Tree-based Models for Regression

- Regression trees
- Regression forests
  - `randomForest` based on bagging
  - `gbm` based on boosting

Tree-based Models for Regression

- Input vector $X = (X_1, X_2, \ldots, X_p) \in \mathcal{X}$
- Response variable $Y \in \mathbb{R}$
- Trees are constructed by recursively splitting regions of $\mathcal{X}$ into two sub-regions, beginning with the whole space $\mathcal{X}$.

  For simplicity, focus on recursive binary partitions.
- `R` page: check the fitted regression tree on `BostonHousingData` based on two features `lon` and `lat`.

- Notation: node (t), child node (t_L, t_R), split (var j, value s), leaf/terminal node.

- Every leaf node (i.e. a rectangle region R_m in $\mathcal{X}$) is assigned with a constant for regression tree

$$\hat{f}(X) = \sum_m c_m I\{X \in R_m\}.$$

Advantages of Trees

- Easy to interpret

- Variable selection and interactions between variables are handled automatically

- Invariant under any monotone transformation of predictors

How to Build a Tree?

Three elements:

1. Where to split?

2. When to stop?

3. How to predict at each leaf node?

Prediction at Leaf Nodes

Each leaf node (corresponds to region R_m) contains some samples.

Assign the prediction for a leaf node to be the average (of the response variable Y).

$$\hat{f}(X) = \sum_m c_m I\{X \in R_m\}.$$

$$\min_{c_m} \sum_{i=1,\ x_i \in R_m}^{n} (y_i - c_m)^2,$$

$\Longrightarrow c_m =$ average of y_i's whose $x_i \in R_m$

Where to Split?

- A split is denoted by (j, s): split the data into two parts based on whether "var j < value s".
- For each split, define a split criterion $\Phi(j, s)$
  - deduction of RSS for regression
- Trees are built in a top-down greedy fashion. Start with the root: try all possible variables $j = 1 : p$ and all possible split values[a], and pick the best split, i.e., the split having the best Φ value. Now, data are divided into the left node and right node. Repeat this procedure in each node.

[a] For each variable j, sort the n values (from n samples), and choose s to be a middle point of two adjacent values. So at most $(n - 1)$ possible values for s.

Goodness of Split $\Phi(j, s)$

For Regression tree, we look at the deduction of RSS if we split samples at node t into t_R and t_L:

$$\Phi(j, s) = \text{RSS}(t) - \Big[\text{RSS}(t_R) + \text{RSS}(t_L)\Big],$$

where

$$\begin{aligned} \text{RSS}(t) &= \sum_{x_i \in t} (y_i - c_t)^2, \\ c_t &= \text{AVE}\{y_i : x_i \in t\}. \end{aligned}$$

Note that $\Phi(j, s)$ is always positive if we split the data into two groups (even randomly), unless the mean of the left node and the one of right node are the same.

Issues: Split Categorical Predictors

- For a categorical predictor with m levels, there are $2^{m-1} - 1$ possible partitions of the m labels into two groups.
- However, for regression with square error, the computation simplifies: order the m levels by their mean values of Y, and then split the categorical variable as if it were an ordered predictor — there are only $(m-1)$ potential splits.

Issues: Missing Predictor Values

- Discard any observation with missing values $\longrightarrow$ serious depletion of the training set.
- Splitting criteria are evaluated on non-missing observations.
- Once a split (j, s) is determined, what to do with observations missing X_j?

- Find surrogate variables that can predict the binary outcome “$X_j < s$” and “$X_j \geq s$” using a one-split tree.

- Rank those surrogate variables along with the blind rule “go with majority”.

- Any observation that is missing X_j is then classified with the first surrogate variable, or if missing that, the second surrogate variable (or the blind rule) is used, and etc.

When to Stop?

- A simple one : stop splitting at a node if the gain from any split is less than some pre-specified threshold.
- BUT, this is short-sighted.
- Another strategy: grow a large tree and then prune it (i.e., cut some branches).

Preliminaries for Pruning

First, grow a vary large tree $T_{\max}$

1. until all terminal nodes are nearly pure;

2. or when the number of data in each terminal node is less than certain threshold;

3. or when the tree reaches certain size.

As long as the tree is sufficiently large, the size of the initial tree is not critical.

Notation : *subtree* $T' \prec T$, *branch* T_t.

