Decision tree

Machine Learning II

December 23, 2020

1 Description

Decision trees are the initial and most basic algorithm in the CART family (Classification and Regression Trees).

These algorithms are nothing but a set of if-then-else rules, very easy to understand and to implement. Trees can discover hidden patterns in the data and given this patterns, predict categorical or continuous variables. They can be expressed in terms of feature relationship so they're very interpretable.

2 Key concepts

- Recursive partitioning: iterative way of splitting the data so that our final splits (leaves) are as homogeneous as possible
- Split value: certain value in a feature with according to which we split the data into two pieces. Each split produces two branches.
- Node: graphical representation of a split value
- Leaf: the end of each if-else rule. Leafs can contain one or more observations. The less observations in each leaf, the more overfitting.
- Loss: measure of how impure or heterogeneous groups are after a split
- Impurity: measure of how mixed the data is within a partition Pruning: manually reducing the number of branches to reduce overfitting.

3 How it works?

Let's introduce first how to measure impurity, then how the partitioning process works, and then we'll do it recursively.

For measuring impurity first we need to understand when is a record misclassified when dealing with a binary variable. If p is the proportion of misclassified records in a partition A, then $(1-p)$ is the proportion of correctly classified records. The value of p ranges from 0 (perfect) to 0.5 (random guessing).

The two most common measures of impurity are the *Gini impurity* and *Entropy of information.*

- Gini impurity

$$I(A) = p \cdot (1-p) \tag{1}$$

- Entropy of information

$$I(A) = -\sum_{j=1}^{n} P(X_j) \cdot log_2 P(X_j) = -p \cdot \log_2 p - (1-p) \cdot \log_2 (1-p) \tag{2}$$

Now that we know how to measure impurity, let's learn about partitioning.

```
# partitioning
for each feature in features:
    for split_value in feature:
        1. create two partitions according to split_value
        2. measure impurity in each partition
select feature and split_value that produces minimum impurity
```

If we do the partitioning iteratively we'll be able to classify the whole dataset

```
# recursive partitioning
1. A = entire dataset
2. apply partitioning to A, resulting in A1 and A2
3. repeat 2 in A1 and A2
4. stop when no further partition can be made that improves the impurity
```

4 What could go wrong?

Now that we know how to grow a tree, we can find situations in which the tree keeps growing and in every split the rules are more detailed and specific creating terminal partitions that are very small and pure. The more splits and partitions, the less bias and more variance we achieve in our estimator. If we don't balance the growth we will fall into overfitting very easily -our estimator won't be able to generalize enough and will perform poorly on the test dataset.

To control the growth of our tree we can use *pruning*. This tool allows us to decide how many branches, and the size of the intermediate (branches) and terminal (leaves) partitions. In Python's scikit-learn, we can control this by tuning the following hyperparameters:

- `max_depth`: maximum depth of the tree. It can be visualized as the maximum length between the root node to a leaf.
- `min_sample_split`: minimum number of observations that must be present in a partition to be split in a node.
- `min_samples_leaf`: minimum number of observations that must be contained in a leaf (terminal partition).

5 When to use it?

- When not only looking for predictions but also explainability of a system
- Little data preproccessing needed. Trees can deal with almost everything.
- Good to use when the classes are balanced, if that's not the case, better use first some balancing methods before using trees.

6 Summary

Type	Regression, Classification
Tolerates many features?	Strong
Parametrization	Simple, intuitive
Memory needed	Large
Data requirements	Small
Overfitting	Very high
Difficulty	Weak
Training time	Weak
Prediction time	Weak
Interpretability	Very good