of tree algorithms and Neural networks (that we will study in Chap. 10).

1.3 Vocabulary Reminder

Here is a short reminder of tree vocabulary:

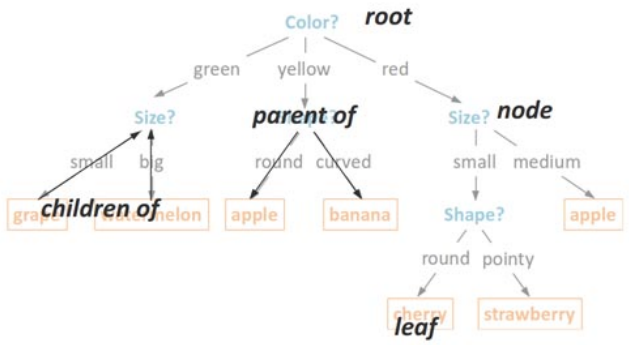

Figure 2: Tree vocabulary

1.4 Partition of the feature space

Classification and regression trees (CART) is the generic name for a recursive procedure to split a training set and organize it into a tree. The models are obtained by recursively partitioning the training set into smaller and smaller subsets and fitting a simple prediction model within each partition. As a result, the partitioning can be represented graphically as a decision tree. During the CART procedure we will have to do several choices:

— Binary or multi-way splits?

— Which feature(s) to use at each node? i.e. how to split?

— When to stop growing a tree?

2 Binary-tree versus non binary-tree

A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. A tree with arbitrary branching factor can always be equivalently represented by a binary tree.
The following binary tree is equivalent to the non-binary tree of Figure 1.

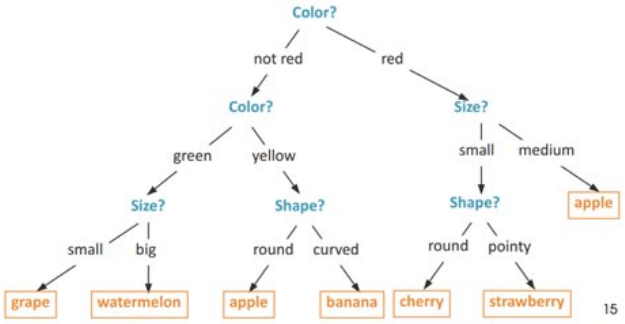

Figure 3: Binary tree equivalent to Figure. 1

3 How to grow a tree

3.1 Introduction

We are only considering monothetic trees, that is to say trees with only one feature per node. In this conditions, the decision boundary is orthogonal to the axes. The splitting variable (j) and the splitting point (s) define 2 regions:

$$R_l = \{x : x_j \leq s\} \text{ and } R_r = \{x : x_j > s\}$$

For a regression tree, we choose j and s to minimize SE (standard error):

$$\min_{j,s} \left(\sum_{i:x_i \in R_l(j,s)} (y_i - c_l)^2 + \sum_{i:x_i \in R_r(j,s)} (y_i - c_r)^2 \right)$$

For a classification tree, we choose j and s to minimize impurity :

$$\min_{j,s} \left(\frac{|R_l(j,s)|}{n_{tot}} \times \text{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n_{tot}} \times \text{Imp}(R_r(j,s)) \right)$$

3.2 Impurity

Decision trees are grown by posing a series of questions about the feature of the items. In order to do so, we add question nodes incrementally, using labeled training examples to guide the choice of questions. Ideally, a single, simple question would perfectly split the training examples into their classes. If no question exists that gives such a perfect separation, we choose a question that separates the examples as cleanly as possible.

Several measures have been designed to evaluate the degree of inhomogeneity, or impurity, in a set of items. For decision trees, the most common measures are entropy, the Gini index and the classification error. Given a measure of impurity, we choose a question that minimizes the weighted average of the impurity of the resulting children nodes.

3.3 Classification error

The classification error corresponds to the minimum probability that a training point will be misclassied at node (s,j) and is defined by the following formula :

$$\text{Imp}(R_m) = 1 - \max_k(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training instances from class k in R_m.

The value of the classification error is always between 0 and 1. If all examples from one class belong to R_m then $\text{Imp}(R_m) = 1 - \max(1) = 0$. If we have 2 balanced classes and instances are randomly split at (s, j), then $\text{Imp}(R_m) = 1 - \max(0.5, 0.5) = 0.5$

3.4 Entropy

The concept of entropy comes from the information theory. Entropy is a measurement of information (or rather lack of information). We calculate the information gain by making a split. This measures how you reduce the uncertainty about the label. It is defined by the following formula:

$$\text{Imp}(R_m) = -\sum_k \hat{p}_{mk} \log_2(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training instances from class k in R_m.

If all examples from one class belong to R_m then $\text{Imp}(R_m) = 0$. If we have 2 balanced classes and instances are randomly split at (s, j), then $\text{Imp}(R_m) = -(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})) = 1$

3.5 Gini Impurity

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

It is defined by the following formula:

$$\text{Imp}(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training instances from class k in R_m.

