

STAT 542: Statistical Learning

Tree and Random Forest

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: <https://teazrq.github.io/stat542/>

April 3, 2022

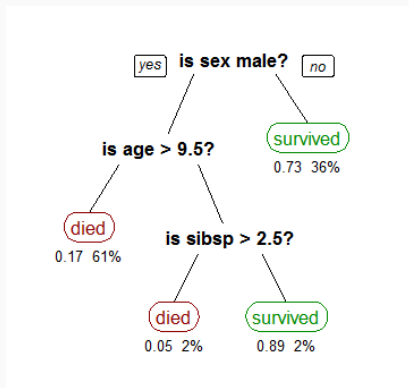
Department of Statistics
University of Illinois at Urbana-Champaign

Classification and regression Trees (CART)

Tree-based methods

- Tree-based methods are nonparametric methods that **recursively partition** the feature space into hyper-rectangular subsets, and make prediction on each subset.
- Two main streams of models:
 - Classification and regression Trees (**CART**): Breiman, Friedman, Olshen and Stone (1984)
 - ID3/C4.5: Quinlan, 1986, 1993.
- Both are among the top algorithms in data mining (Wu et al., 2008)
- In statistical literature, CART is more popular.

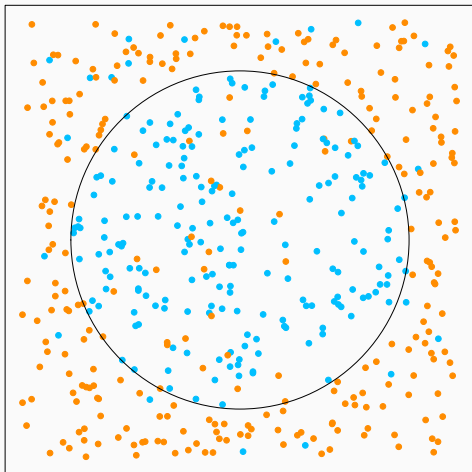
Titanic Survivals



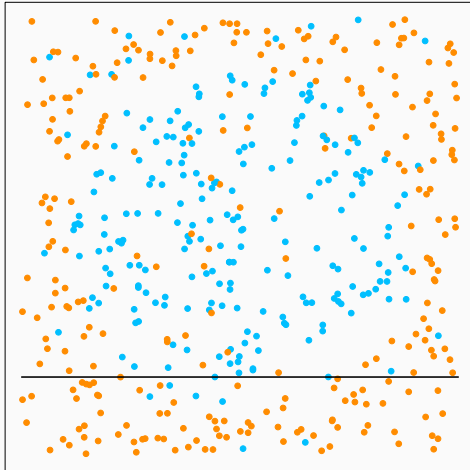
Classification and regression Trees

- How tree works in classification? (We will formally introduce classification problems later)
- Example: independent x_1 and x_2 from uniform $[-1, 1]$, 90% to observed class 1 (blue) within the circle $x_1^2 + x_2^2 < 0.6$, and 90% to observed class 2 (orange) outside the circle.
- All existing classification methods that we have introduced require either transformation of the space (SVM) or distance measure (k NN, kernel)
- Tree solves this by directly cutting the feature space using a binary splitting rule in the form of $\mathbf{1}\{x \leq c\}$

