Tree, Bagging, Random Forest, Boosting

CS115: Math for Computer Science

Wednesday, 25-12-2024

Roadmap

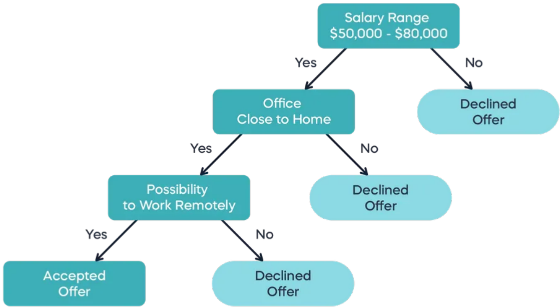
1. Tree
2. Bagging
3. Random Forest
4. Boosting

Roadmap

1. Tree
2. Bagging
3. Random Forest
4. Boosting

What is Decision Tree?

- Decision Tree is a flowchart-like structure used for making decisions and predictions.
- It is a popular method for tasks involving classification and regression.



Main idea

- We will grow a binary tree in order to predict a response or class Y from many inputs $X_1, X_2, \ldots, X_n$.
- At each internal node in the tree, we apply a test to one of the inputs, say X_i.
- Depending on the outcome of the test, we go to either the left or the right sub-branch of the tree
- Eventually we come to a leaf node, where we make a prediction.

Regression tree

1. We divide the predictor space—that is, the set of possible values for $X_1, \ldots, X_n$—into J distinct and non-overlapping regions $R_1, \ldots, R_J$.
2. For every observation that falls into the region R_j , we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

How do we construct the regions $R_1, \ldots, R_J$?

- The regions could have any shape.
- We prioritize diving regions into *high-dimensional rectangles*, or *boxes*

Our main goal is to minimize the RSS, given by

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

where $\hat{y}_{R_j}$ is the mean response for the training observations

- It's not practical to check all possible ways to divide the data into regions.
- To avoid this issue, we use a *top-down*, *greedy* approach called *recursive binary splitting*.
  - Start at the top of the tree and split the data into two groups at each step.
  - At each step, choose the best possible split without looking ahead to future steps.

1. **Start with All Data:** Treat the entire dataset as one region.
2. **Find the Best Split:**
   - Test possible splits for each variable X_j.
   - Divide data into two groups:

     $$R_1(j,s) = \{X \mid X_j < s\}, \quad R_2(j,s) = \{X \mid X_j \geq s\}.$$

   - Calculate Residual Sum of Squares (RSS):

     $$\text{RSS}(j,s) = \sum_{i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \hat{y}_{R_2})^2.$$

   - Pick the split j, s that minimizes RSS.
3. **Split the Data:** Separate data into $R_1(j,s)$ and $R_2(j,s)$.

1. **Repeat Splitting:**
   - Apply the same process to each region.
   - Keep splitting to further reduce RSS.
2. **Set Stopping Rules:** Stop when:
   - A region is too small.
   - Splits no longer improve RSS significantly.
   - Maximum tree depth is reached.
3. **Make Predictions:** Assign observations to regions and predict their outcomes using the average response from the training data in each region.

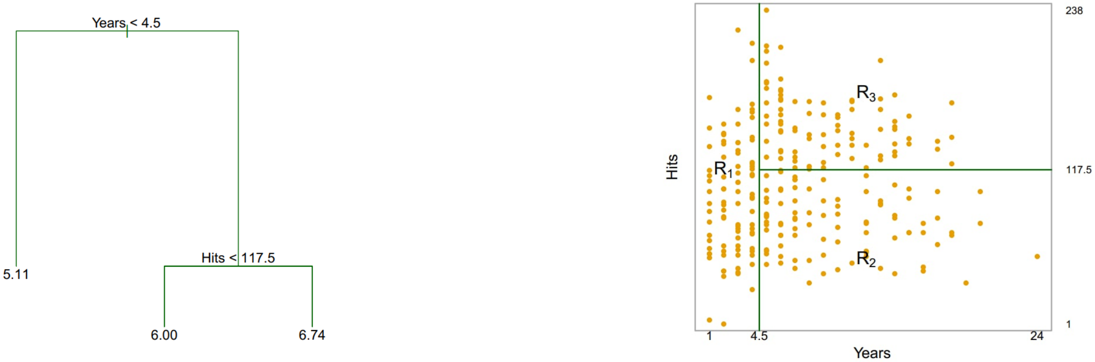


Figure 1: A regression tree predicting a baseball player's log salary, based on years in major leagues and hits in the previous year. Splits are labeled $X_j < t_k$ (left branch) and $X_j \geq t_k$ (right branch).