If all examples from one class belong to R_m then $\text{Imp}(R_m) = 0$. If we have 2 balanced classes and instances are randomly split at (s, j), then $\text{Imp}(R_m) = \frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2} = \frac{1}{2}$.

If we have two regions R_l and R_r then the impurity is defined by:

$$\text{GI}(j, s) = \frac{|R_l(j, s)|}{n_{tot}}\text{Imp}(R_l(j, s)) + \frac{|R_r(j, s)|}{n_{tot}}\text{Imp}(R_r(j, s))$$

Assuming that the split respects the overall distribution $\forall k, p_{mk} = \frac{|C_k|}{N}$, then all regions are identically distributed and have Gini impurity:

$$\text{GI}(R) = \sum_{k=1}^{K} \frac{|C_k|}{N}(1 - \frac{|C_k|}{N}) = 1 - \sum_{k=1}^{K}(\frac{|C_k|}{N})^2$$

For a K-way split $\text{GI}(j, s) = \sum_{l=1}^{K} \frac{1}{K}\text{GI}(R) = \text{GI}(R)$ and in particular if $|C_k| = \frac{N}{K}$ then $\text{GI} = 1 - \frac{1}{K}$. If the split is perfect then all regions have a proportion of 1 of one class and of 0 of the other, and hence have Gini impurity $\text{GI}(R) = 0$.

3.6 When to stop growing a tree?

If a tree is too large it might overfit while a small tree might underfit. Therefore we have to establish a strategy in order to decide when to stop growing a tree. There are two main strategies:

- we can either grow the tree until a minimum node size (number of training points in the region) is reached
- or we can prune the tree using cost-complexity pruning

Cost-complexity pruning

In a similar way to what we did with regularisation we define the cost complexity with the following formula which includes a complexity penalty :

$$C_\alpha(Tree) = \sum_{m=1}^{\#Tree} N_m Q_m(Tree) + \alpha(\#Tree)$$

where:

- #Tree is the number of regions in our pruned tree
- N_m is the number of training instances in R_m
- Q_m is the error on R_m
- α is a coefficient which defines the balance between model complexity and goodness of fit

The goal is to find a subtree which will minimise the cost-complexity. The more leaf nodes that the tree contains the higher complexity of the tree because we have more flexibility in partitioning the space into smaller pieces, and therefore more possibilities for fitting the training data. There is also the issue of how much importance to put on the size of the tree. The complexity parameter α adjusts that.

4 Advantages and drawbacks of trees

Advantages	Drawbacks
Easy to explain Mirror human-decision making Displayed graphically and easily interpreted Can easily handle quatitative variables Naturally handle multiclass problems	Bad predictive accuracy in general

In order to improve the predictive accuracy we will use forests.

5 Forest

Ensemble learning. The idea is that aggregating many weak learners can substantially increase their performance, because trees have weak predictive power in general. This method is called ensemble learning and uses the principle of wisdom of crowds, that by combining multiple individual classifiers, we can average out their uncorrelated errors. This is what we can observe with the example below, by combining many 'staircase' boundaries, we finally obtain a diagonal separation.

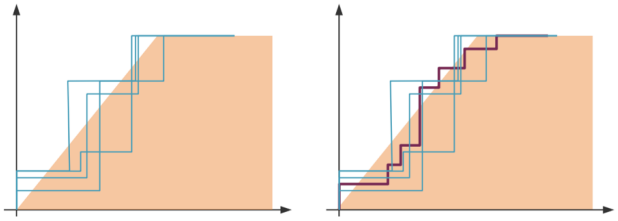

Figure 4: Principle of ensemble learning

5.1 Building ensembles

To build ensembles, we have to:

- **subsample the training data** : by either bagging the data, picking multiple times the same point, or boosting which is resampling based on performance.
- Use **different features** (with multiple input representations, we will also learn how to select features in the chapter 11).
- Use different **parameters** of the learning algorithm.

When we combine weak learners :

- if it is a non-trainable combination: it should be done by **voting** (classification) or **averaging** (regressing).
- if it is a trainable combination: we use **weighted averaging** based on performance on a validation set or **meta-learner** with which the outputs of the individual learners are features for another learning algorithm.

5.2 Bagging trees

The principle is to bootstrap the training data (take repeated samples) and build one predictor from each of these samples. The final prediction is the **average** of the samples when considering a **regression** problem and a **majority vote** for a **classification** problem.

5.3 Random forest

The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables: before each split, select q (out of p) input variables at random as candidates for splitting. We typically choose q = $\sqrt{p}$. This method has very good predictive power in practice!

6 Summary

Decision trees are easy to interpret.
Decisions trees elegantly deal with :

- Quantitative variables
- Multiple classes
- Multimodal distributions

Decision trees have limited predictive power, but this can be addressed thanks to ensemble methods:

- Bagging
- Random forests