Machine Learning

Module 3.2 - Models: Classification And Regression Trees

Marc-Olivier Boldi

Master in Management, Business Analytics, HEC UNIL

Spring 2024

Table of Contents

Table of Contents

CART

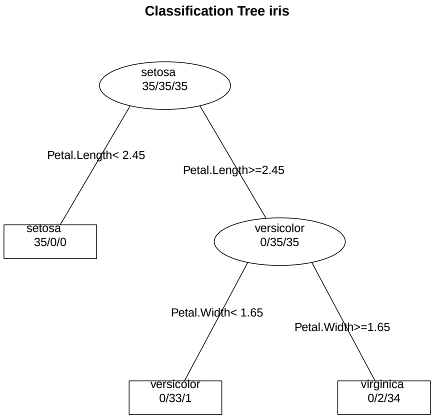
CART stands for **C**lassification **A**nd **R**egression **T**rees. It consists of

- A hierarchical set of binary rules,
- A representation in a shape of a tree.

It applies to both regression and classification tasks.

The prediction formula

Iris data (on 105 data; 35/35/35).



If Petal length < 2.45 then Setosa, else

- if Petal width < 1.65 then Versicolor, else Virginica.

The prediction formula

The set of rules including the order of the features and the thresholds at each node can be embedded into a set of parameters θ.

After the training, the tree contains of final nodes. We use the notation below,

- Applying the rules to features x send the instance (y, x) to the final node that we call $N(x)$.
- In the training set $(y_1, x_1), \ldots, (y_n, x_n)$, the set of instances that were sent to node N is called $I(N)$. E.g., $I(N) = \{1, 3, 4\}$ means that instances $(y_1, x_1), (y_3, x_3), (y_4, x_4)$ were sent to node N.
- The prediction for features x is noted $f(x; \theta)$. That prediction only depends on the node to which x is sent. In other words,

$$N(x) = N(x') \quad \Longrightarrow \quad f(x; \theta) = f(x'; \theta).$$

The prediction formula: regression

For regression, $f(x;\theta)$, the prediction for x, is the mean of the y_i's that were sent to the node of x during the training.

In more fancy terms,

$$f(x;\theta) = \frac{1}{|I(N(x))|} \sum_{i \in I(N(x))} y_i.$$

The prediction formula: classification

For classification,

- The node $N(x)$ has a predicted probability vector: probabilities one each possible class

$$p(x;\theta) = (p_1(x;\theta), \ldots, p_C(x;\theta)),$$

- The prediction $f(x;\theta)$ is the class that has highest probability:

$$f(x;\theta) = \arg \max_{c=1,\ldots,C} p_c(x;\theta).$$

The predicted probabilities $p(x;\theta)$ at node $N(x)$ are the proportions of each class taken on all the instances sent to $N(x)$ during the training.

In more fancy terms, for any class c,

$$p_c(x;\theta) = \frac{1}{|I(N(x))|} \sum_{i \in I(N(x))} 1_{\{y_i = c\}}.$$

The loss function: regression

For the regression case, the most used loss function is the MSE:

$$\bar{\mathcal{L}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\{y_i - f(x_i;\theta)\}^2$$

The loss function: classification

For the classification case, the most used loss is the **entropy**. It is different from the one of logistic regression[1]. In the case of C classes, the entropy is

$$\begin{aligned}\mathcal{L}(y,p) &= -p_1 \log p_1 - p_2 \log p_2 - \cdots - p_C \log p_C \\ &= -\sum_{c=1}^{C} p_C \log p_c.\end{aligned}$$

A the global entropy is thus

$$\bar{\mathcal{L}}(\theta) = -\sum_{i=1}^{n}\sum_{c=1}^{C} p_c(x_i;\theta)\log p_c(x_i;\theta).$$

[1]For logistic regression, it is the cross-entropy.

The loss function: classification

Other possible choices for the loss function are:

The **classification error**:

$$\mathcal{L}(y, p) = 1 - p_y.$$

If $y = c$, then p_c should be large and $1 - p_c$ should be small.

The **Gini index**:

$$\mathcal{L}(y, p) = p_1(1 - p_1) + \cdots + p_C(1 - p_C) = \sum_{c=1}^{C} p_c(1 - p_c).$$

This index is smaller if p has one large component and the other ones small (e.g., $(1, 0, 0, 0)$). It is thus minimum when the prediction is certain. For this, it is similar to the entropy.