- It is used to **predict a qualitative response**
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

- We also use recursive binary splitting for classification tree

An alternative to RSS is the classification error rate

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

where $\hat{p}_{mk}$ represents the proportion of training observations in the region m that are from class k

- However, classification error rate is not sufficently sensitive as two other measures (Gini-index and Cross-entropy)

Gini Index

The Gini index is defined by:

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

- The Gini index is called a measure of node purity.
- A small value of $\hat{p}_{mk}$ indicates that a node contains predominantly observations from a single class.

Cross Entropy

An alternative to the Gini index is the cross entropy

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

Since

$$0 \leq \hat{p}_{mk} \leq 1$$

this implies that

$$0 \leq -\hat{p}_{mk} \log(\hat{p}_{mk}).$$

Note: The cross entropy will take on a value near 0 if the $\hat{p}_{mk}$'s are all near 0 or 1.

Why Prune?

- Fully grown trees are likely to overfit training data.
- Pruning simplifies trees to improve generalizability.

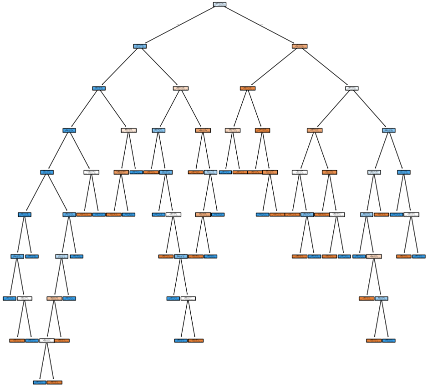


Figure 2: Unprunned Tree

Pre-Pruning

- Stops tree growth early using constraints (e.g., depth, node size).
- Prevents overfitting but may underfit if stopped too soon.

Post-Pruning

- Fully grows the tree, then removes unnecessary branches.
- Balances complexity and performance but is computationally expensive.

Complexity pruning function

The cost complexity pruning function is given by:

$$C(T) = R(T) + \alpha|T|$$

where:

$C(T)$ is the total cost of the tree,
$R(T)$ is the empirical risk (e.g., Gini, RSS,...),
α is the pruning parameter
$|T|$ is the number of terminal nodes (leaves) in the tree.

Tree Prunning

Example

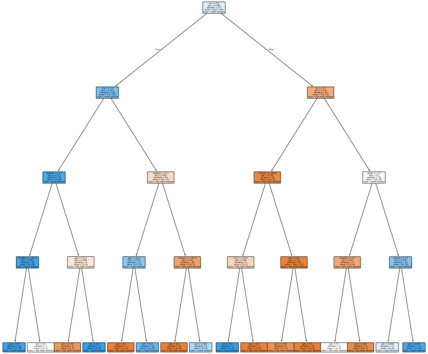


Figure 3: Pre-prunning

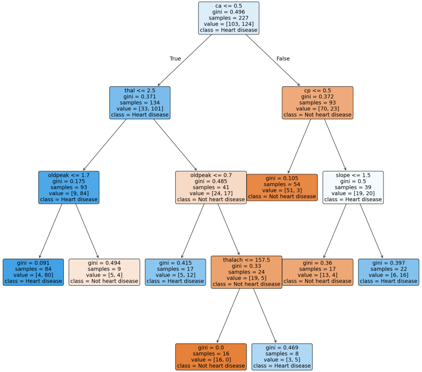
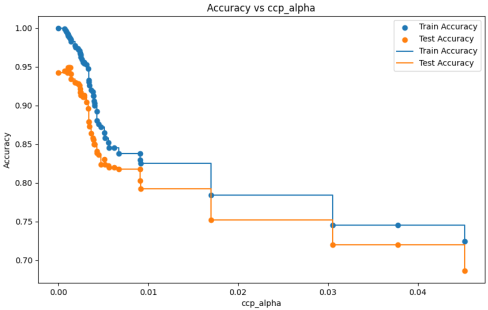
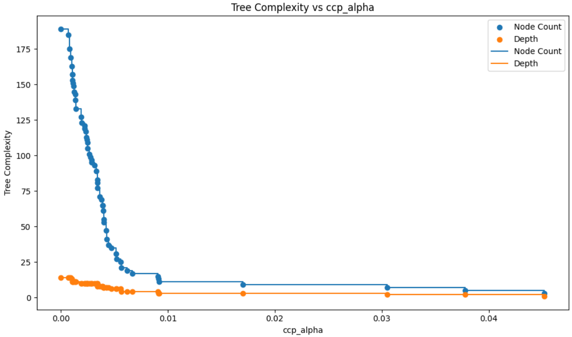


Figure 4: Post-Prunning

Tree Prunning

α effect





Roadmap

1 Tree

2 Bagging

3 Random Forest

4 Boosting

What is Bootstrap?

