Lecture 20 Tree-based Methods IV: Boosting and Others

ECE 625: Data Analysis and Knowledge Discovery

Di Niu

Department of Electrical and Computer Engineering
University of Alberta

March 25, 2021

Outline

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees.
- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees.

Boosting algorithm for regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, \cdots, B$, repeat:
   b=2
   2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r). r = (r1, ..., rn)
   2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   key thing to boosting: d must be small
   2.3 Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Boosting Idea

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting algorithm for classification trees

1. Set $\hat{f}_k(x) = 0$ for all $k = 1, \cdots, K$.
2. For $b = 1, \cdots, B$, repeat:
   - 2.1 Set $p_k(x) = e^{\hat{f}_k(x)} / \sum_{k=1}^{K} e^{\hat{f}_k(x)}$, for $k = 1, \cdots, K$.
   - 2.2 For $k = 1, \cdots, K$:
     - 2.2.1 Compute $r_{ik} = y_{ik} - p_k(x_i)$, where y_{ik} is the class indicator variable for the ith subject with values of 0 or 1.
     - 2.2.2 Fit a tree $\hat{f}_k^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r_k). r_k = {r_ik}.
     - 2.2.3 Update $\hat{f}_k$ by adding in a shrunken version of the new tree:

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

3. Output the boosted model, $\hat{f}_k(x) = \sum_{b=1}^{B} \lambda \hat{f}_k^b(x)$, and assign x to class k subject to $k = \arg\max_k p_k(x) = e^{\hat{f}_k(x)} / \sum_{k=1}^{K} e^{\hat{f}_k(x)}$.

Gradient Boosting Decision Trees

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

   loss gradient at the prev prediction

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   fit to gradients instead of residual

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute

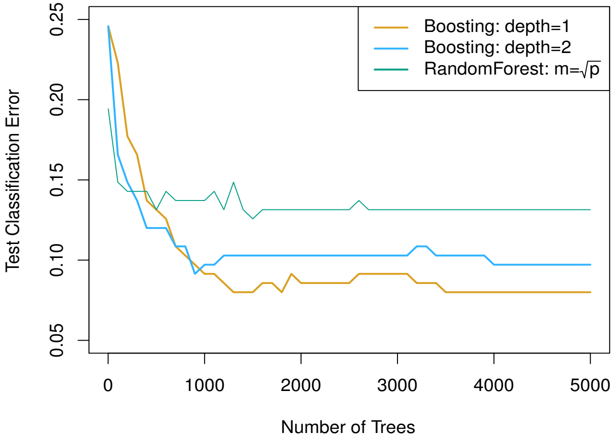
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

   gradient descent

   prev prediction delta
3. Output $\hat{f}(x) = f_M(x)$.

Gene Expression Data



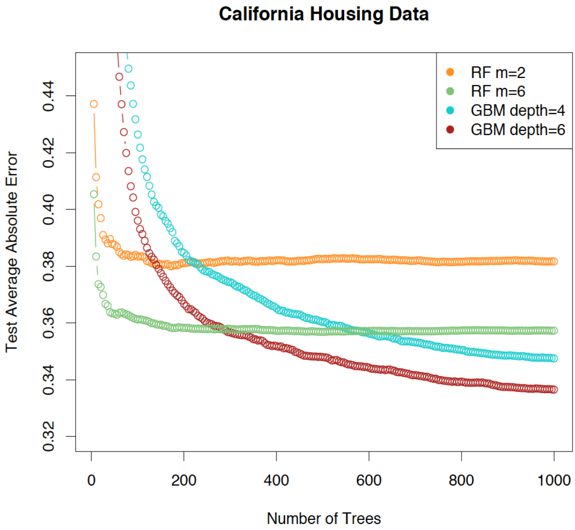
Gene Expression Data

- The R package gbm (gradient boosted models) handles a variety of regression and classification problems.
- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal.
- The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tree is 24%.

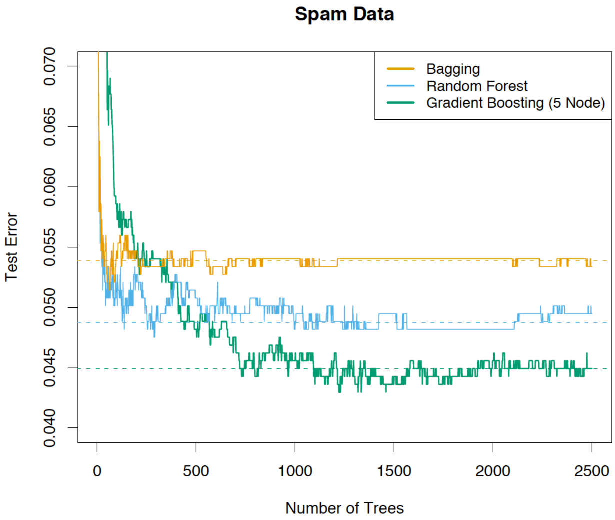
Tuning parameters for boosting

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
- The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

California Housing Data

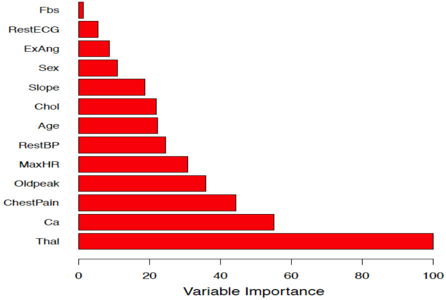


Spam Data


Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees

Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.
- Variable importance plot for the Heart data



Summary

- Decision trees are simple and interpretable models for regression and classification.
- However they are often not competitive with other methods in terms of prediction accuracy.
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods — random forests and boosting — are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Summary and Remark

- Boosting
- Other issues
- Read textbook Chapter 10 and `R` code
- Do `R` lab