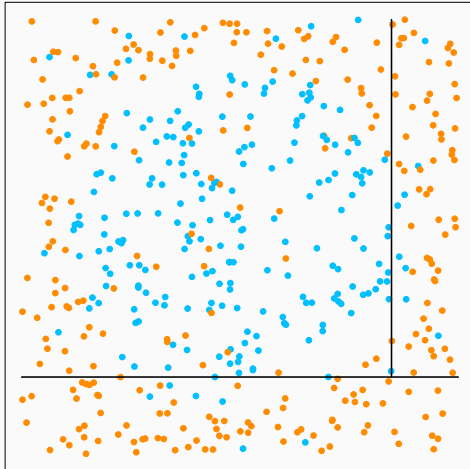
Example



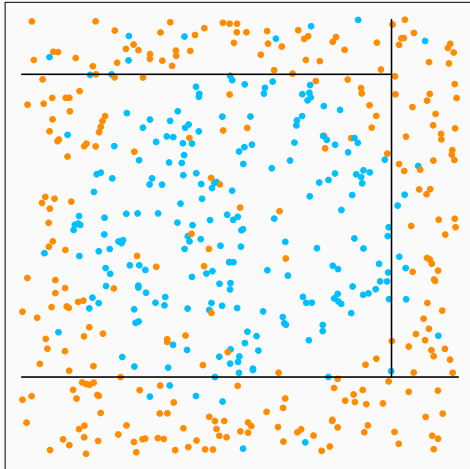
Example



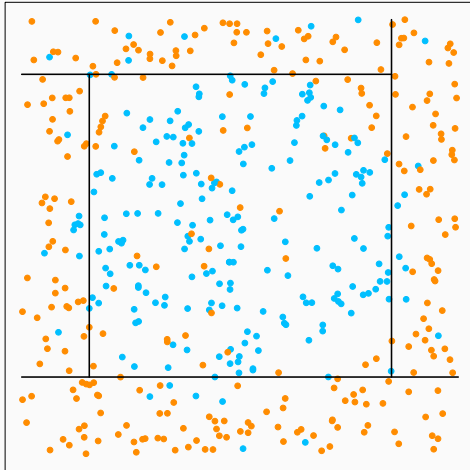
Example



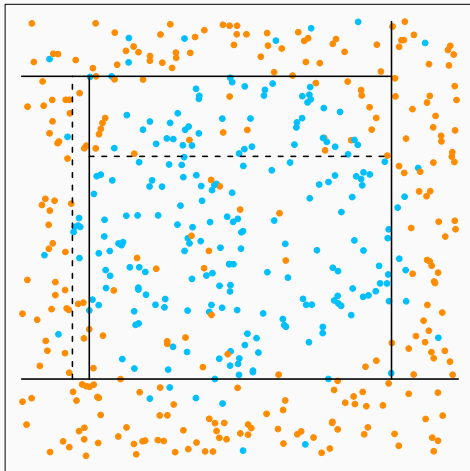
Example



Example



Example

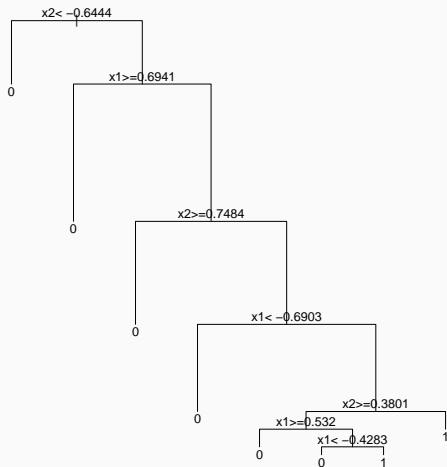


Example

- `rpart` is one of the popular packages that provide CART fitting and plot
- Other choices include `tree`, `party`.
- **Read the reference manual carefully!!!**

```
1 > library(rpart)
2 > x1 = runif(500, -1, 1)
3 > x2 = runif(500, -1, 1)
4 > y = rbinom(500, size = 1, prob = ifelse(x1^2 + x2^2 < 0.6, 0.9, 0.1))
5 > cart.fit = rpart(as.factor(y)~x1+x2, data = data.frame(x1, x2, y))
6 > plot(cart.fit)
7 > text(cart.fit)
```

Example

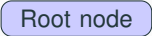


Why tree-based methods?