Boostrapping

- A process involves sampling from the original dataset with replacement to create multiple new datasets.

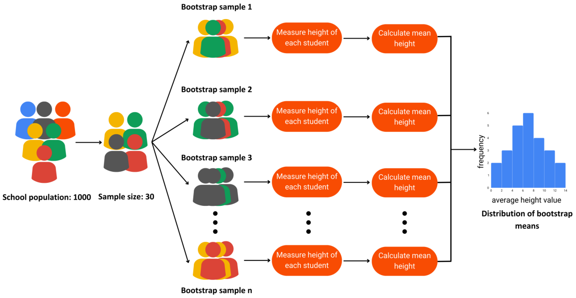
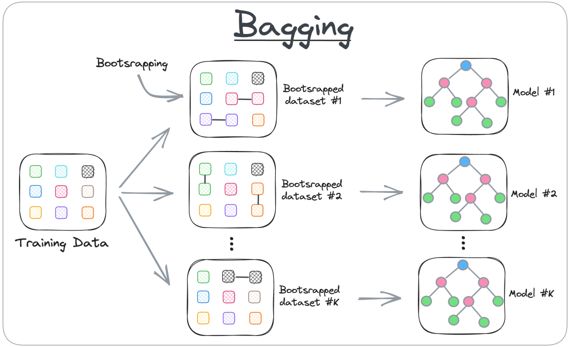


Figure 5: Illustration of the bootstrap resampling process.

- **Bagging or Bootstrap Aggregating** is a technique used to improve the performance of machine learning algorithms by reducing their variance.
- It works by training multiple models on different subsets of the data and then combining their predictions.



Steps of Bagging

- Bagging creates K different training sets by randomly sampling the original dataset with replacement.
- We then train the model on each bootstrapped dataset.

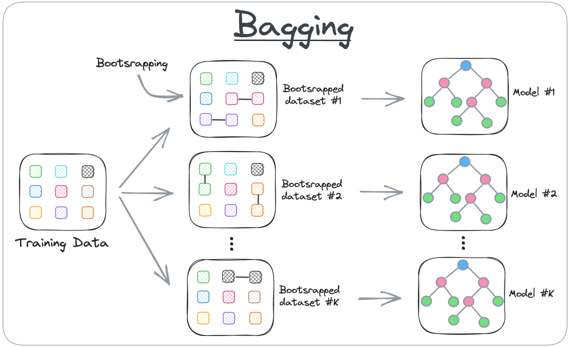


Figure 7: Illustration of Bagging Process

For regression task

We average the predictions of all the models.

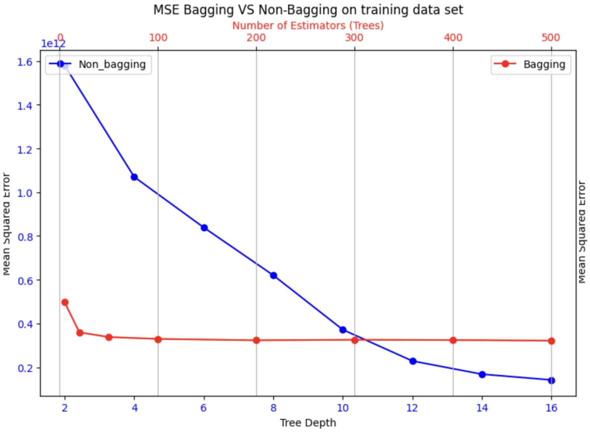
For classification task

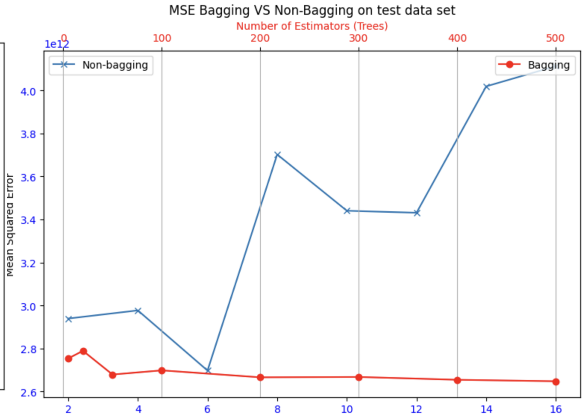
We take a majority vote among the predictions made by all the models.