Minimum Complexity-cost Pruning

For any subtree $T \prec T_{\max}$, define the Complexity-cost

$$R_\alpha(T) = R(T) + \alpha|T|, \tag{1}$$

- $R(T)$: RSS for regression tree T

- $|T|$: tree size, i.e., the number of leaf nodes

- $\alpha > 0$: cost (penalty) of adding a split

Questions: *i)* How to minimize (1) for a given α? *ii)* How to choose α?

Pick the best subtree that minimizes the cost

$$T(\alpha) = \text{argmin}_{T \preceq T_{\max}} R_\alpha(T) = \text{argmin}_{T \preceq T_{\max}} \left[R(T) + \alpha |T| \right]$$

$T(\alpha)$ may not be unique.

Define the optimal subtree $T^*(\alpha)$ to be the smallest one among $T(\alpha)$'s

(1) $R_\alpha(T^*(\alpha)) = \min_{T \preceq T_{\max}} R_\alpha(T)$.

(2) $T^*(\alpha) \preceq$ any $T(\alpha)$.

$T^*(\alpha)$ is unique.

$$R_\alpha(T) = R(T) + \alpha|T|$$

Some Facts

- For a pair of leaf nodes (t_L, t_R), there exists α^*, such that

  1. for any $\alpha \geq \alpha^*$, we would like to collapse them to just node t;
  2. for any $\alpha < \alpha^*$, keep the two leaf nodes.

  That is, α^* is the maximal price we would like to pay to keep that split.

  Next we extend this calculation to compute the maximal price we would like to pay to keep a branch T_t.

- For any non-leaf node t, do the following calculation to find out the maximal price we'd like to pay for keeping the whole branch T_t.

  Focus only on samples at node t.

  - Cost for keeping branch T_t: $R_\alpha(T_t) = R(T_t) + \alpha|T_t|$
  - Cost for cutting branch T_t: $R_\alpha(\{t\}) = R(\{t\}) + \alpha$
  - Calculate

$$\alpha^* = \frac{R(\{t\}) - R(T_t)}{|T_t| - 1}.$$

  That is, if the given $\alpha > \alpha^*$, then it is too expensive to keep this branch and we would like to cut the whole branch and make t a leaf node.

Weakest-Link Pruning

The weakest-link pruning algorithm.

- Start with $T_0 = T_{\mathsf{max}}$ and $\alpha_0 = 0$.
- For any non-leaf node t, denote the maximal price we'd like to pay to keep T_t by $\alpha(t)$.
- $\alpha_1 = \min_t \alpha(t)$. The corresponding (non-terminal node) t_1 is called the weakest link. Cut the branch at t_1.
- Next update the maximal price for each non-leaf node (we only need to recompute the maximal price for nodes that are parents/grandparents of t_1). Find α_2 and cut the branch at the 2nd weakest link. Keep doing this until we get to the root.

The steps above generate a Solution Path:

$$T_{\mathsf{max}} = T_0 \succ T^*(\alpha_1) \succ T^*(\alpha_2) \succ \cdots \succ \{\text{root node}\}$$

$$0 = \alpha_0 < \alpha_1 < \alpha_2 < \cdots$$

All possible values of α are grouped into $(m+1)$ intervals:

$$\begin{aligned} I_0 &= [0, \alpha_1) \\ I_1 &= [\alpha_1, \alpha_2) \\ &\vdots \\ I_m &= [\alpha_m, \infty) \end{aligned}$$

where all $\alpha \in I_i$ share the same optimal subtree $T^*(\alpha_i)$.

Cross-validation

How to Choose α? K-fold Cross-validation (`rpart`):

1. Fit a big tree T_{max} and compute $I_0, I_1, \ldots, I_m$

$$\begin{aligned} \text{Set } \beta_0 &= 0 \\ \beta_1 &= \sqrt{\alpha_1 \alpha_2} \\ &\vdots \\ \beta_{m-1} &= \sqrt{\alpha_{m-1}\alpha_m} \\ \beta_m &= \infty \end{aligned}$$

where each β_j is a ‘typical value’ for its interval I_j.

2. Divide data into K groups and repeat $k = 1, \ldots, K$:

   - Fit a full model on the data set except the k-th group and determine the optimal subtrees:

   $$T_0 \succ T^*(\beta_1) \succ \cdots \succ T^*(\beta_m) \succ \{\text{root node}\}$$

   - Compute the prediction error on the k-th group for each tree models.

3. Produce the CV plot over different α values and pick the optimal α_{min} or α_{1se}.