The greedy algorithm

Finding the optimal θ cannot be obtained using a Newton-Raphson algorithm because $f(x;\theta)$ is not differentiable.

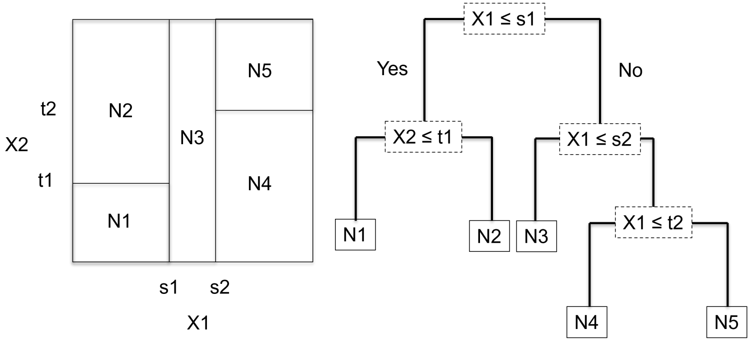
To obtain the optimal $\hat{\theta}$, one should search among all possible splitting combinations. The complexity of such task is enormous and cannot be achieved even with the most powerful imaginable computers.

An approximate solution is obtained by the **greedy algorithm**.

Table of Contents

Splitting the data space

The tree splits the space into rectangles[2]. Below with two features:



Each rectangle is associated to one node and one prediction.

[2]Several trees may be equivalent.

A greedy algorithm

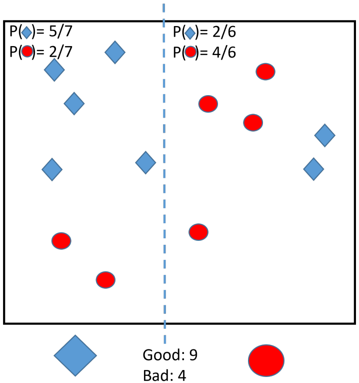
The greedy algorithm grows the tree step by step as follows:

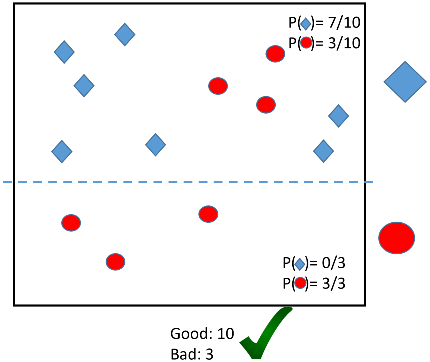
- Start with no partition,
- Along each feature x_j, find the best split of the training set by this feature. This gives p splits (one for each feature).
- Compare the p splits and select the best one: the one diminishing the loss the most.

You obtain two nodes, N_1 and N_2, say. Repeat the above rule on each node until a stopping rule is reached (see later).

A greedy algorithm

Example of classification:



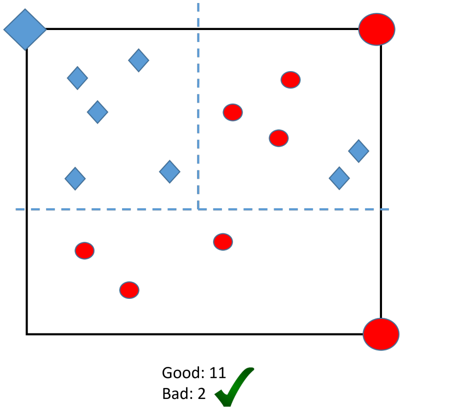


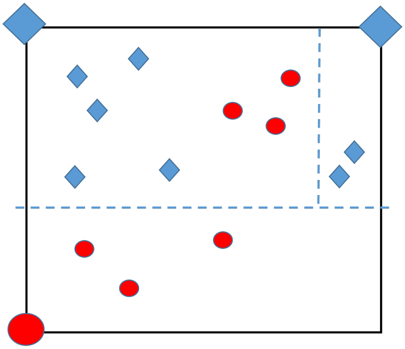
Entropy:
-(5*(5/7)*ln(5/7) + 2*(2/7)*ln(2/7) + 4*(4/6)*ln(4/6) + 2*(2/6)*ln(2/6))=3.73

Entropy:
-(7*(7/10)*ln(7/10) + 3*(3/10)*ln(3/10) + 0*(0/3)*ln(0/3) + 3*(3/3)*ln(3/3))=2.83

Note: by convention $0 \times \ln 0 = 0$.

