Classification and Regression Trees (CART)

Anca Ralescu

CART Training Algorithm (growing trees)

Simple idea:

1. Split the training set in two subsets using a single feature k and a threshold t_k (e.g., petal length $\leqslant$ 2.45 cm).
   - To choose k and t_k: search for the pair (k, t_k) that produces the purest subsets (weighted by their size).
2. Split subsets by the same logic.
3. Recursion stops once the tree reached a maximum depth or if no split which reduces impurity can be found.

CART Cost Function for classification

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right},$$

where $\begin{cases} G_{left/right} \text{measures the impurity of the } left/right \text{ subset} \\ m_{left/right} \text{ is the number of instances in the left/right subset.} \end{cases}$

Gini impurity index

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2, \text{where}$$

- $p_{i,k}$ is the ratio of class k instances among the training instances in the ith node.

CART is a greedy algorithm:

- it greedily searches for an optimum split at the top level, then repeats the process at each level.
- It does not check whether or not the split will lead to the lowest possible impurity several levels down.

- it often produces a reasonably good solution, but it is not guaranteed to be the optimal solution.
- the time complexity of the optimal solution is $O(e^m)$: $NP-Complete$ problem.

Complexity of prediction

Overall prediction complexity is $O(\log_2(m))$

Training complexity is $O(n \times m \log(m))$, where n is the number of features or the $maximum$ number of features used.

Figure 1 illustrates a a classification tree built using the Gini impurity:

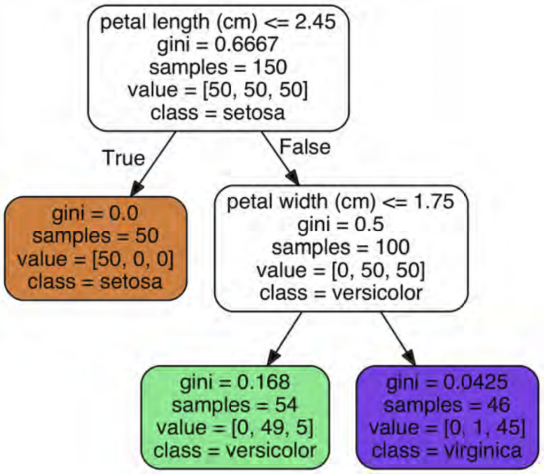

Figure 1: Decision/Classification tree for the Iris dataset based on the Gini impurity

The decision boundaries corresponding to the tree from Figure 1 are shown in Figure 2.

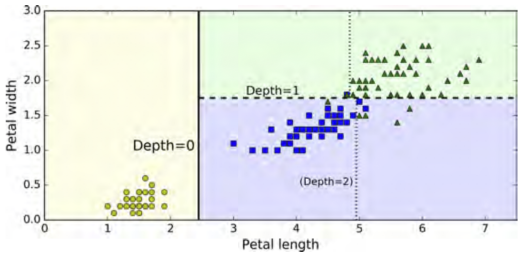

Figure 2: Decision Tree decision boundaries

Gini impurity index versus Entropy (white box model)

- The choice makes no big difference with respect to the final result: the trees obtained are quite similar.
- Gini impurity is slightly faster to compute: $\Longrightarrow$ it is a good default.
- When the trees differ, Gini impurity tends to isolate the most frequent class in its own branch of the tree, while entropy tends to produce slightly more balanced trees.

Estimating class probabilities

A Decision Tree can estimate the probability that an instance x, belongs to a particular class k:

1. Traverse the tree to find the leaf node for instance x
2. Return the ratio of training instances of class k in this node.
    - Example: flower with petals 5 cm long and 1.5 cm wide corresponds to the leaf node, is the depth-2 left node. So, DT should output the probabilities:
        (a) P(Setosa) = 0/54 = 0
        (b) P(Versicolor) = 49/54 = 0.907
        (c) P(Virginica) = 5/54 = 0.093

Very Important Note 1 *The estimated probabilities would be identical anywhere else in the bottom-right rectangle of Figure 2, if the petals were 6 cm long and 1.5 cm wide (even though it seems obvious that it would most likely be an Iris- Virginica in this case).*

