Graph Theory Project

December 2023

1. Outline of the project

The goal of this project is to use the tree graph as an inference (prediction) tool. You are provided with a dataset of observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$, $n = 100$. Each observation $\mathbf{x}_i$ is a 2-vector containing a value belonging to $\mathbb{R}$ and a label belonging to $\{0, 1\}$. We assume here that the two labels represent two different groups of observations and the final goal is to build a decision tree that will assign a label (i.e. a group) to a new observation $\tilde{\mathbf{x}}$ with unknown label. You will use the CART (Classification and Regression Tree) Algorithm to solve this problem. This algorithm aims to build a binary decision tree that assign a label to $\tilde{\mathbf{x}}$. Every vertex of a binary decision tree is such that it has exactly 0 or 2 children and every vertex contains a threshold value. To classify a new observation $\tilde{\mathbf{x}}$, we give $\tilde{\mathbf{x}}$ as an input to the classification tree and check if said observation is smaller than the threshold value contained in the root node. If it is the case, we travel to the first child node, else to the second one. We repeat this process until we get to a node (let say N_t) with no children (called a leaf). Then, we predict the label of $\tilde{\mathbf{x}}$ by taking all the datapoints in $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ such that they belong to the node N_t and use as an estimator of the label of $\tilde{\mathbf{x}}$ the most common label value for all the datapoints in $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ such that they belong to such node. The goal is, given $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to find a set of threshold values that will give some predictive power to the binary decision tree.

You can use Python, R or C++ to complete the project and should submit through moodle only a **documented** source code file.

2. Step by step

1. You are provided a .csv file, open it with Python/R/C++. The first column (V1) represents the value of each datapoint while the second column (V2) represents its label.

2. You will build the decision tree T now. You will use the so-called Gini impurity to assess the quality of a split of the dataset, split the data in two subsets then find another good split for each one of these subsets and so on. We denote the label of $\mathbf{x}_i$ as $l(\mathbf{x}_i)$ and the value of $\mathbf{x}_i$ as $v(\mathbf{x}_i)$. At the first step, you need to determine the value for the first split $c_1 \in \mathbb{R}$. First check if the number of observations you have in

the first node is larger that some threshold α with $\alpha = 8$ to avoid overfitting. α acts here as a parameter imposing that the number of leaves won't be too large and its value (8) is given arbitrarily. Let define
$P_1 = |\{\mathbf{x}_i \mid v(\mathbf{x}_i) < c_1\}|$ and $Q_1 = |\{\mathbf{x}_i \mid v(\mathbf{x}_i) > c_1\}|$ and $p_1 = \frac{\sum_{\{\mathbf{x}_i | v(\mathbf{x}_i) < c_1\}} l(\mathbf{x}_i)}{P_1}$ and $q_1 = \frac{\sum_{\{\mathbf{x}_i | v(\mathbf{x}_i) > c_1\}} l(\mathbf{x}_i)}{Q_1}$. Here p_1 and q_1 represent the proposition of label 1 observations in the subsets made of the observations respectively smaller and larger than c_1. You will chose c_1 such that it minimises

$$P_1 p_1(1-p_1) + Q_1 q_1(1-q_1),$$

the so-called Gini impurity. Then, iterate : check if $|\{\mathbf{x}_i \mid v(\mathbf{x}_i) < c_1\}| > \alpha$ and determine in the same way as before c_2 for the subset of observations $\{\mathbf{x}_i \mid v(\mathbf{x}_i) < c_1\}$, check if $|\{\mathbf{x}_i \mid v(\mathbf{x}_i) > c_1\}| > \alpha$, then determine c_3 for for the subset $\{\mathbf{x}_i \mid v(\mathbf{x}_i) > c_1\}$ and so on up until every leaf (node without children) is such that it contains at maximum α observations. You need to store all the threshold values $c_1, \ldots, c_k$ in a tree data structure that should allow you, given a node N_k to get both the parents and children nodes of N_k. You can implement the tree data structure how you see fit. If you want to do it properly, the best way is to use object oriented programming but, since the problem at hand here is very simple it is also acceptable to store the tree in a matrix $\mathbf{M}$ of size $k \times 4$. With such a structure, nodes are identified with integers $i \in \{1, \ldots, k\}$ and the node i, $1 \leq i \leq k$ is such that the i-th line of $\mathbf{M}$, denoted $\mathbf{M}_i$ satisfies $\mathbf{M}_i = (c_i, j, l, m)$ with c_i the value of the threshold contained in the node, $j \in \{1, 2, \ldots, k\}$ the parent node, $l \in \{1, 2, \ldots, k\}$ the first child node and $m \in \{1, 2, \ldots, k\}$ the second child node. Of course, no split will occur in a leaf by constricution so you can put for instance $c_i = 0$ arbitrarily for any leaf (or entirely discard such nodes and don't even put them in N). What is expected of you at this point is to provide this classification tree T, implemented through the matrix $\mathbf{M}$ or any other way you like that allows to make prediction about new unlabelled data.

Remark: In practice, the threshold α should be determined by cross-validation by looking at the misclassification error to get an optimal classification tree but here, for the sake of simplicity this α is given to you as a lower bound on the number of observations in a node.

3. Now, given new observations $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$ and $\tilde{\mathbf{x}}_3$ such that they have values $v(\tilde{\mathbf{x}}_1) = 1.2$, $v(\tilde{\mathbf{x}}_2) = -4$ and $v(\tilde{\mathbf{x}}_3) = 3$, write a short script using your decision tree T to predict the labels of $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$ and $\tilde{\mathbf{x}}_3$. To do that, first determine in which leaf (node without children) $v(\mathbf{x}_i)$, $i = 1, 2, 3$ belongs, then predict its label by the most common label

value observed in the subset of observations present in said leaf.

3. Grading

There is no need to be extra efficient in your implementation of both the data tree structure and the algorithm that finds the optimal split c_i, $i = 1, \ldots, k$ even if a clean and efficient implementation will be rewarded. You will be mainly evaluated on two main points:

(1) Is the tree T that your algorithm built a proper tree? I.e., can we recover for every node its parent node, children nodes and the value c_i of the split it contains?
(2) Is the tree T a proper prediction tree allowing to make sensible prediction regarding the label of the new points $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$ and $\tilde{\mathbf{x}}_3$? I would encourage you to check that the predicted label you get in the end make sense with respect to the dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Important: don't forget to (briefly) document your code.