Notes on Decision Tree and Random Forest

Shasha Liao
Georgia Tech

November 20, 2020

Contents

1 CART: Classification And Regression Tree (Breiman et al. 1984) 2

2 Regression Tree 2

3 Tree pruning 3

4 Classification Tree 3
4.1 Different measures of errors $Q_j(J)$ 4

5 Pros and Cons of decision tree 4

6 Random forest for regression and for classification 5

7 Bagging 5

8 Bootstrap 5

9 Random forest 6

10 OOB error 7

11 Boosting trees 7

12 Forward stagewise optimization for boosting trees 7

13 Three tuning parameters of Boosting 8

- Decision Trees are a non-parametric supervised learning method for classification and regression.
- It predicts a target variable by learning simple decision rules inferred from the data features.
- Cons: simple and easy to interpret
- Pros: can be noisy and overfit to data

1 CART: Classification And Regression Tree (Breiman et al. 1984)

- Partition the feature space into a set of rectangles
- Decision rule can be represented by a tree (binary tree most common)
- Regression tree: Fit a simple model (e.g., a constant) for each rectangle
- Classification tree: Majority vote for samples in the rectangle

2 Regression Tree

- Restrict attention to recursive binary partitions
- First split the space into two regions and model the response by the mean of Y in each region.
- Choose the variable and split-point to achieve the best fit
- Then one or both of these regions are split into two more regions
- This process is continued until some stopping rule is applied
- Steps:
  - Greedy approach: constructing decision trees "top-down"
  - Start with all the samples
  - Choose a variable at each step that best split the region by trying all the possible options and choose the one that reduces the error the most.
    - Splitting a variable j
    - Splitting position s

* Define half-spaces:

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

Solve the problem for optimal splitting and "regression":

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2].$$

Once j, s are decided, optimal prediction is simple

$$\hat{c}_1 = ave(y_i|x_i \in R_1(j,x)), \hat{c}_2 = ave(y_i|x_i \in R_2(j,x)).$$

3 Tree pruning

- How large should a tree be?
  Large tree may overfit data; Small tree underfits and dose not not capture important structure.

- Determining tree depth by controlling the data-fit complexity tradeoff:
  Grow a large tree, stop when a minimum node size has reached, then performing pruning by minimizing the cost function

$$C_\alpha(S) = \sum_{j=1}^{|J|} \sum_{x_i \in R_j} (x_i - \hat{c}_j)^2 + \alpha|J|$$

  where

  - the first term measures data-fitting and the second term controls the complexity of the model
  - $\sum_{x_i \in R_j}(x_i - \hat{c}_j)^2$ is the data fitting error for the jth node
  - J is the set of terminal nodes in the decision tree, or the regions in the feature space
  - $\alpha > 0$ is the regularization parameter, it can be decided using cross-validation

4 Classification Tree

- Classification outcome takes values $1, 2, ..., K$

- Need different criterion for splitting node and pruning tree

- $$\hat{p}_{jk} = \frac{1}{N_j} \sum_{x_i \in R_j} \mathbb{I}(y_i = k)$$
$$k(j) = \text{argmax}_k \hat{p}_{jk}$$
where $k(j)$ is the result of the majority voted class and N_j is the number of samples in region R_j.

- Tree pruning
$$C_\alpha(J) = \sum_{j=1}^{|J|} N_j Q_j(J) + \alpha |J|,$$
where $N_j Q_j(J)$ measures the error for the j-th node.

4.1 Different measures of errors $Q_j(J)$

- Misclassification error:
$$\frac{1}{N_j} \sum_{x_i \in R_j} \mathbb{I}(y_i \neq k(j)) = 1 - \hat{p}_{jk}.$$

- Gini index:
$$\sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \hat{p}_{jk} \hat{p}_{jk'} = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$$

- Cross-entropy
$$-\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}$$
* These measures measure the purity of the region.
* For example, if a region only contains one class of samples, the Gini index in that region will be 0.
* However, if a region contains more than two classes of samples and they have equal number of samples, then the Gini index will be large.
* Gini index and Cross-entropy are differentiable measures of errors, which is good for training the model using Gradient Descent or Stochastic Gradient Descent.

5 Pros and Cons of decision tree

- Trees (if grown large enough) can capture complex structures
- A main issue of the tree-based method is noisy outcome
- They benefit from averaging

6 Random forest for regression and for classification

- **Bootstrap** samples from training data
- Grow a tree for each batch of bootstrap samples
- Regression: average tree's predictions
- Classification: take majority vote across trees

Random forest is basically a bagging approach, another way of combining weak learners. (We also talked about Boosting)

7 Bagging

- Run weak learners on bootstrap replicates of the training set
- Average weak leaners
- Reduce variance

Random forest is basically a bagging approach, another way of combining weak learners. Here the weak learners are decision trees. We can also use decision tree to do boosting.

