Lecture 9: Tree-Based Methods

ENEE 691 – Machine Learning and Photonics @ UMBC

Ergun Simsek, Ph.D. and Masoud Soroush, Ph.D.

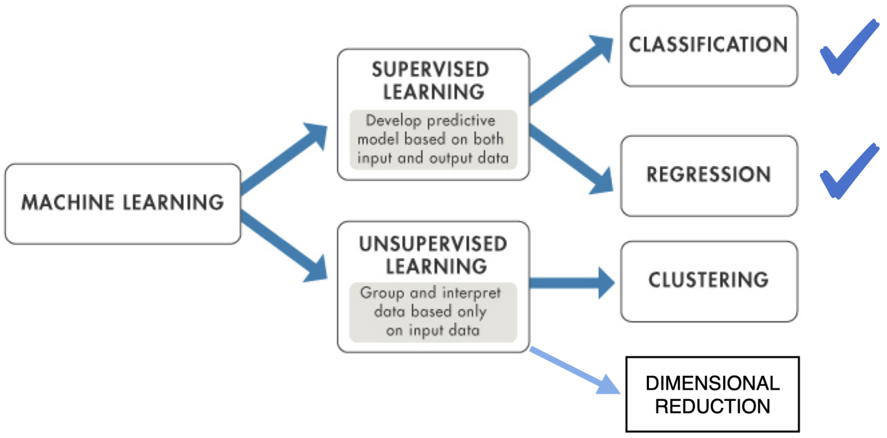
March 23, 2023

1 Tree-Based Methods

In this chapter, we introduce the tree-based methods that can work with both categorical and continuous target variables, and hence, they can be applied as both classification and regression algorithms in the realm of supervised learning.

1.1 Background

The algorithms we have introduced so far are either applied for regression analyses or for the classification in supervised learning. Today, we will introduce a simple and useful approach that can be used for both regression and classification problems.



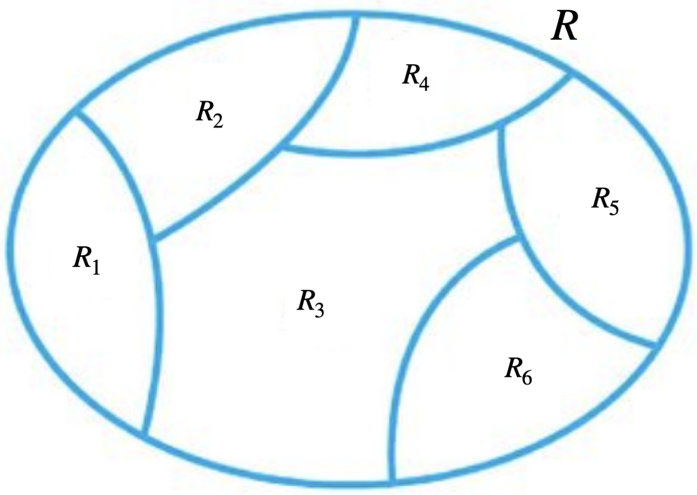
Tree-based methods involve segmenting the feature space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode response value for the training observations *in the region to which it belongs*. Since the set of splitting rules used to segment the feature space can be summarized in a tree, these approaches are known as decision tree methods.

1.2 Regression Trees

To set the stage, assume that we have d continuous features, represented by $\vec{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$. The target variable in this case is a continuous variable represented by $y \in \mathbb{R}$. The basic idea is to stratify the feature space $R \subset \mathbb{R}^d$ into a finite number of simple non-overlapping regions (*i.e.* we consider a finite *partition* of the feature space). Then to make predications for a given observation $\vec{x}^*$, we use the mean of the target variable y of the *region $\vec{x}^*$ belongs to*. In summary, the regression tree involves the two following steps:

- We divide the feature space — that is, the set of possible values for $x_1, x_2, \cdots, x_d$ — into J distinct and non-overlapping regions, R_1, R_2, ..., R_J. Note that the regions R_j must satisfy the following properties:

$$\begin{cases} R_j \neq \emptyset, \ \forall j \\ R_j \cap R_k = \emptyset, \ \text{for } j \neq k \\ R_1 \cup R_2 \cup \cdots \cup R_J = R \end{cases} \tag{1}$$



- To make predication for a given observation that belongs to region R_j, we use the mean of the target values for the *training observations* in the region R_j. Note that the predictions for all observations that belong to the same region R_j is the same.

As an example, assume that we have only two regions R_1 and R_2, and the mean of y for the training set in region R_1 is 5 and in region R_2 is 8. Then the prediction for the target y for a given observation $\vec{x}^*$ is $\hat{y} = 5$ if $\vec{x}^* \in R_1$, and is $\hat{y} = 8$ if $\vec{x}^* \in R_2$.

