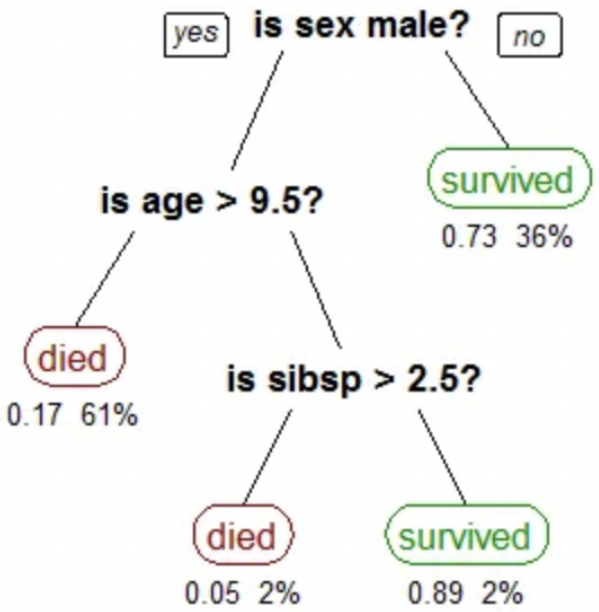
STAT562 Lecture 10: Decision Trees

Beidi Qiang

SIUE

Introduction



- Decision tree involves segmenting the feature space into a number of regions. The set of splitting rules used to segment the feature space is summarized in a tree.
- To make a prediction for a given example, we typically use the mean or the mode of the training examples in the region to which it belongs.
- Decision trees can be applied to both regression and classification problems .

Leaves, Branches and Nodes

- The regions hat doesn't split anymore are known as terminal nodes or leaves of the tree.
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as internal nodes.
- We refer to the segments of the trees that connect the nodes as branches.

Building a Tree

Process of building a tree:

- We divide the feature space (the values of $X_1, X_2, \cdots, X_p$) into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
- For every example that falls into the region R_j , we make the same prediction, which is simply the mean (regression) or the mode (classification) of the label values for the training examples in R_j.

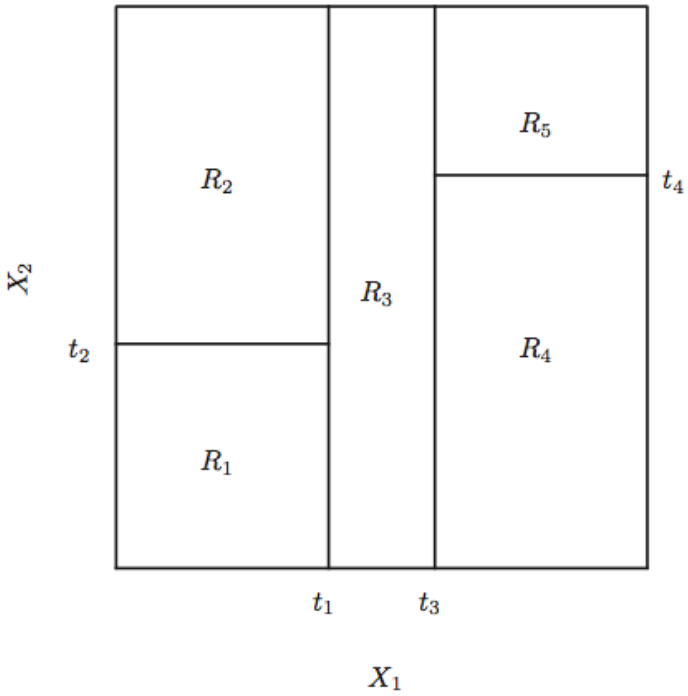
How do we choose regions $R_1, \cdots, R_J$?