- Tree-based methods: non-parametric, flexible model structure
- Can handle high-dimensional data without modifying the algorithm (CART may not work well in this setting)
- Single tree model — simple decision rules, interpretable
- Ensemble tree model — high prediction accuracy

Single tree methods: recursive partitioning

- How tree-based methods work?
 - Initialized the root node: all training data



Root node

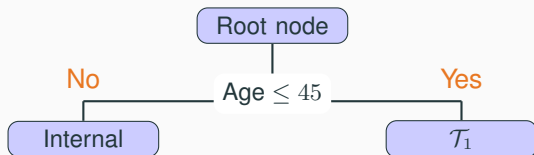
Single tree methods: recursive partitioning

- How tree-based methods work?
 - Initialized the root node: all training data
 - Find a splitting rule $\mathbf{1}\{X^{(j)} \leq c\}$ and split the node



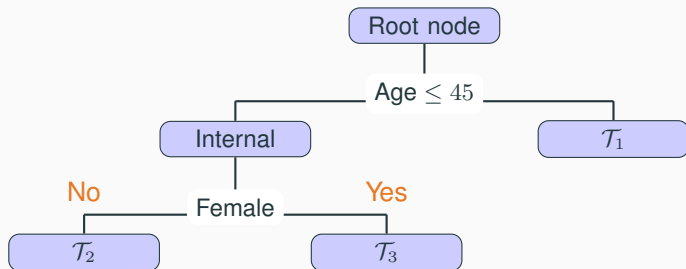
Single tree methods: recursive partitioning

- How tree-based methods work?
 - Initialized the root node: all training data
 - Find a splitting rule $1\{X^{(j)} \leq c\}$ and split the node
 - Recursively apply the procedure on each daughter node



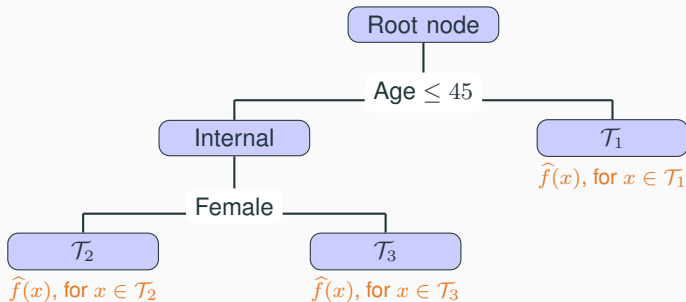
Single tree methods: recursive partitioning

- How tree-based methods work?
 - Initialized the root node: all training data
 - Find a splitting rule $\mathbf{1}\{X^{(j)} \leq c\}$ and split the node
 - Recursively apply the procedure on each daughter node



Single tree methods: recursive partitioning

- How tree-based methods work?
 - Initialized the root node: all training data
 - Find a splitting rule $\mathbf{1}\{X^{(j)} \leq c\}$ and split the node
 - Recursively apply the procedure on each daughter node
 - Predict each terminal node using within-node data



- How to construct the splitting rules?
- For classification problems
 - continuous predictors
 - categorical predictors
- For regression problems
- Tree pruning

Splitting rules for continuous predictors

- Splitting of continuous predictors are in the form of $\mathbf{1}\{X^{(j)} \leq c\}$
- Consider a node \mathcal{T} , we want to split this node into two child nodes.
- We first evaluate the overall **impurity** of \mathcal{T} by the **Gini index** (CART), assuming that there are K different classes of the outcome, $1, \dots, K$,

$$\text{Gini}(\mathcal{T}) = \sum_{k=1}^K \hat{p}_k(1 - \hat{p}_k) = 1 - \sum_{k=1}^K \hat{p}_k^2$$

where $\hat{p}_k = \frac{\sum_i \mathbf{1}\{y_i=k\} \mathbf{1}\{x_i \in \mathcal{T}\}}{\sum_i \mathbf{1}\{x_i \in \mathcal{T}\}}$ is the within node frequency of class k .