Regularization

Decision trees tend to adapt to the data, and hence they tend to overfit. To prevent this regularization is used. The following conditions may be imposed. The quantities involved in these conditions are called *hyperparameters*. The regularization restricts the values of the following *hyperparameters*:

- max_depth
- $min\_samples\_split$: this is the minimum number of examples a node must have before it can be split;
- $min\_samples\_leaf$: the minimum number of samples a leaf node must have;
- $min\_weight\_fraction\_leaf$: this is the same as $min\_samples\_leaf$, but expressed as a fraction of the total number of weighted instances;
- max_leaf_nodes: maximum number of leaf nodes;

- $max\_features$: maximum number of features that are evaluated for splitting at each node;

Increasing min_* hyperparameters or reducing max_* hyperparameters regularizes the model.

The Moon Dataset

Figure 3 shows a data set with two classes which are not linearly separable. It is possible to map this data set into a space of higher much higher) dimension where the two classes become linearly separable.

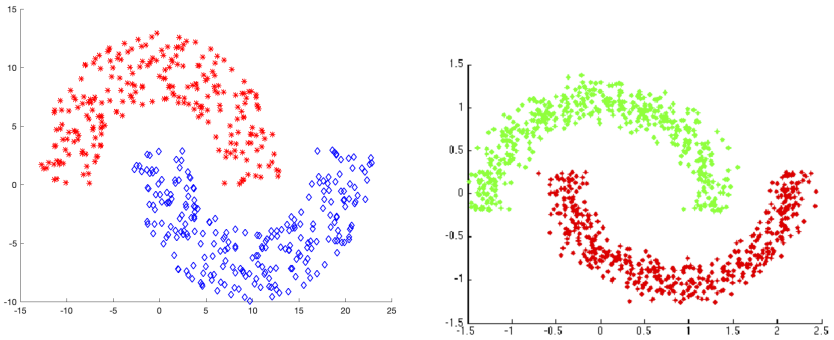

Figure 3: Moons Dataset

Figure 4 shows two DTs for the moons dataset: the left is trained with the default hyperparameters (i.e., no restrictions); the right DT is trained with $min\_samples\_leaf = 4$. The model on the left is overfitting; the model on the right will generalize better.

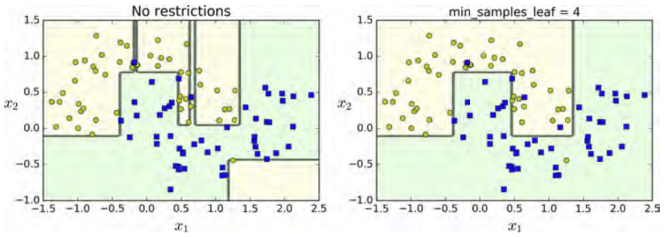

Figure 4: Regularization using *min_samples_leaf*

Regression Trees

DTs can perform regression tasks as well. In this case the 'label' y has a continuous value, i.e., $\in \mathbb{R}$.

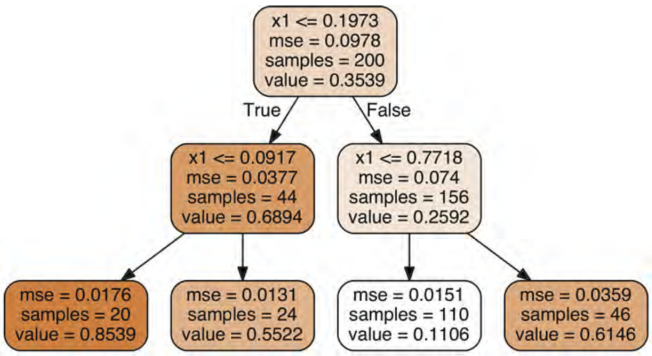

Figure 5: Regression tree.

Figure 5 shows a regression tree.

The tree looks similar to a classification tree, except that at each node it computes the *node value*, and the MSE. These are obtained by averaging the values of the nodes in that node. The predictions made by this tree are shown in Figure 6.

