BANA 7046 Data Mining I
Lecture 5. Tree-Based Methods

Shaobo Li
University of Cincinnati

\[\text{\textsuperscript{1}}\text{Partially based on Hastie, et al. (2009) ESL, and James, et al. (2013) ISLR}\]
Overview

- Supervised learning
- Simple for interpretation
- Nonparametric method

We will talk about
- Classification and Regression Tree (CART)
- Bagging, Random Forest, and Boosting
A Simple Example of Regression Tree

- Predicting baseball player’s log-salary using
  - X1: number of years he has played in the major leagues;
  - X2: number of hits he made in previous year
The values in terminal nodes
Top-down
At each splitting point, go left if TRUE
Figure: Salary is color-coded from low (blue, green) to high (yellow, red)
Data Partition — Stratification
Classification and regression tree
Binary splitting for variable $j$ at point $s$.
Top-down and Greedy algorithm
- Start from the top of tree
- The split is optimal at current step

Question:
- How to determine the optimal splitting point?
- What criteria should we use?
- When to stop growing the tree?
For each split, we try to determine two regions $R_1$ and $R_2$ by finding the optimal splitting point $X_j = s$, that is

$$R_1 = \{ \text{Data} | X_j \leq s \} \quad \text{and} \quad R_2 = \{ \text{Data} | X_j > s \}$$

This is done by scanning all possible splitting point for each $X$.

Such optimal splitting point minimizes residual sum squares

$$\sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

where $\bar{y}_{R_1}$ and $\bar{y}_{R_2}$ are average of $Y$ in each region.

Repeat this process for each region until the decrease of RSS is not significant.
Stopping Strategy — Cost-Complexity Pruning

- To avoid overfitting!
- Suppose \( T_0 \) is a very large tree (overfitted tree). Denote \( T \) as a subtree of \( T_0 \), and \(|T|\) as the size of that tree (number of terminal nodes). We minimize the cost complexity criterion

\[
C_\alpha(T) = \sum_{m=1}^{\frac{|T|}{\alpha}} RSS_m + \alpha |T|,
\]

where \( RSS_m \) is residual sum squares in region (terminal) \( m \).

- Larger \( \alpha \) penalizes the tree size (\(|T|\)) more
  - Recall how do we use AIC/BIC for variable selection
  - Every \( \alpha \) corresponds to a unique optimal tree \( T_\alpha \).
- We use cross-validation to choose the optimal \( \alpha \).
- \( cp \) is the complexity parameter \( \alpha \) we have discussed.
Classification Tree

- Outcome is categorical variable
- The same idea as for regression tree
- The only difference is the splitting criteria
- What should we use for the criteria?
Splitting Criteria for Classification Tree

- Misclassification rate
- Gini index

\[ \text{Gini index} = \sum_{k \neq k'} \hat{p}_k \hat{p}_{k'} = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k) \]

where \( \hat{p}_k \) is the proportion of \( k \) category observation in one node.

- Entropy

\[ \text{Entropy} = - \sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k \]

For each split, gini index (or entropy) is calculated for two regions, and the weighted average is used as the criterion to be minimized.
Example

- Suppose a node contains binary outcome (0 or 1) with each 400 observations.
- Split strategy 1: (300, 100) and (100, 300)
- Split strategy 2: (400, 200) and (0, 200)
- What is the misclassification rate, Gini index, and Entropy for each split strategy?
Categorical Predictor

- If a categorical predictor has \( q \) possible values, how many possible splits?
- Computationally infeasible for large \( q \).
- Solution for binary outcome:
  - Sort the \( q \) classes according to the proportion of "1" in each of the \( q \) classes.
  - Split the ordered variable as if it is numeric variable.
  - This split is the optimal in terms of Gini index and Entropy.
- But we should avoid categorical variable which has too many categories. Why?
Asymmetric Cost

- Similar idea as we discussed in last lecture
- Define a loss matrix with different weights in misclassified cells.

<table>
<thead>
<tr>
<th></th>
<th>Pred=1</th>
<th>Pred=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True=1</td>
<td>0</td>
<td>$w_0$</td>
</tr>
<tr>
<td>True=0</td>
<td>$w_1$</td>
<td>0</td>
</tr>
</tbody>
</table>

- Loan application example (1=default): $w_1 < w_0$
- The weights are incorporated during each split
Bagging

- Limitation of a single tree: high variance, unstable
  - Change of points in the sample would lead to different split.
  - Current split heavily relies on previous splits.

- Bagging — Bootstrap aggregation

\[ \hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x) \]

- Leo Breiman (1996)
- Fit many trees on bootstrap samples, and then take average.
- Variance can be significantly reduced

- For classification tree, how to take average?
Out-of-bag (OOB) Prediction

- Similar to cross-validation
- A type of testing error — to prevent overfitting
- For every bootstrapped sample, it is used as training sample, while the rest is used as testing sample. OOB is the averaged error of such testing sample.
Random Forests

- **Improvement of Bagging**
  - The trees in Bagging are correlated to each other
  - It may add variance and prediction error

- **Goal**: decorrelate a series of trees while maintain strength

- **How RF works**: randomly choose $m$ predictors as candidate splitting variables for each split

- **Choice of $m$**
  - Smaller $m$ decreases the correlation, but also decreases strength
  - Larger $m$ increases the correlation, but also increases strength
  - Use OOB error to determine the optimal $m$
  - By default, $m \approx \sqrt{p}$ for classification and $p/3$ for regression

See the inventor’s (Leo Breiman) [website](http://www.stat.berkeley.edu/~breiman/) for more details.
Illustration of De-correlation
Change of Testing Error across $m$
Boosting Regression Tree

- Additive: sequentially grow the tree
- Combination of many small trees
- Boosting for regression tree
  1. Set $\hat{f}(x) = 0$, and $r_i = y_i$
  2. For $b = 1, \ldots, B$, repeat:
     - Fit a small tree $\hat{f}^{(b)}(x)$ with $d$ splits based on the training sample
     - Update residual $r_i \leftarrow r_i - \lambda \hat{f}^{(b)}(x)$, and treat this residual as new response, where $\lambda$ is shrinkage parameter
     - Update the tree $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{(b)}(x)$
  3. Output the boosted regression tree

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^{(b)}(x)$$
AdaBoost — Boosting for Classification

- Originally designed for classification \( Y = \{-1, 1\} \)

**Algorithm:**

1. Assign equal weights to all observations \( w_i = 1/N \)
2. For \( b = 1, \ldots, B \), repeat:
   - Fit a classifier \( G_b(x) \) to the training sample using weights \( w_i \)
   - Compute weighted misclassification error
     \[
     err_b = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_b(x_i))}{\sum_{i=1}^{N} w_i}
     \]
   - Compute \( \alpha_b = \log((1 - err_b)/err_b) \)
   - Update weights \( w_i \leftarrow w_i \exp\{\alpha_b I(y_i \neq G_b(x_i))\} \)
3. Output \( G(x) = \text{sign}\{\sum_{b=1}^{B} \alpha_b G_b(x)\} \).
Performance Comparison

Data Mining I

Lecture 5. Decision Tree

The graph shows a comparison of different models over the number of trees. The models compared are Bagging, Random Forest, and Gradient Boosting (5 Node). The y-axis represents test error, and the x-axis represents the number of trees. The graph illustrates how each model's test error changes as the number of trees increases.