Impurity Measures

- Now propose a split: $\mathbf{1}\{X^{(j)} \leq c\}$.
- This separates all subjects in the node into two disjoint parts: \mathcal{T}_L and \mathcal{T}_R
- For each child node, we can again evaluate $\text{Impurity}(\mathcal{T}_L)$ and $\text{Impurity}(\mathcal{T}_R)$
- The reduction of impurity is measured by

$$\text{score} = \text{Gini}(\mathcal{T}) - \left(\frac{N_{\mathcal{T}_L}}{N_{\mathcal{T}}} \text{Gini}(\mathcal{T}_L) + \frac{N_{\mathcal{T}_R}}{N_{\mathcal{T}}} \text{Gini}(\mathcal{T}_R) \right),$$

where $N_{\mathcal{T}_L}$, $N_{\mathcal{T}_R}$ and $N_{\mathcal{T}}$ are the sample size for the corresponding node.

- Go through **all variables j and all cutting points c** to find the split with the best score
- Using **rpart** or **tree**, the magnitude of this score is reflected by the height of each split (plot in page 13)

Other Measures

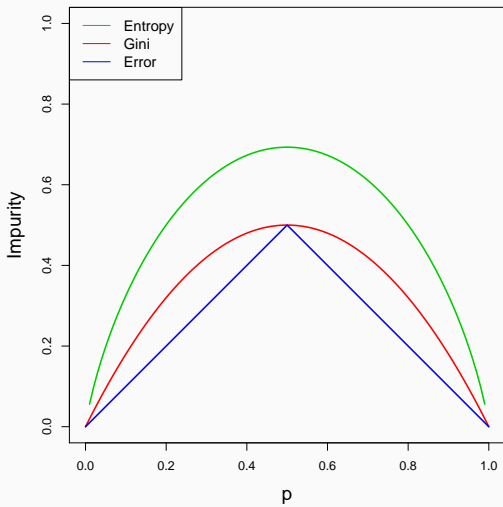
- Gini index is not the only measurement
- Shannon entropy (ID3/C4.5)

$$\text{Entropy}(\mathcal{T}) = - \sum_{k=1}^K \hat{p}_k \log(\hat{p}_k)$$

- Similarly, we can use Shannon entropy to define the reduction of impurity and search for the best splitting rule
- There are other differences between C4.5 and CART, for example, C4.5 can create multiple child nodes from one node.
- Misclassification error

$$\text{Error}(\mathcal{T}) = 1 - \max_{k=1, \dots, K} \hat{p}_k$$

Comparing Measures



Comparing Measures

	Class 1	Class 2	\hat{p}_1	\hat{p}_2	Gini	Entropy	Error
\mathcal{T}	7	3	7/10	3/10	0.420	0.611	0.3
$\mathcal{T}_{\text{left}}$	3	0	3/3	0	0	0	0
$\mathcal{T}_{\text{right}}$	4	3	4/7	3/7	0.490	0.683	3/7

$$\text{score}_{\text{Gini}} = 0.420 - (3/10 \cdot 0 + 7/10 \cdot 0.490) = 0.077$$

$$\text{score}_{\text{Entropy}} = 0.611 - (3/10 \cdot 0 + 7/10 \cdot 0.683) = 0.133$$

$$\text{score}_{\text{Error}} = 3/10 - (3/10 \cdot 0 + 7/10 \cdot 3/7) = 0$$

Comparing Measures

- Gini index and Shannon Entropy are more sensitive to the changes in the node probability
- They prefer to create more “pure” nodes
- Misclassification error can be used for evaluating a tree, but may not be sensitive enough for building the tree.

Categorical Predictors

- For categorical predictor $X^{(j)}$ taking values in $\{1, \dots, M\}$, we search for a subset A of $\{1, \dots, M\}$, and evaluate the child nodes created by the splitting rule

$$\mathbf{1}\{X^{(j)} \in A\}$$

- Maximum of $2^{M-1} - 1$ number of possible splits
- When M is too large, this can be computationally intense.
- Some heuristic methods are used, such as randomly sample a subset of categories to one child node, and compare several random splits.