Now the important question is how we construct the regions R_j's. In principle, the regions R_j can have any shape, but in the tree-based approach, these regions are constructed in a particularly simple manner. The feature space is divided into **higher dimensional rectangles** (or boxes). We should aim to find the rectangles (boxes) in a way that the Residual Square Sum (RSS) function

$$RSS = \sum_{j=1}^{J} \sum_{k \in R_j} (y_k - \bar{y}_{R_j})^2 \tag{2}$$

is minimized. In (2), $\bar{y}_{R_j}$ denotes the mean of the target variable y for the training dataset in the region R_j. Although our goal (*i.e.* minimizing RSS in (2)) is very explicit, it is not computationally viable to consider all possible partitions of the feature space[1], and that is where the tree partitioning enters! To cure this problem, we form the rectangles (boxes) through a **recursive binary splitting** approach. This is a *top-down* approach where one initially sits at the top of the tree where all observations belong to the same region (*i.e.* there is only one region). Then at each successive step, one splits the feature space and adds two more regions (*i.e.* one adds two new branches to the tree). This is a greedy approach, as at each step of constructing the tree, the best split is made *at that particular step* (You do not look ahead for lower branches).

Now, let us formalize the above procedure. At the top of the tree, we select a feature, say x_j, and a splitting point, say $x_j = t$. Then, we split the feature space into two regions $R_1(j,t)$ and $R_2(j,t)$:

$$R_1(j,t) = \{\vec{x} \in R \,|\, x_j < t\}, \qquad R_2(j,t) = \{\vec{x} \in R \,|\, x_j \geq t\}\,. \tag{3}$$

We then form the RSS, and minimize it in order to find the best choice for the splitting feature j and the splitting point t:

$$RSS = \sum_{\{i \,|\, \vec{x}^{(i)} \in R_1(j,t)\}} (y_i - \bar{y}_{R_1(j,t)})^2 + \sum_{\{i \,|\, \vec{x}^{(i)} \in R_2(j,t)\}} (y_i - \bar{y}_{R_2(j,t)})^2 \tag{4}$$

In (4), $\bar{y}_{R_1(j,t)}$ and $\bar{y}_{R_2(j,t)}$ are the means of the target variable y for the training observations in regions $R_1(j,t)$ and $R_2(j,t)$, respectively.

Remark: Note that minimizing the RSS in (4) tells us what feature x_j and what splitting point $x_j = t$ should be taken.

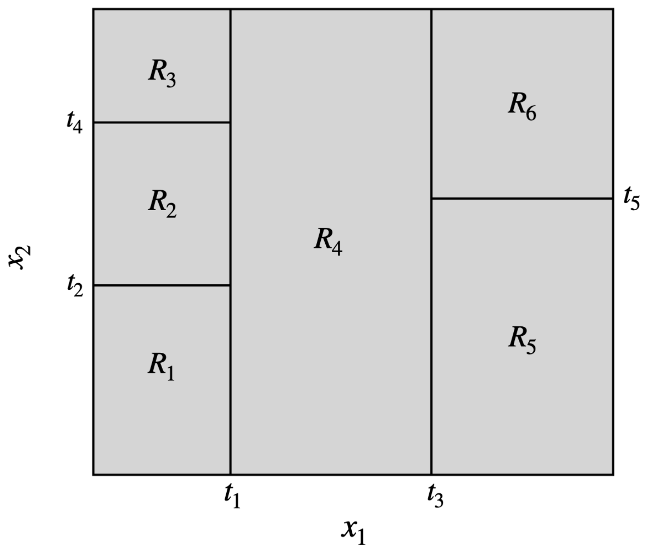
Remark: If the number of features d is not too large, the above minimization procedure can be performed pretty quickly!

Next, we repeat the splitting process. We look for the best feature and best splitting point in order to minimize the RSS within each of the resulting regions. Note that, this time, instead of splitting the entire feature space, we split one of the two previously identified regions (*i.e.* either R_1 or R_2). We now have three regions. Again, we continue to split one of these three regions further, so as to minimize the RSS. This process continues until a stopping criterion is reached. For instance, we may continue until no region contains more than five observations.

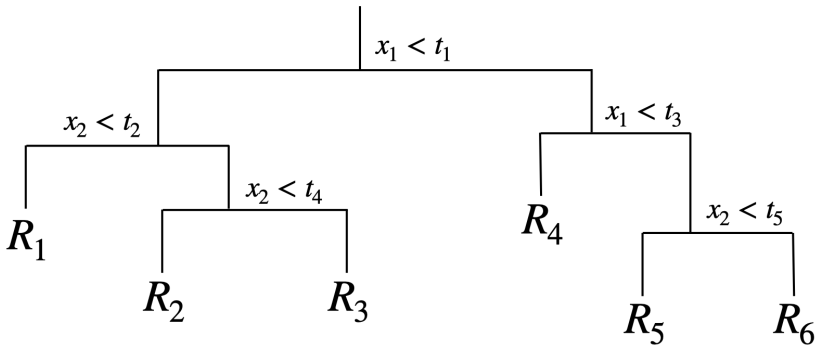
At the end of the process, the higher-dimensional rectangles R_j's are formed, and in order to make a prediction for a given test observation, we offer the mean value of y for the training observations in the region the test observation belongs.