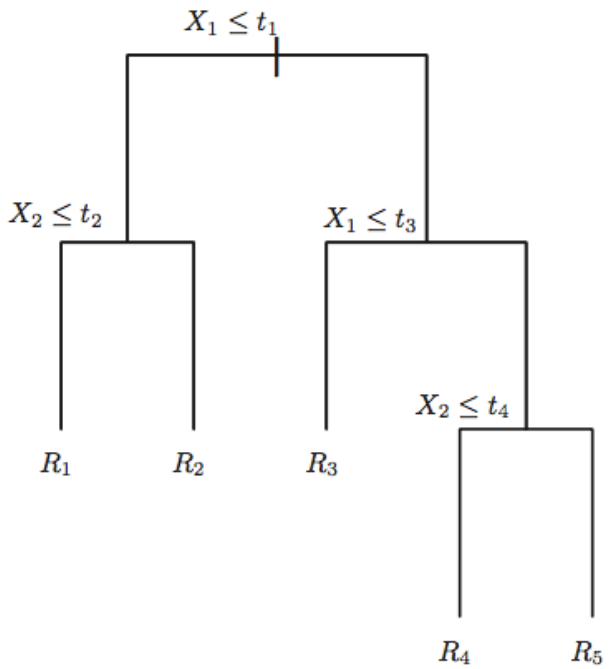
- The goal is to find boxes $R_1, \cdots, R_J$ that minimize the SSE.
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes. So we take a top-down approach that is known as "recursive binary splitting".

Algorithm: Recursive Binary Splitting

- We first select the predictor X_j and the cutpoint s such that splitting into the regions $R_1(j, s) = \{X|X_j < s\}$ and $R_2(j, s) = \{X|X_j \geq s\}$ leads to minimum cost.
- We repeat the process, looking for the best predictor and best cut point in order to split the data further so as to minimize the cost within each of the resulting regions (R_1 and R_2). We then split one of them.
- We now have three regions. Again, we look to split one of these three regions further, so as to minimize the cost.
- The process continues until a stopping criterion is reached.

An example of a five-region tree





Split Criterions for Trees

For Regression trees, we use

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

For Classification tree, we use one of the following:

- Gini index: $G = \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_k \hat{p}_{mk}^2$
- Cross-entropy: $D = -\sum_k \hat{p}_{mk} log(\hat{p}_{mk})$

where $\hat{p}_{mk}$ represent the proportion of training observations in the mth region that are from the kth class.

Comments

Advantages:

- Regression tree model is easier to interpret and has a nice graphical representation.
- Trees can easily handle categorical features directly without the need to create dummy variables.

Disadvantages:

- Complex tree without pruning is likely to overfit the data, leading to poor test set performance.
- Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.
- Decision trees can be unstable. Small variations within data can produce a very different decision tree.

Tree Pruning

We like a smaller tree with not too many splits. The post-pruning strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree. The following algorithm is called "Cost complexity pruning", also known as weakest link pruning.

- For a given tuning parameter α, we pick a subtree $T \in T_0$ that minimize the quantity $cost + \alpha|T|$, where $|T|$ is the number of terminal nodes of the tree. Here cost can be SSE in regression problems or deviance in classification problems.

$$deviance = -2\sum_n \sum_k n_{mk} log(\hat{p}_{mk})$$

- The tuning parameter controls a trade-off between the subtree complexity and its fit to the training data (similar to LASSO in linear regression).
- We can select a value of the tuning parameter α using a validation set or using cross-validation.

Summarized Algorithm to Build a Tree

- Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
- Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, for a sequence of values of α.
- Use K-fold cross-validation to choose α.
- Return the subtree that corresponds to the chosen α.

Example: Boston data

```
library(tree)

tree.b=tree(medv .,Boston)
summary(tree.b)
plot(tree.b);text(tree.b)

#pruning
cv.b=cv.tree(tree.b)
plot(cv.b$size, cv.b$dev, type="b")
prune.b=prune.tree(tree.b,best=5)
plot(prune.b);text(prune.b)
```

Example: Default Data

The tree library is used to construct classification and regression trees:

```
train=sample(1:10000,6000)
tree.d=tree(default .,Default,split="gini",subset=train)
summary(tree.d)
plot(tree.d);text(tree.d)

#pruning
cv.d=cv.tree(tree.d)
plot(cv.d$size, cv.d$dev, type="b")
prune.d=prune.tree(tree.d,best=5)
plot(prune.d);text(prune.d)

#predict class on test data
test=Default[-train,]
pred.d=predict(prune.d,test,type="class")
table(pred.d,test$default)
```