CART: a greedy algorithm





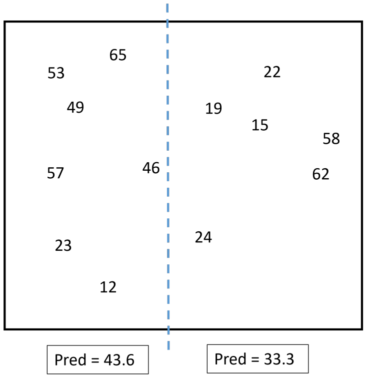
Entropy (upper node):
-(5*(5/5)*ln(5/5) + 0*(0/5)*ln(0/5) + 3*(3/5)*ln(3/5) + 2*(2/5)*ln(2/5))=1.65

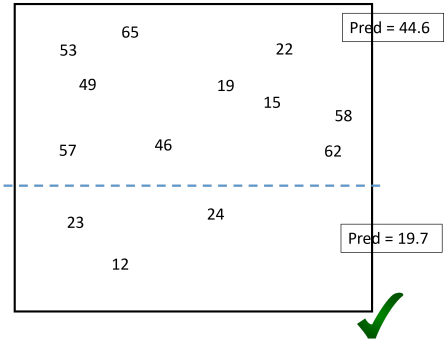
Entropy (upper node):
-(5*(5/8)*ln(5/8) + 3*(3/8)*ln(3/8) + 2*(2/2)*ln(2/2) + 0*(0/0)*ln(0/0))=2.57

etc.

A greedy algorithm

Example of regression:





MSE:
((65-43.6)^2 + … + (12-43.6)^2+(22-33.3)^2+…+(24-33.3)^2)/13 = 336.2

MSE:
((65-44.6)^2 + … + (62-44.6)^2+(23-19.7)^2+…+(24-19.7)^2)/13 = 251.9

Stopping rule

The algorithm could go on splitting the feature space forever. Several stopping rules can be used. For example in `R` with `rpart` (see `?rpart.control`):

- `minsplit`: the minimum number of observations that must exist in a node in order for a split to be attempted (default is 20).
- `cp`: Any split that does not decrease the overall lack of fit by a factor of cp is not attempted (default to 0.1).
- `minbucket`: the minimum number of observations in any terminal node (default to `minsplit/3`).
- etc.

Even with these rules, once the tree is grown, it might be too large. The procedure of cutting nodes is called `pruning` the tree.

Table of Contents

Interpretation

Trees can be interpreted by reading directly on the graph the link between the features and the outcome.

- The most important feature is at the top of the graph. The first split influence the other ones.
- The final nodes should be organized in a logical way to ease the reading. E.g.,
  - (regression) Low predictions to the left, high predictions to the right.
  - (binary) predictions "0" to the left, and "1" to the right.

Interpretation must be done with cautious. The tree structure is unstable: several trees can produce similar predictions. The greedy algorithm may just pick one at random.

Table of Contents

Occam's razor

For linear and logistic regressions, the more variables are in the model, the more complex that model is.

For trees, model complexities can be measured by the length of the tree: a shorter tree is more simple than a longer one.

To apply Occan's razor, one needs to shorten the tree without impeding too much its prediction quality.

To **prune** the tree consists of cutting branches while maintaining enough prediction quality.

1-SE rule

There exist several ways to prune the tree. One of the most used is the **1-SE** rule.

- Build a long tree (using the default stopping rules)
- For all the sub-trees compute the error of the tree
- For the best tree (lowest error) compute the uncertainty on that error measure: the **standard error** SE.
- Any tree whose error is below the lowest error plus one SE is considered equivalent to the best tree.
- Select the shortest tree among those that are equivalent to the best one.

Technical details

- The SE is computed using cross-validation (see later).
- The 1-SE rule can be replaced by other rules (like cutting at a given CP, etc.).

In practice,

- the pruning is not a guarantee to obtain a good model.
- It should be used to simplify the model and avoid overfitting.
- (empirical) it works well to build a very large tree then to prune it.