Jacopo Dapueto s4345255 A.A. 2019/2020

Decision Tree for classification

Introduction

A Decision Tree e is a flowchart-like tree structure that allows us to split the dataset into partitions and it's made up of:

- **Internal nodes** which denote a test on an attribute;
- **Branches** representing the outcomes of the tests;
- **Leaves nodes** holding the classes labels (or classes distributions);
- The topmost node in a tree is the **root node**.

Decision Tree induction is a supervised learning algorithm. It has a top-down approach, it starts with the training set and recursively it is partitioned into smaller subsets as the tree is being built.

A basic decision tree algorithm for classification problems is the following:

```
Generate_DecisionTree(D[1]):
(1) create a node $N$;
(2) if tuples in $D$ are all of the same class C, then
(3)     return $N$ as a leaf node labeled with the class C;
(4) apply bestAttribute(); // to find the best splitting criterion (attribute and split-point);
(5) label node $N$ with splitting criterion;
(6) for each outcome j of splitting criterion:
        // partition the tuples and grow subtrees for each partition
(7)     let $D_j$ be the set of data tuples in $D$ satisfying outcome j; // a partition
(8)     if $D_j$ is empty then:
(9)             return $N$ as a leaf node labeled with the majority class in $D$ to node $N$;
(10)    else:
                attach the node returned by Generate_DecisionTree($D_j$) as child of the node $N$;
    endfor;
(11) return $N$;
```

The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as "pure" as possible. A partition is pure if all the tuples in it belong to the same class.
The algorithm uses the same process recursively to form a decision tree at each resulting partition D_j of D.

The recursive algorithm stops only when any one of the following terminating conditions is true:

1. All the tuples in partition D at the node N belong to the same class (steps 2 and 3).
2. There are no tuples for a partition D_j is empty (step 8). In this case, a leaf is created with the majority class in D (step 9).

Attribute Selection Measures

With an attribute selection measure we are trying to select the splitting criterion that "best" separates a given data partition D into individual classes. Conceptually, the "best" splitting criterion is the one that most closely results in such a scenario. Attribute selection measures are also known

[1] With D I denote a partition as $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ contained in a node.

as splitting rules because they determine how the tuples at a given node are to be split. The attribute selection measure provides a ranking for each attribute describing the training set and the one having the best score for the measure is chosen as the splitting attribute for the given tuples.

Two kinds of attribute selection measure are the **Information Gain** and **Gini Index**.

Information Gain

Let node *N* hold the tuples of partition D. The attribute with the highest **information gain** is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the tuples in the resulting partitions, which is measured as:

$$entropy(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \qquad where\ p_i = \frac{|C_{i,D}{}^2|}{|D|}$$

The p_i is the nonzero probability that an arbitrary tuple in D belongs to class C_i[3] and is estimated with $\frac{|C_{i,D}|}{|D|}$: if $p_i = 0$ then $p_i \log_2(p_i) = 0$.

$entropy(D)$ is the average amount of information needed to identify the class label of a tuple in D. Now we need to estimate how much information is needed, after partitioning, to have an exact classification. Assuming we want to partition D using the attribute A and the test on it has v different outcomes, the measurement is given by

$$entropy_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} entropy(D_j)$$

The smaller the $entropy_A(D)$, the greater the purity of the partitions.

Suppose we have an attribute *A* that is continuous-valued, we must determine the best **split-point** for *A*: first of all we sort the values of *A* in increasing order, the midpoint between each pair of adjacent values is considered as a split-point. Having d values for the attribute *A*, we consider $d-1$ possible split-points.

For each possible split-point for *A*, we evaluate $entropy_A(D)$, where $v = 2$. The split-point with the **minimum** $entropy_A(D)$ is selected as the split-point for *A*: D_1 is the set of tuples in D satisfying *A ≤ split-point*, and D_2 is the set of tuples satisfying *A > split-point.*

The Information gain is defined as the difference between the original information requirement and the new requirement obtained after partitioning on *A*.
That's $Gain(A) = entropy(D) - entropy_A(D)$ and it measures the expected reduction in the information requirement caused by knowing the value of *A*.

Gini index

the Gini index measures the impurity of D:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 = \sum_{i=1}^{m} p_i(1-p_i)$$

[2] $C_{i,D}$ be the set of tuples of class C_i in D.

[3] The class label attribute has m distinct values defining m distinct classes, C_i (for $i = 1,\dots,m$).

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $\frac{|C_{i,D}|}{|D|}$.

The Gini index considers only binary split for each attribute: if a partition D is splitted into D_1 e D_2 using the attribute A, we have

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

For continuous-valued attributes the procedure is the same as we saw for the Information gain.

The reduction in impurity if we partition the data using the attribute A is measured with

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

The attribute that maximizes $\Delta Gini(A)$ is chosen as the splitting attribute.

Tree Pruning

With Decision Tree Induction the model can easily overfit the training set and it is sensitive to noisy data. There are many algorithm with the aim of reducing overfitting: almost all “prune” the decision tree in order to get a smaller and less complex tree than an unpruned one. **Pruned** tree are usually faster and better at classifying the **test** set. The two main approaches are *pre-pruning* and *post-pruning*.

Pre-pruning

This approach limit the growth of the tree during the training, it prevents the generation of non-significant branches. It’s done using *terminal conditions* in order to terminate the algorithm.

While constructing a tree the Information gain and the Gini index can be used to halting the algorithm: at the node N if the Information gain or the Gini index are less than a threshold then the node becomes a leaf, otherwise the partitioning can keep going on. If the threshold is high the tree could be oversimplified, otherwise if the threshold is very low we could get a tree with very little simplifications.

The maximum depth of the tree can be used as a parameter to prevent the creation of new nodes: depending on the value the algorithm could result on a oversimplified or not simplified tree.

Both the “best” threshold and the “best” maximum depth can be estimated with cross validation.

Post-pruning

This is the most common approach and it removes subtrees from a fully grown tree: a subtree at a given **internal node** is replaced with a leaf having as a label the most common class in the subtree.

There is a wide range of postpruning algorithm, I implemented the *Reduce Error Pruning* which is the simplest and most understandable method in decision tree pruning. The available data is divided into 3 sets: training set, pruning set and test set. So the tree is trained using the training set and pruned with the pruning set in the following way: starting from the bottom, for each internal node the error (over the pruning set) is calculated as the subtree was pruned and if it smaller or equal to the error as the subtree wasn’t pruned, then the subtree is pruned. Otherwise the subtree is not pruned.
The algorithm has a linear computational complexity and the performance in terms of accuracy are similar to the most of other pruning methods but it’s better of most in terms of tree size.