8 Tree-Based Models

Andrew Liang

11/22/2020

Notes

- Useful for interpretation, but typically not competitive with best supervised learning approaches in terms of prediction

Basics of Decision Trees

Regression Trees

Process of Building a Regression Tree:

1. Divide the predictor space (set of possible values of $X_1, X_2, \ldots, X_p$ into J distinct and non-overlapping regions $R_1, R_2, \ldots, R_J$
2. For every observation in theR_j region, we make the same prediction, which is simply the mean of the response values for the training observations in R_j

- In step 1, we construct $R_1, R_2, \ldots, R_J$ such that the predictor space is divided into high-dimensional rectangles, or *boxes*, for ease of interpretation
  - goal is to minimize 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.

- Another way to look at it is that we consider all predictors $X_1, X_2, \ldots, X_p$ and all possible values of cutpoint s for each of the predictors, then choose the predictor and cutpoint such that the resulting tree has lowest RSS
  - for any j and s, we define pair of half-planes:

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

and we seek to value of j and s to minimize:

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

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

Once the regions $R_1, R_2, \ldots, R_J$ have been created, we can then predict response for a given test observation using mean of train obs in the region to which that test obs belongs

Tree Pruning

- Process above may likely overfit data as the resulting tree may be too complex
  - smaller tree with fewer splits can lead to lower variance and better interpretation at the cost of a little bias
- Better strategy may be to grow a very large tree T_0, and then *prune* it back to get a subtree
  - want to get a subtree that leads us to the lowest test error rate
  - however, using CV for every subtree may be infeasible, so we need to select a small set of subtrees
- can use *cost complexity pruning*, consider a sequence of trees indexed by a nonnegative tuning parameter α
  - for each value of α, there corresponds a subtree $T \subset T_0$ such that it minimizes:

$$\sum_{m=1}^{|T|} \sum_{i:x \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

where $|T|$ indicates the number of terminal nodes of tree T, and R_m is the region corresponding to the mth terminal node

- as the number of terminal nodes increases, there is a penalty α, so the above quantity will tend to be minimzied for a smaller subtree
  - select α using CV

Algorithm for building Regression Trees

1. Use recursive binary splitting to grow a large tree on training data, stopping when each terminal node has fewer than some minimum # of observations
2. Apply cost complexity pruning to large tree in order to obtain a sequence of best subtrees as a function of α
3. Use K-fold CV to choose α. Divide training observations into K fold. For each $k = 1, \ldots, K$:
   - repeat steps 1 and 2 on all but kth fold of training data
   - evaluate mean squared prediction error on data in left-out kth fold, as a function of α
   - average out the results for each value of α, and pick α to minimize error
4. Return the subtree from step 2 that corresponds to chosen value of α

Classification Trees

- very similar to regression tree, but predicts qualitative response instead
  - predict that each observation belongs to the *most commonly occuring class* or training observations in the region to which it belongs
- also interested in the *class proportions* among training observations that fall into that region
- also use recursive binary splitting to grow a classification tree, but instead of RSS, *classification error rate* is used as the criterion for making the binary splits
  - simply the fraction of training obs in that region that don't belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training observations in th mth region that are from the kth class. In practice however, two other measures are preferable:

- The *Gini index*:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

is a measure of total variance (of which we want to minimize) across the K classes. G takes on a small value if $\hat{p}_{mk}$ is close to zero or one. It is a measure of *node purity* - a small value indicates that a node contains predominantly observations from a single class

- The *Entropy*:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

similarly to the Gini, will take on a value near zero if all $\hat{p}_{mk}$ are near zero or one. Both measurements are quite similar numerically

- generally, classification error rate is preferable if prediction accuracy is the goal of the final pruned tree

Trees vs Linear Models

- if relationship between features and response is well approximated by a linear model, then a linear regression would outperform trees
- if there is a highly non-linear and complex relationship between features and response, then trees may outperform linear regression