Tree for Regressions

- When the outcomes y_i 's are continuous, all we need is a corresponding impurity measure and score
- Consider the weighted variance reduction:

$$\text{score}_{\text{Var}} = \text{Var}(\mathcal{T}) - \left(\frac{N_{\mathcal{T}_L}}{N_{\mathcal{T}}} \text{Var}(\mathcal{T}_L) + \frac{N_{\mathcal{T}_R}}{N_{\mathcal{T}}} \text{Var}(\mathcal{T}_R) \right)$$

where for any node \mathcal{T} , $\text{Var}(\mathcal{T})$ is just the variance of the node samples:

$$\text{Var}(\mathcal{T}) = \frac{1}{N_{\mathcal{T}}} \sum_{i \in \mathcal{T}} (y_i - \bar{y}_{\mathcal{T}})^2$$

Overfitting and Tree Pruning

- A large tree (with many splits) can easily overfit the data
- Small tree may not capture important structures
- Tree size is measured by the number of splits
- Balancing tree size and accuracy is the same as the “loss + penalty” framework
- One possible approach is to split tree nodes only if the decrease in the loss exceed certain threshold, however this can be short-sighted
- A better approach is to grow a large tree, then prune it

Cost-Complexity Pruning

- Fit the entire tree T_{max} (possibly one observation per terminal node). Specify a **complexity parameter** α .
- For any sub-tree of T_{max} , denoted as $T \preceq T_{max}$, calculate

$$\begin{aligned}C_{\alpha}(T) &= \sum_{\text{all terminal nodes } t \text{ in } T} N_t \cdot \text{Impurity}(t) + \alpha|T| \\ &= C(T) + \alpha|T|\end{aligned}$$

where N_t is the number of observations in t , $|T|$ is the size of T , i.e., the number of terminal nodes

- Find T that minimize the $C_{\alpha}(T)$
- Large α results in small trees
- Choose α using CV

Cost-Complexity Pruning

- We do not need to exhaust all possible sub-trees of T_{max} (computationally intense)
- **Weakest-link cutting** is used to find the best sub-tree:
 - Look at an internal node t of T_{max} , and denote the entire branch starting from t as T_t
 - Compare: remove the entire branch (collapse T_t into a single terminal node) vs. keep T_t

$$\alpha \leq \frac{C(t) - C(T_t)}{|T_t| - 1}$$

- Cut the branch that has the smallest value on the right hand side (the smallest α), then iterate
- Will provide a solution path of α (from smallest to largest).

Missing Values

- If each variable has 5% chance have missing value, then with 50 variables, there are only 7.7% of the samples that has complete measures.
- Traditional approach is to discard observations with missing values, or impute them
- Tree-based method can handle them by either putting them as a separate category, or using surrogate variables whenever the splitting variable is missing.

- Advantages of tree-based method:
 - handles both categorical and continuous variables in a simple and natural way
 - Invariant under all monotone transformations of variables
 - Robust to outliers
 - Flexible model structure, capture interactions, easy to interpret
- Limitations
 - Small changes in the data can result in a very different series of splits
 - Non-smooth. Some other techniques such as the multivariate adaptive regression splines (MARS, Friedman 1991) can be used to generate smoothed models.