- **This means the class predicted by most trees becomes the final prediction.**

$$\hat{y}_{\text{bagging}} = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$$

- In practice, this process helps reduce overfitting (variance) without significantly increasing bias.





Roadmap

1 Tree

2 Bagging

3 Random Forest

4 Boosting

- "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely **inaccuracy**." - *The Elements of Statistical Learning*
- **Random Forests** is a combination of the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.

Random Forest

- **Random forest** is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them.
- **Random forests** are popular and are implemented in a variety of packages because they are *easy to train and tune*.

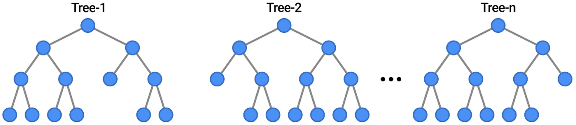
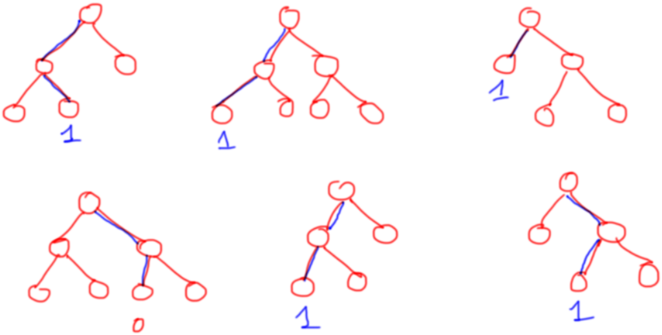


Figure 8: Random forest

Random Forest
Example



$$\Rightarrow \begin{cases} 1 : 5 \\ 0 : 1 \end{cases} \Rightarrow 1$$

- Trees are ideal candidates for **bagging**, since they can capture **complex interaction structures** in the data and if grown sufficiently deep, have relatively **low bias**.
- Since trees are notoriously **noisy**, they benefit greatly from **averaging**.

- For B **independent and identically distributed (i.i.d.)** random variables, the variance of the average is: $\frac{1}{B}\sigma^2$

If the variables are **identically distributed** with **pairwise positive correlation** ρ, the variance of the average is:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- As B increases, the second term **diminishes**, but the first term **remains**. Therefore, the **correlation** between pairs of bagged trees ρ **limits** the benefits of averaging.

- Random forests enhance the variance reduction by **reducing the correlation** between trees. This is achieved through the **random selection of input variables** during the tree-building process.
- Before each split, **randomly select** $m \leq p$ input variables as candidates for splitting. Common values for m include $\sqrt{p}$ or **even as low as 1.**
- Inventors' Recommendation:
  - For **classification:**
    - Default value for m: $[\sqrt{p}]$
    - Minimum node size: 1
  - For **regression:**
    - Default value for m: $[p/3]$
    - Minimum node size: 5

- The accuracy of the **Random Forest (RF)** can be assessed using the OOB data. Specifically:
  - The proportion of OOB samples that are **correctly classified** by the RF provides an estimate of the model's accuracy.
  - In contrast, the proportion of OOB samples that are **incorrectly classified** is the **error of the outside gap**.

Random Forest
Variable Importance

- A **variable importance plot** from a Random Forest model is a graphical representation that **ranks the importance** of each variable (or characteristic) in the prediction process.
- There are two primary measures of variable importance:
  - **Mean Decrease in Accuracy**: Indicates how much the accuracy decreases when the variable is excluded from the model.
  - **Mean Decrease in Gini**: Represents the decrease in the impurity of the node (Gini impurity) contributed by the variable.

