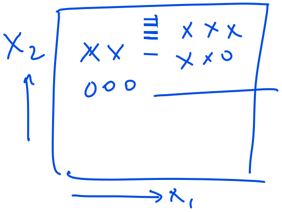
DSO530 Statistical Learning Methods

Lecture 7a: Decision Trees

Dr. Xin Tong
Department of Data Sciences and Operations
Marshall School of Business
University of Southern California
xint@marshall.usc.edu

Decision Trees



- Decision tress are supervised learning methods.
- They can be used for both regression and classification.
- They involve partitioning the predictor space into a number of simple regions (boxes, in particular).
- To make a prediction for a given observation, we typically use the mean (regression) or the mode (classification) of the training observations in the region to which it belongs.
- For binary classification problems, we can of course change the decision threshold from 1/2 to other values.

$\mathbb{1}(P(Y=1|X=x) > \text{threshold})$

Region 1 $3/5 > 1/7 \Rightarrow$ sample threshold.

Region 2 $1/6 > 1/7 \Rightarrow$ $\therefore$ still predict threshold. even w/ small

Decision Tree for Hitters Data

- For a given decision tree, making prediction is straightforward.

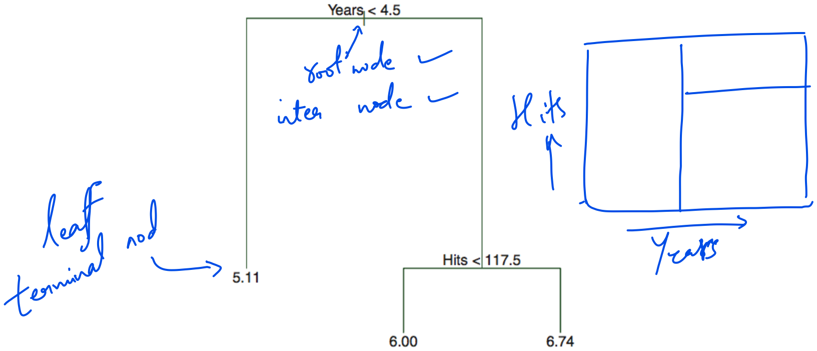


FIGURE 8.1. *For the* Hitters *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* Years<4.5*, and the right-hand branch corresponds to* Years>=4.5*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations*

Some terms for a tree

- Root: no parent node, two children nodes
- Internal node: two children nodes
- Terminal node (leaf): one parent node, no children node
- Branch: a segment of the trees that connect the nodes

At terminal nodes, we make predictions.

Some terms for a tree

- Root: no parent node, two children nodes
- Internal node: two children nodes
- Terminal node (leaf): one parent node, no children node
- Branch: a segment of the trees that connect the nodes

At terminal nodes, we make predictions.

The previous decision tree corresponds to a partition of the feature space

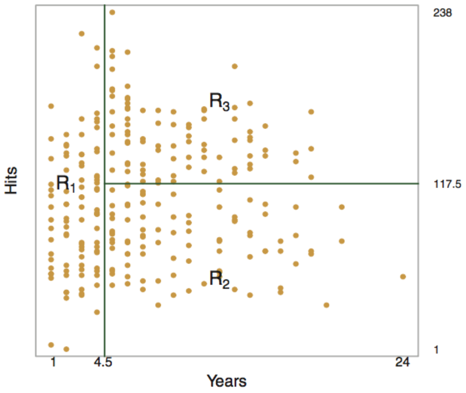


FIGURE 8.2. *The three-region partition for the* Hitters *data set from the regression tree illustrated in Figure 8.1.*

Figure 2: Feature space partition

Some thoughts after seeing ths first tree example

- In each split of the tree, we need to decide
  - which variable to split?
  - where do we split this variable?
- To answer the two above questions, we need some formal criteria
- In the least squares approach to linear regression, we used RSS as a criterion. Can we borrow it?
- Another question: we should think about a stopping rule. That is, when do we stop splitting? or else every terminal node will be 1 pt.
- Instead of the binary split, why didn't people split a variable into three parts in a step? Actually preferred sometimes, but easier to split in 2.
- Some people say once you split a variable, you should not use this variable in subsequent splits. Do you agree with this comment? If not, can you name some situations in which we'd better allow splitting a variable multiple times. multiple times

Decision tree for regression

- The RSS analog for decision tree for regression involves two steps:
  - We divide the feature space (i.e., the set of possible values for $X_1, X_2, \cdots, X_p$) into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
  - For every observation that falls into the region R_j , we make the same prediction, which is simply the mean of the response values for the training observations in R_j.
- But how to find regions $R_1, \cdots, R_J$? The goal is to find boxes $R_1, \cdots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 ,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

How to do the partition?

- It is computationally infeasible to consider every possible partition of the feature space into J boxes.
- We take a top-down, greedy approach that is known as *recursive binary splitting*.
- It is a *greedy* approach because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
- For any j and s, we define the pair of half-planes by

$$R_1(j,s) = \{X | X_j < s\}, \text{ and } R_2(j,s) = \{X | X_j \geq s\},$$

and set the values of j and s that minimize the equation

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

where $\hat{y}_{R_1}$ is the mean response for the training observations in $R_1(j,s)$, and $\hat{y}_{R_2}$ is the mean response for the training observations in $R_2(j,s)$.

