Lecture 11: Decision Tree and Random Forest

Statistical Methods for Data Science

Yinan Yu

Department of Computer Science and Engineering

December 13, 2021

Today

1. Decision trees
   - Classification tree
   - Regression tree
   - Pruning
   - ID3, C4.5, C5.0, CART
2. Ensemble methods
   - Bias-variance trade-off
   - Bagging
   - Random forest
3. Summary

Learning outcome

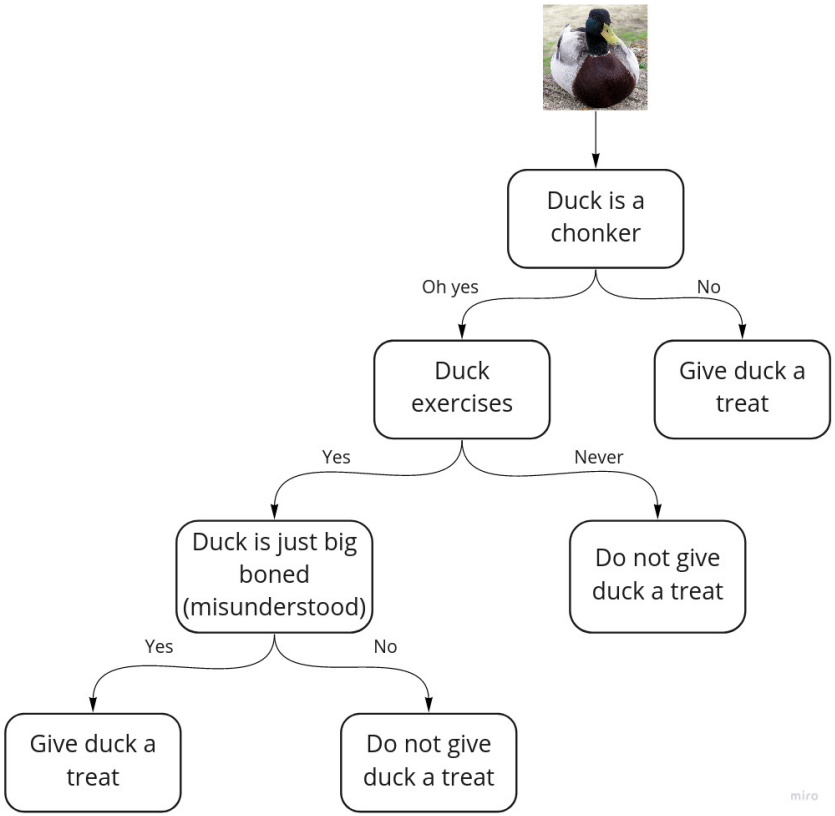
- Understand classification and regression decision trees
- Be able to compute the Gini score
- Be able to explain how to compute the split in the CART algorithm
- Be able to explain the random forest algorithm

Today

1. Decision trees
   - Classification tree
   - Regression tree
   - Pruning
   - ID3, C4.5, C5.0, CART
2. Ensemble methods
   - Bias-variance trade-off
   - Bagging
   - Random forest
3. Summary

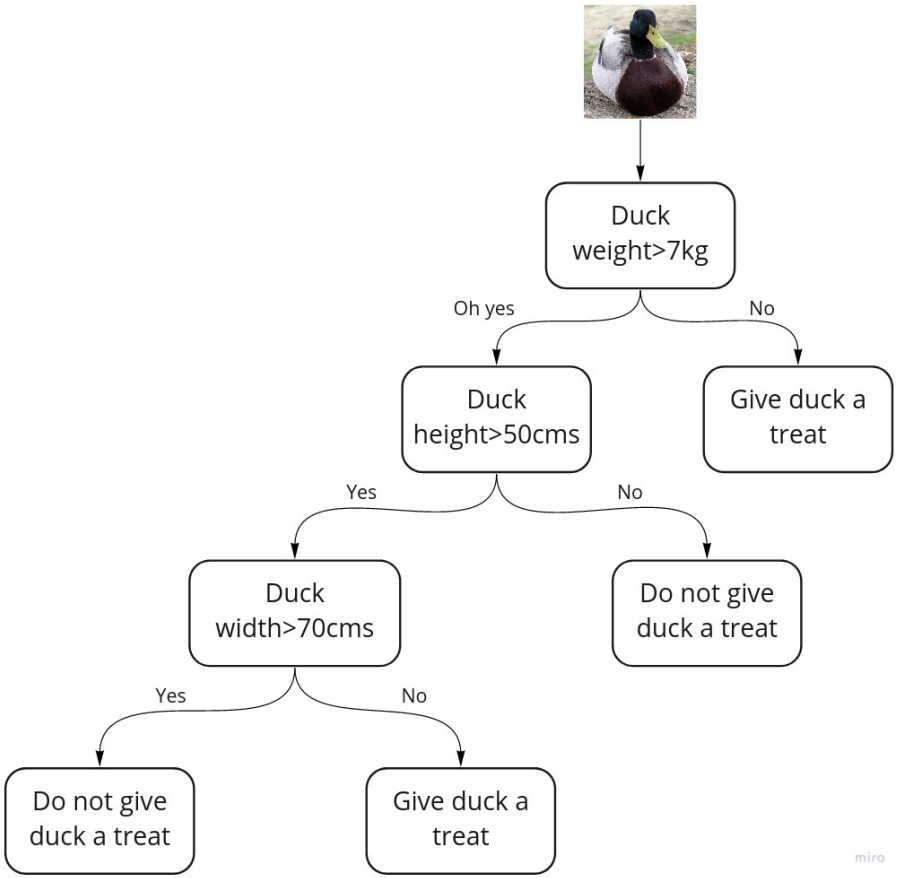
Classification tree with categorical variables