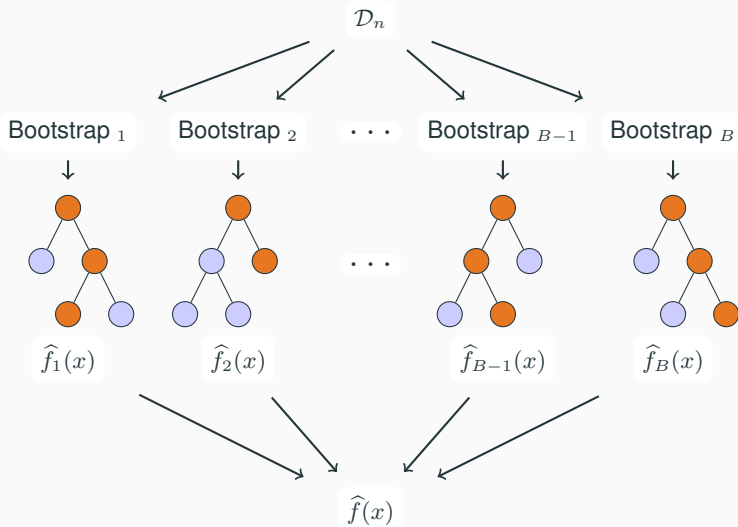
Random Forests

Weak and Strong Learners

- Back in the mid-late 90's, researchers started to investigate whether aggregated “weak learners” (unstable, less accurate) can be a “strong learner”.
- Bagging, boosting, and random forests are all methods along this line.
- Bagging and random forests learn individual trees with some random perturbations, and “average” them.
- Boosting progressively learn models with small magnitude, then “add” them
- In general, Boosting, Random Forests \succ Bagging \succ Single Tree.

- Bagging stands for “**Bootstrap aggregating**”
- Draw M bootstrap samples from the training dataset, fit CART to each, then average
- **Motivation**: CART is unstable as we discussed earlier, however, perturbing and averaging can improve stability and leads to better accuracy

Ensemble of trees



Bagging Predictors

- Bootstrap sample **with replacement**: some observations can be repeated multiple times. $\sim 63.2\%$ unique samples
- Fit a CART model to each bootstrap sample (may require tuning using CV).
- To combine each learner, for classification problems:

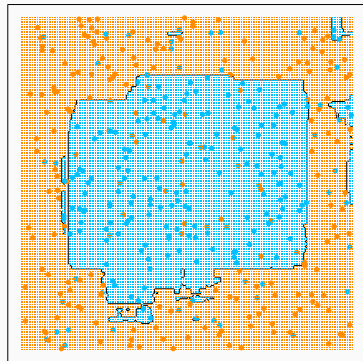
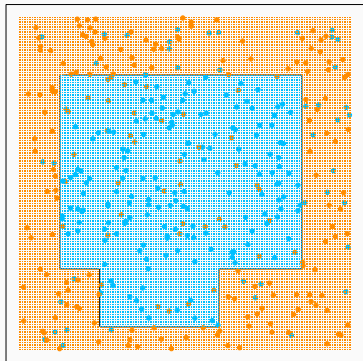
$$\hat{f}_{\text{bag}}(x) = \text{Majority Vote}\{\hat{f}_b(x)\}_{b=1}^B,$$

and for regression problems:

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

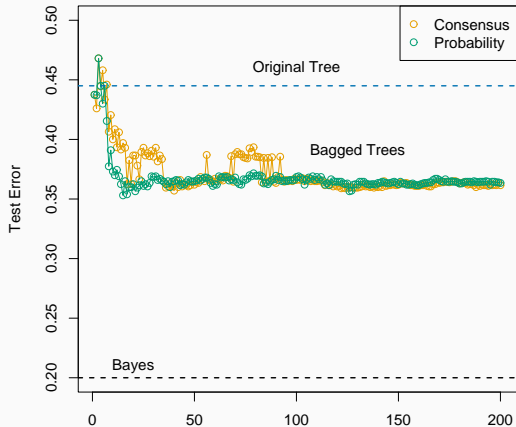
- Dramatically reduce the variance of individual learners
- CART can be replaced by other weak learners

CART vs. Bagging



CART vs. Bagging on our previous example, both are pruned

CART vs. Bagging



Remarks about Bagging

- Bagging can dramatically reduce the variance of unstable “weak learners” like trees, leading to improved prediction
- The simple structure of trees will be lost due to bagging, hence it is not easy to interpret
- However, the performance of bagging is oftentimes not satisfactory. Why?
- Different trees have high correlation which makes averaging not very effective
- How to de-correlate trees?