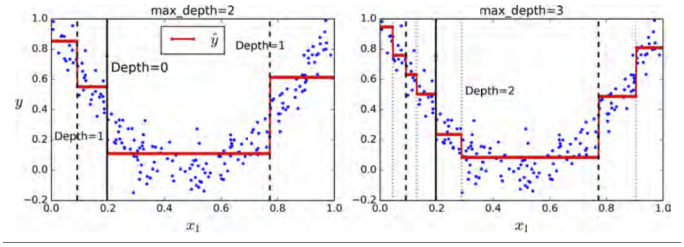

Figure 6: Regression tree predictions. Left *max_depth* is set to 2; Right *max_depth* is set to 3.

CART cost function for regression

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right},$$

where $\begin{cases} MSE_{node} = \sum_{example \in node} \left(\hat{y} - y^{(example)}\right)^2 \\ \hat{y} = \sum_{example \in node} y^{(example)} \end{cases}$

Regularization of Regression Trees

Use restricted hyperparameters in order to avoid overfitting (Figure 7)

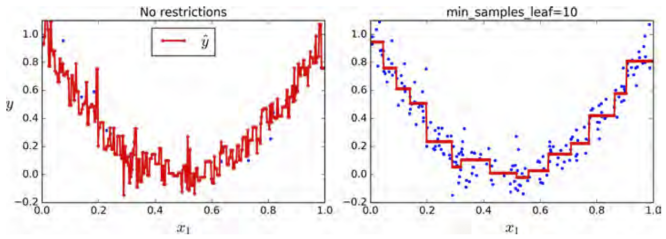

Figure 7: Regression tree regularization: Left, no regularization; Right, *min_samples_leaf* = 10

Limitations of DT

Instability to rotation

Because DT have orthogonal decision boundaries (all splits are perpendicular to an axis), which makes them sensitive to training set rotation.

Figure 8 shows a simple linearly separable dataset:

- Left Decision Tree can split it easily;
- After the dataset is rotated by $45°$, the decision trees of the right produces unnecessarily convoluted decision surfaces.
- Although both DTs fit the training set perfectly, it is very likely that the model on the right will not generalize well.

One way to limit this problem is to use Principal Component Analysis, which will lead better results.

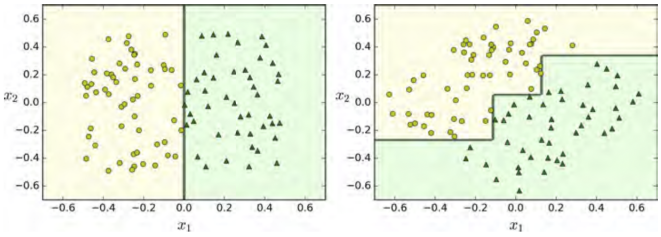

Figure 8: Sensitivity to training set rotation

Instability to details in the data

The above suggests that the main issue with DTs is their extreme sensitivity to small variations

Example 1 *Removing the widest Versicolor from the iris training set (the one with petals 4.8 cm long and 1.8 cm wide) and re-training a DT, may result in the decision surface shown in Figure 9, right. This looks different from the previous one shown in Figure 2, also shown in Figure 9, left.*

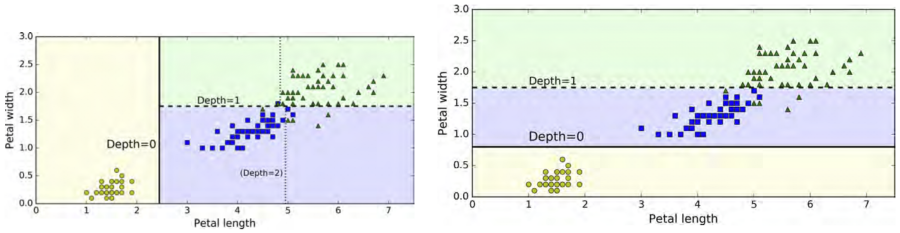

Figure 9: Sensitivity to data details: The original DT for iris dataset (left); Decision Surface for the data set where the widest instance of the Versicolor was removed (right)