Pros of Trees

- very easy to explain
- may closely mirror human decision-making more than regression and classification approaches in previous methods
- displayed graphiccaly and easily interpreted
- can handle qualitative predictors without the need to create dummy variables

Cons of Trees

- generally don't have same level of predictive accuracy as other regression and classification techniques
- tend to be very non-robust, small change in data can cause large change in the tree

Bagging, Random Forests, Boosting

Bagging

- in order to solve the high variance problem in trees, *bagging* can help reduce it through boostrapping procedures
  - recall in bootstrap, given n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2
    - variance of mean $\overline{Z}$ is given by $\frac{\sigma^2}{n}$, thus has a lower variance
    - essentially we use repeated samples from our original training data to create B different training sets:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

which is called bagging

- in the context of regression trees, we construct B different trees using B boostrapped training sets, then avg the resulting predictions
  - avg out all the trees reduces the variance
- in classification trees, for a given test observation, we can record the class predicted by each of the B tres, and take a *majority vote*, which is the overall prediction most commonly occurring among all the B predictions
- important to note that using a large B will not lead to overfitting
  - generally use $B = 100$ to achieve sufficient performance

Out-of-Bag Error Estimation

- very straightforward way of estimating test error of a bagged model, without the need of CV
- on avg, each bagged tree makes use around two-thirds of observations (sampling with replacement of training set)
  - remaining one-third of observations not used are the *Out-of-Bag* (OOB) observations
  - can predict response for ith observation using each of the trees in which that observation was OOB
    - yields around $B/3$ predictions for ith observation
  - with B sufficiently large, OOB error is essentially the same as LOOCV
    - convenient when perform CV would be computationally infeasible

Variable Importance Measures

- bagging improves prediction accuracy at the expense of interpretability
  - can obtain summary of important predictors using RSS or Gini index
    - for regression, record the total amount the RSS is decreased due to splits over a given predictor
    - for classification, sum the total amount that the Gini index is decreased by splits over a given predictor

Random Forests

- improves over bagging through docorrelation of the trees
- similar to bagging, we build a number of trees based on bootstrapping, but now we choose a *random sample of m predictors* as split candidates from the full set of p predictors
  - split is only allowed to use one of those m predictors
    - a fresh sample of m predictors chosen at each split
  - typically choose $m \approx \sqrt{p}$
  - thus at each split, the algorithm is not even allowed to consider a majority of the available predictors
    - helps *decorrelate* trees, to prevent the domination of one strong predictor on all the trees
  - when $m = p$, then the process is just bagging, hence bagging is a special case of random forest

Boosting

- boosting works similarly to bagging, but now each tree is grown *sequentially*
  - uses info from previous tree
  - doesn't use bootstrap, instead tree is fit on a modified version of original data

Algorithm for Boosting

1. Set $\hat{f}(s) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, 2, \ldots, B$, repeat:

- Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to training data (X, r)
- Update $\hat{f}$ by adding shrunken version of new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

- Update the residuals:

$$r_i \leftarrow r_i + \lambda \hat{f}^b(x)$$

3. Output the boosted model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

- the idea is to slowly improve $\hat{f}$ in areas where it doesn't perform well
- shrinkage λ slows the process down, allowing more and different shaped trees to attack residuals

Boosting has three tuning parameters:

1. Number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, hence we use CV to choose B

2. shrinkage parameter λ, a small positive number and controls rate at which boosting learns (typical values are 0.01 or 0.001). Very small λ can require large B to achieve good performance

3. Number of splits d in each tree. Controls complexity of boosted ensemble, and often $d = 1$ works well, where each tree is a *stump* of a single split. When $d = 1$, boosted ensemble is fitting an additive model, since each term involves only one variable. Generally, d is the *interaction depth*

- because growth of a tree depends on previous trees, smaller trees are typically sufficient
  - smaller trees can aid interpretability