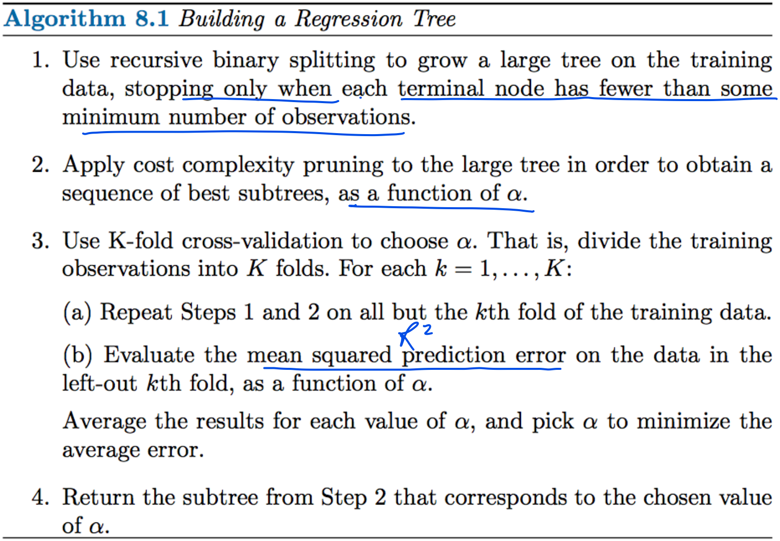
- The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.
- Note that it is very tempting to stop where the reduction in RSS for the best split fall below a threshold. But this criterion is too greedy as a mediocre split might lead to the a very good split down the road.
- But when we grow a very deep and bushy tree T_0, we have overfitted the training data (this tree has high variance or high bias?). To solve the problem:
- *Cost complexity pruning* (weakest link pruning): consider a sequence of trees indexed by a nonnegative tuning parameter α (how to tune it?). For each value of α, there exists a substree $T \subset T_0$ such that

  no. of terminal nodes

$$\sum_{m=1}^{\boxed{|T|}} \underbrace{\sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2}_{\text{RSS}} + \underbrace{\alpha}_{\text{tuning parameter, if too ~~big~~ then not to prune}} \boxed{|T|}$$

  is as small as possible. (Does this remind you LASSO and Ridge?)
- As α increases, we get a sequence of nested trees
- The next algorithm summarizes the entire tree building process

DecisionTreeRegressor in sklearn for implementation



Algorithm 8.1 *Building a Regression Tree*

1. 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.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   R^2

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Trees for classification

$k=2$, $m+n$

$$\begin{array}{|cc|} 11 & 22 \\ 11 & 22 \\ 11 & \end{array}$$

$$\frac{6}{10}\left(1-\frac{6}{10}\right) + \frac{4}{10}\left(1-\frac{4}{10}\right)$$

$= 0.48$ {not complicated for binary classification}

Gini Index

- RSS is not a proper criterion for classification problems
- The most natrual and intuitive substitue is the classification error
- But it turns out that classification error is not sufficiently sensitive for tree-growing
- In practice people use $\xrightarrow{}$ k is total # of classes.
  - *Gini index* $G = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$
  - *entropy* $D = -\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk}$
  - where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class, and K is the total number of classes
- The gini index is slightly faster to compute and is the default criterion used in the `DecisionTreeClassifier` model of scikit-learn
- Both Gini index and entropy measure *node purity*
- When $\boxed{K=2}$, what is the maximum value for Gini index? what is the smallest?

$\hookrightarrow$ binary classification.

If node is pure the GI = 0

$\rightarrow$ Answer:

	Class 1	class 2
$\hat{p}_{mk}$	x	$1-x$
$1-\hat{p}_{mk}$	$1-x$	$1-(1-x)=x$

$\therefore$ Gini Index $= x(1-x)+(1-x)x$
$= 2x(1-x)$

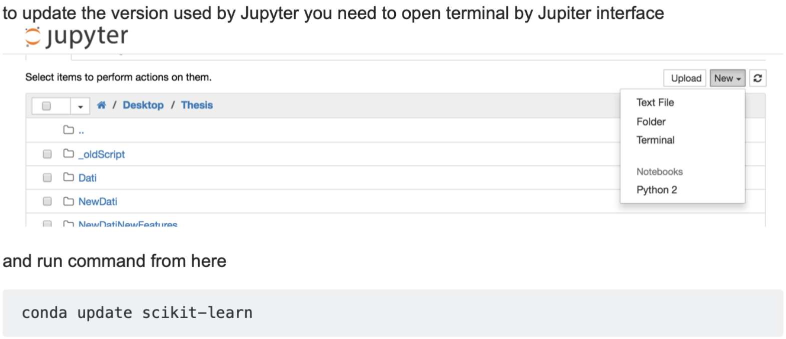
min at 0 & max at 1/2

Advantages and disadvantages of trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Why? No mathematical formula needed in the communication
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous lectures.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Unfortunately, trees generally do not have the great predictive accuracy.
- This disadvantage motivates the ensemble methods such as *bagging*, *random forests* and *boosting*.

Update your sklearn if the version is older than 0.22

- Cost-complexity pruning did not work with older sciki-learn


to update the version used by Jupyter you need to open terminal by Jupiter interface

and run command from here

```
conda update scikit-learn
```

Figure 3: Update instruction