- Several articles came out in the late 90's discussing the advantages of using random features.
- “The Random Subspace Method for Constructing Decision Forests” by Ho (1998) greatly influenced Breiman's idea of random forests
- In Ho's method, each tree is constructed using a randomly selected subset of features
- Random forests take a step forward: each splitting rule only consider a random subset

Tuning parameters: *mtry*

- An important tuning parameter of random forests is *mtry*
- At each split, randomly select *mtry* variables from the entire set of features $\{1, \dots, p\}$
- Search the best variable and splitting point out of these *mtry* variables
- Split and proceed to child nodes
- This procedure turns out working remarkably well, even in high-dimensional data

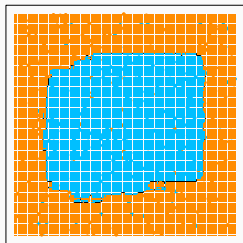
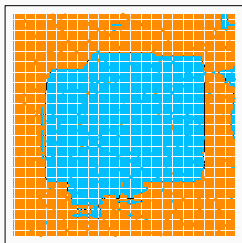
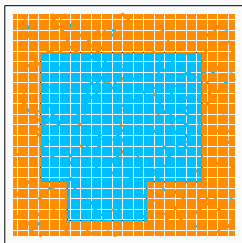
Tuning parameters: n_{min}

- Another important tuning parameter is n_{min} (terminal node size)
- Random forests **does not perform pruning** anymore
- Instead, splitting does not stop until the terminal node size is less or equal to n_{min} , and the entire tree is used.
- n_{min} controls the trade-off between bias and variance in each tree
- In the most extreme case, $n_{min} = 1$ means exactly fit each observation, but this is not 1-NN!
- Lin and Jeon (2004) formulate random forests as “potential nearest neighbor” (a much larger neighborhood than k NN)

Tuning parameters

- A summary of important tuning parameters in Random forests (using R package `randomForest`)
 - `ntree` : number of trees, set it to be large. Default 500.
 - `mtry` : number of variables considered at each split. Default $p/3$ for regression, \sqrt{p} for classification.
 - `nodesize` : terminal node size, same as n_{min} . Default 5 for regression, 1 for classification
- Overall, tuning is quite crucial in random forests

CART vs. Bagging vs. RF

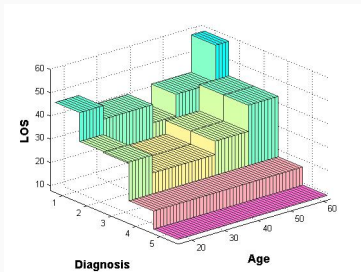


CART

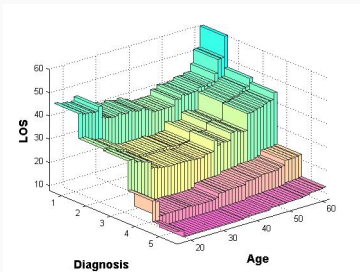
Bagging

RF

CART vs. RF in regression



CART



RF

Model patients' length of stay (LOS) in hospital: Diagnosis (categorical) and Age (continuous)

Variable Importance

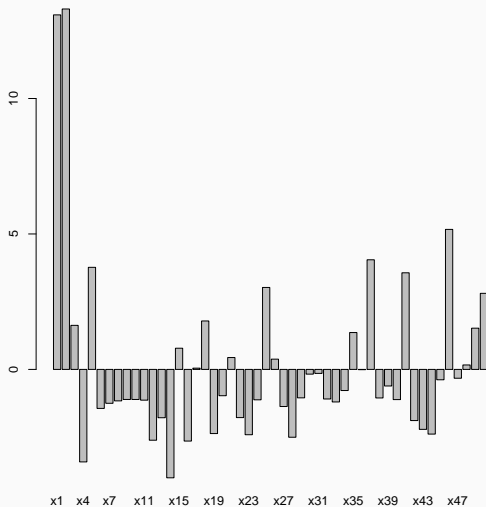
- There is a build-in feature called variable importance that utilize the samples not selected by bootstrapping (out-of-bag data):
 - For each tree m , use the out-of-bag data as the testing set to obtain the prediction error: Err_0^m
 - For each variable j , randomly permute its value among the testing samples, and recalculate the prediction error: Err_j^m
 - calculate for each j

$$\text{VI}_{mj} = \frac{\text{Err}_j^m}{\text{Err}_0^m} - 1$$

- Average VI_{mj} across all trees (index m) to get the importance of each variable $\text{VI}_{.j}$

- In many practice areas, random forests variable importance measure is used extensively
- Usually the misclassification error is used instead of Gini index
- Higher VI means larger loss of accuracy due to the loss of information on $X^{(j)}$, hence more important.