The following figure represents a two-dimensional feature space that has been split into 6 rectangles (boxes).

[1] For a dataset with only $n = 30$ samples, the number of all partitions that can be made by the samples is 846749014511809332450147 which is immensely big!



The following tree represents output of the binary recursive splitting corresponding to the above two-dimensional example.



Remark: In the literature, the final regions (boxes) that are formed at the end of the binary recursive splitting (*e.g.* R_1 through R_6 in the above tree diagram) are referred to as the **nodes** or **leaves** of the tree.

1.3 Classification Trees

Classification trees are used to predict a categorical target variable y. The process of growing a classification tree is very similar to that of a regression tree. However, there are two differences:

- For predicting the class of a given observation, we assign the **most commonly occurring class** of training observations of the region the test observation belongs.

- Instead of RSS (that was used in regression trees), we should use new metrics the *Gini index* or the *cross entropy*.

Now, we need to explain how the Gini index and the cross entropy are defined. Suppose the categorical target variable y can take K distinct class $k = 1, 2, \cdots, K$. Let $\hat{p}_{j,k}$ denote the proportion of training observations in region R_j that belong to class k:

$$\hat{p}_{j,k} = \frac{\text{Number of training observations in region } R_j \text{ which are in class } k}{\text{Total number of training observations in region } R_j} \, . \tag{5}$$

The Gini index, G_j, for region R_j is then defined by

$$G_j = \sum_{k=1}^{K} \hat{p}_{j,k}(1 - \hat{p}_{j,k}) \, . \tag{6}$$

Remark: Note that the Gini index G_j measures impurity at the j-th node (*i.e.* in region R_j). This means that a *small value of G_j is an indicative of the fact that the j-th node has been predominantly occupied by observations coming from one single class*.

Remark: Gini index is a measure of impurity. In constructing the classification trees, we should *minimize the Gini index at each binary splitting point* (rather than the RSS function):

$$\frac{N_{left}}{N} G_{left} + \frac{N_{right}}{N} G_{right} \, , \tag{7}$$

where N_{left}, and N_{right} are the numbers of training observations in the left and right branches, respectively. This is the anolog of equation (4) for classification trees.

An alternative metric to the Gini index is notion of cross entropy. The cross entropy, CE_j, for region R_j is defined by

$$CE_j = -\sum_{k=1}^{K} \hat{p}_{j,k} \log(\hat{p}_{j,k}) \, . \tag{8}$$

Note that since $0 \leq \hat{p}_{j,k} \leq 1$, cross entropy CE_j is a positive semi-definite quantity (*i.e.* it is greater or equal to zero). Similar to the Gini index, cross entropy CE_j takes a small value if the j-th node is pure. Instead of minimizing the Gini index at each binary splitting point, one can alternatively minimize the cross entropy.

1.4 Feature Importance

In the regression analysis lecture, we discussed how the notion of correlation (Pearson, Spearman, and Kendall) could be used to identify the most relevant features for a given continuous target variable. Decision trees offer a novel and different approach to identify the most relevant features. This approach works for both continuous as well as categorical target variables simply because decision trees work for both regression and classification tasks. A notion of *feature importance* can be defined in the decision tree algorithm based on the amount of RSS (for a continuous target) or Gini impurity index (for a categorical target) reduction determined by every single feature.

Remark: This feature (*i.e.* calculating the feature importance) has been built in the scikit-learn library, and we can readily use it whenever a regression or a classification tree is constructed.

1.5 Pros and Cons of Trees

The tree-based approach has both advantages and disadvantages. Among the advantages of tree-based methods, we can mention the following:

- **Simplicity:** The logic of the tree-based methods is particularly simple. In other words, trees are very easy to explain!
- **Graphical Presentation:** Trees can be represented graphically through tree graphs, and are easily interpreted!
- **Categorical Features:** Trees can easily handle categorical features without the need to create dummy variables or any intermediate steps.

Some of the disadvantages of tree-based methods are:

- **Limited Accuracy:** Decision trees do not generally offer the same level of accuracy as other regression and classification methods.
- **Lack of Robustness:** In general, decision trees are not very robust against noise. This implies that a small change in the data may lead to a large change in the final estimated results!

Question: As mentioned above, one of the disadvantages of trees is their limited accuracy. Based on what you learned today about trees, why do you think this is the case?