8 Bootstrap

- Idea: Bootstrapping learns about the sample characteristics by taking resamples and use the information to infer the characteristics of the population.
- Retake samples with replacement from the original samples, i.e. taking samples from empirical distribution
- Provides a powerful tool to calculate the standard error of an estimator, construct confidence intervals, and many other uses.
- Example: Estimate variance of estimator
  - We would like to decide the variance of an estimator $\hat{\theta}$
  - Use data to generate B batches of new samples and evaluate $\hat{\theta}$ for each resampled batch

$$Var(\hat{\theta}) \approx \frac{1}{B-1} \sum_{i=1}^{B} (\hat{\theta}_i - \bar{\theta})^2$$

9 Random forest

- For $b = 1, ..., B$
- Draw bootstrap sample Z_b of size N from training data
- Grow a **random forest tree** T_b for the bootstrapped data by
  - Select v variables at random from p variables
  - Pick the best variable/split among the v
  - Split the node into two children nodes

  Outcome ensemble of trees T_b, $b = 1, ..., B$.
- To make predictions for a new point x
  - Regression: take average $\frac{1}{B}\sum_{b=1}^{B} T_b(x)$
  - Classification: majority vote of $\{T_b(x)\}$, $b = 1, ..., B$.

  Considerations
- **Difference between bagging and Random forest**: in Random forests, only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered for splitting a node.
- Effect of averaging:
  - Average of B i.i.d. random variables, each with variance σ^2, has variance σ^2/B.
  - If the variables are dependent with positive correlation ρ, the variance of the average is $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$. (This is the case for random forest since the decision trees are not i.i.d, they are dependent with each other as the resample batches can be very similar.)
- The amount of correlation between pairs of bagged tree limit the benefit of averaging
- Before each split, select $v \leq p$ of the input variables at random as candidates of splitting.
  - For classification, $v \approx floor(\sqrt{p})$
  - For regression, the default value $v \approx floor(p/3)$

How to decide the number of trees in a random forest model?

10 OOB error

- Out-of-bag (OOB) samples:
  For each observation $z^i = (x^i, y^i)$, construct its random forest predictor be averaging only those trees corresponding to bootstrap samples in which z^i dose not appear.
- OOB error estimate is almost identical to that obtained by $N - fold$ cross validation
- Help to decide the number of trees: once OOB error stabilizes, the training can be terminated

11 Boosting trees

- A tree can be represented as (parameter $\Theta = \{R_j, \gamma_j\}_{j=1}^J$)

$$T(x, \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{I}(x \in R_j),$$

  where each region has a weight γ_j.
- Minimize the cost function

$$\hat{\Theta} = \text{argmin}_\Theta \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y^i, \gamma_j)$$

  where L is the cost function which can be misclassification rate, Gini index, or cross entropy.

12 Forward stagewise optimization for boosting trees

- Solve a sequence of optimization problems

$$\hat{\Theta}_t = \text{argmin}_{\Theta_t} \sum_{i=1}^{m} L(y^i, f_{t-1}(x^i) + T(x^i; \Theta_t))$$

- Alternating minimization: finding $\gamma(j)$ given R_j, and then find R_j
- For instance, if we have exponential loss like adaboost

$$\hat{\gamma}_t(j) = \frac{1}{2} \log \frac{\sum_{x^i \in R_{jt}} D_t(i) \mathbb{I}(y^i = 1)}{\sum_{x^i \in R_{jt}} D_t(i) \mathbb{I}(y^i = -1)}.$$

The following notes are from Chapter 8.2 of the book *An Introduction to Statistical Learning* by Gareth J, Daniela W, Trevor H, Robert T.

Idea of Boosting Regression Trees: The boosting approach learns slowly. Given the current model, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y, as the response. 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. By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it dose not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals. In general, statistical learning approaches that learn slowly tend to perform well. Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have been grown. (Boosting classification trees proceeds in a similar but slightly more complex way.)

13 Three tuning parameters of Boosting

1 The number of trees B. Boosting can overfit if B is too large. We use cross-validation to select B.

2 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.

3 The number d of splits in each tree (number of terminal nodes: $d + 1$). d controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a simple split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. 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.

Difference between boosting and random forest: in boosting, because the growth of a particular tree takes into account the other trees that have already been grown, smaller trees are typically sufficient. Using smaller trees can aid in interpretability as well; for instance, using stumps (d = 1) leads to an additive model.