Variable Importance in RF



Same simulation setting as the “circle” example, with additional 48 noise variables.

Additional Developments

- Random splitting point: Instead of searching through all possible cuts, we can just generate several random cutting points and choose the best among them (extremely randomized trees, Geurts et al., 2006). This is computationally much faster, especially for large dataset.
- Linear combination split: at each split, a linear combination $\beta^T X$ is created and used as the splitting rule $\mathbf{1}\{\beta^T X \leq c\}$. The β estimations can be searched by a ridge regression (oblique random forests, Menze et al., 2011)

- Denote $\mathcal{A} = \{A_k\}_{k \in \mathcal{K}}$ the collection of all terminal nodes in a single tree, where \mathcal{K} is a set of indices.
- Then, for any target point x_0 , if x_0 falls into the a terminal node A , we would predict its outcome by node averaging:

$$\hat{f}(x_0) = \frac{\sum_{i=1}^n \mathbf{1}\{x_i \in A\} y_i}{\sum_{i=1}^n \mathbf{1}\{x_i \in A\}}$$

- Note that there is a slight difference from the traditional RF, which is to average the node first than average over trees.

- Instead, we can define a **random forest kernel**:

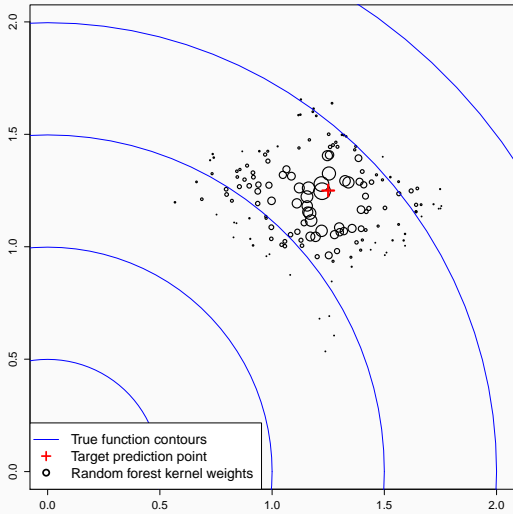
$$\begin{aligned}\mathcal{A}(x_i, x_0) &= \sum_{k \in \mathcal{K}} \mathbf{1}\{x_i \in A_k\} \mathbf{1}\{x_0 \in A_k\} \\ &= \begin{cases} 1 & \text{if } \exists A \in \mathcal{A} \text{ s.t. } x_i, x_0 \in A \\ 0 & \text{o.w.} \end{cases}\end{aligned}$$

- Then $\hat{f}(x_0)$ can be expressed in the N-W kernel estimator form:

$$\hat{f}(x_0) = \frac{\sum_{i=1}^n \mathcal{A}(x_i, x_0) y_i}{\sum_{i=1}^n \mathcal{A}(x_i, x_0)}$$

- We can extend this to the forest version by stacking the kernels
- However, $\mathcal{A}(\cdot, \cdot)$ is not invariant to location shift.

Kernel View of RF



Adaptiveness of RF kernel

- Random Forests is a **kernel method with adaptive bandwidth**:
 - Each tree in a forests defines a kernel: uniform within each terminal node
 - Ensemble: sum of kernels is still a kernel
 - A tree is “more likely” to split on important variables, making their “bandwidth” smaller
 - If the splitting variables are selected wisely, the bandwidth is adaptive to the signal strength
 - The cutting point is more likely to happen on the curvature of the underlying target function
- In a high-dimensional sparse model, random forests performs better (both theoretically and practically) than those non-adaptive local methods

Theoretical Cutting Points

Theoretical cutting points on a function (Ishwaran 2015) prefer places with steepest derivative.