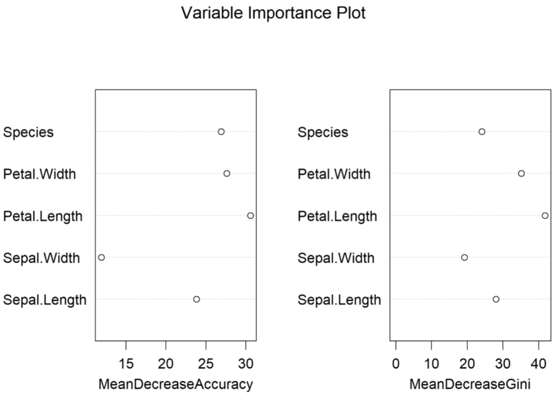


Figure 9: Variable Importance

Roadmap

1 Tree

2 Bagging

3 Random Forest

4 Boosting

- Boosting is an ensemble modeling technique that attempts to build a strong classifier from a number of weak classifiers.
- It is used to improve model accuracy by **reducing its bias**.

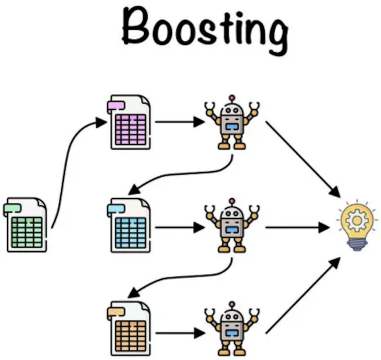


Figure 10: Boosting Technic

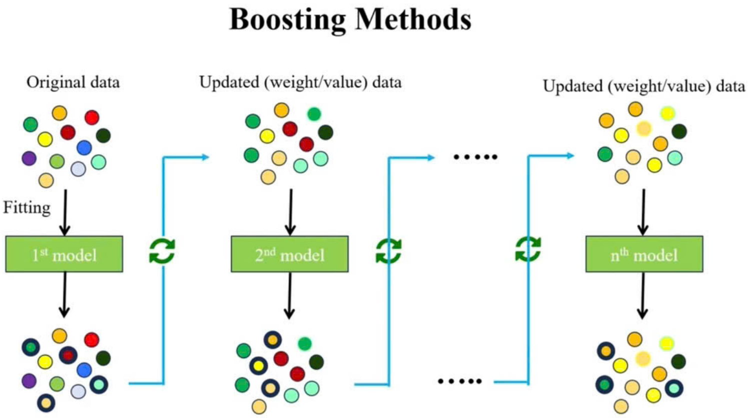


Figure 11: Boosting Process

Dataset

Suppose there is a dataset:$\{(x_1, y_1), \ldots, (x_N, y_N)\}$

with $y_i \in \{-1, 1\}$ and $i \in \{1, \ldots, N\}$

Final Fomula

Write the general final formula of boosting with t weak learners:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Step 1: Initial Weight Assignment

Initially, assign equal weight to each data point:

$$w_i^{(1)} = \frac{1}{N}, \forall i \in \{1, \ldots, N\}.$$

with $\mathrm{w}_i^{(1)}$ is the Weith of i-th data point of the 1st weak-learner

AdaBoost

Step by step

Step 2: Train the Weak Learner

The total error of the t-th weak learner is given by:

$$\epsilon_t = \sum_{i=1}^{N} w_i^{(t)} \cdot I(h_t(x_i) \neq y_i)$$

where:

- $h_t(x_i)$: The t-th weak learner predicts the class label for the i-th data point.
- $w_i^{(}t)$: Weight of the i-th data point of t-th weak learner.
- $I(h_t(x_i) \neq y_i)$: Indicator function, 1 if misclassified, 0 otherwise.

Step 3: Update Weights

Update the weights of the data points as follows:

$$w_i^{t+1} = \frac{w_i^t \cdot e^{\pm \alpha_t}}{\sum_{j=1}^{N} w_j^{t+1}}$$

where:

- e: Euler's number
- α_t: The amount of say of the t-th weak learner, calculated as:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Step 3: Update Weights

Update the weights of the data points as follows:

$$w_i^{t+1} = \frac{w_i^t \cdot e^{\pm\alpha_t}}{\sum_{j=1}^{N} w_j^{t+1}}$$

where:

- $+\alpha$: Misclassified
- $-\alpha$: Classified

Step 4: Combine Weak Learners

Combine all weak learners to form a strong learner. The final prediction is determined by:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where:

- $H(x)$: Final strong learner.
- α_t: The amount of say of the t-th weak learner.
- $h_t(x)$: Prediction of the t-th weak learner.
- T: Total number of weak-learners.

- In practice, this process helps reduce overfitting (Bias).