- **Variable (categorical):** $\boldsymbol{x} = [x_1, x_2, x_3]$,
  $x_1 \in \{\text{chonker, slim}\}$, $x_2 \in \{\text{exercises, never exercises}\}$, $x_3 \in \{\text{big-boned, small-boned}\}$
- **Target (categorical):** $y \in \{\text{give duck a treat, do not give duck a treat}\}$



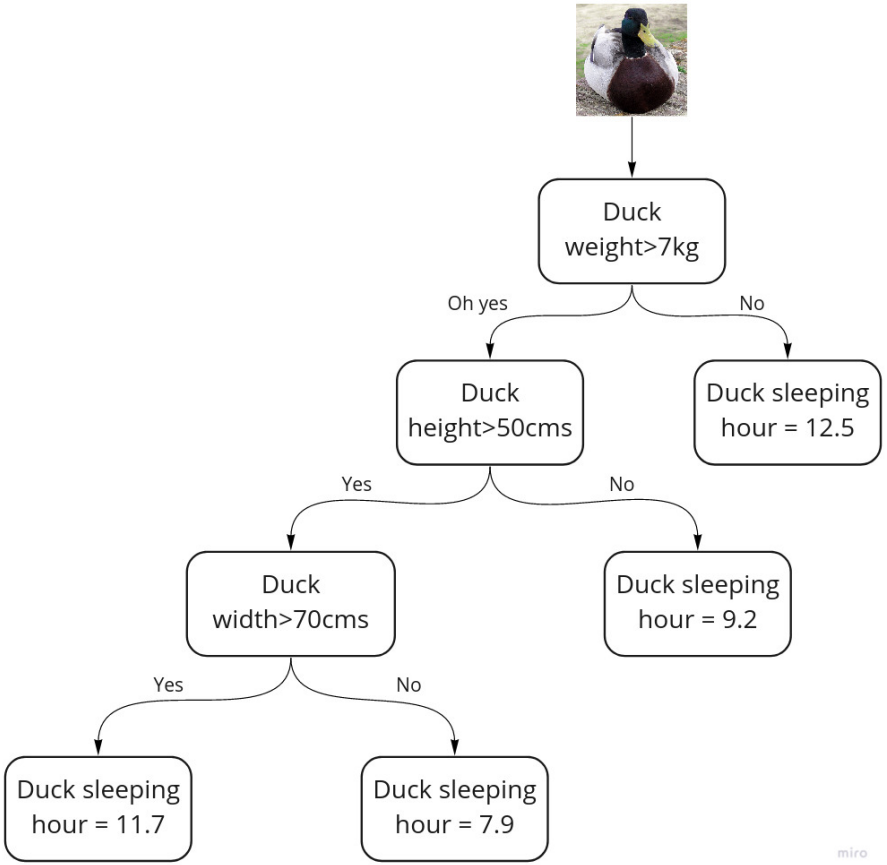
Classification tree with numerical variables

- **Variable (numerical):** $\boldsymbol{x} = [\text{weight}, \text{height}, \text{width}]$
- **Target (categorical):** $y \in \{\text{give duck a treat, do not give duck a treat}\}$



Regression tree with numerical variables

- **Variable (numerical):** $\boldsymbol{x} = [\text{weight}, \text{height}, \text{width}]$
- **Target (numerical):** $y =$ sleep hours per day



Classification tree

Classification decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** or **categorical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$

Classification decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** or **categorical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **categorical** scalar

$$y = \sum_{m=1}^{M} c_m \boldsymbol{I}_{R_m}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_m}$ is the indicater function:

$$\boldsymbol{I}_{R_m}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_m \\ 0, & \text{otherwise} \end{cases}$$

and $c_m \in \{0, \cdots, C-1\}$ is the class index and R_m are the decision regions (leaf nodes)

Classification decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** or **categorical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **categorical** scalar

$$y = \sum_{m=1}^{M} c_m \boldsymbol{I}_{R_m}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_m}$ is the indicater function:

$$\boldsymbol{I}_{R_m}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_m \\ 0, & \text{otherwise} \end{cases}$$

and $c_m \in \{0, \cdots, C-1\}$ is the class index and R_m are the decision regions (leaf nodes)
- **Parameter**:
  - All non-leaf nodes
  - Decision regions R_m (leaf nodes) for $m = 1, \cdots, M$
  - A class index c_m for each region R_m

Classification decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** or **categorical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **categorical** scalar

$$y = \sum_{m=1}^{M} c_m \boldsymbol{I}_{R_m}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_m}$ is the indicater function:

$$\boldsymbol{I}_{R_m}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_m \\ 0, & \text{otherwise} \end{cases}$$

and $c_m \in \{0, \cdots, C-1\}$ is the class index and R_m are the decision regions (leaf nodes)
- **Parameter**:
  - All non-leaf nodes
  - Decision regions R_m (leaf nodes) for $m = 1, \cdots, M$
  - A class index c_m for each region R_m
- **Parameter estimation**: **classification and regression tree (CART)**, Iterative Dichotomiser 3 (ID3), C4.5, C5.0

Classification decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** or **categorical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **categorical** scalar

$$y = \sum_{m=1}^{M} c_m \boldsymbol{I}_{R_m}(\boldsymbol{x})$$

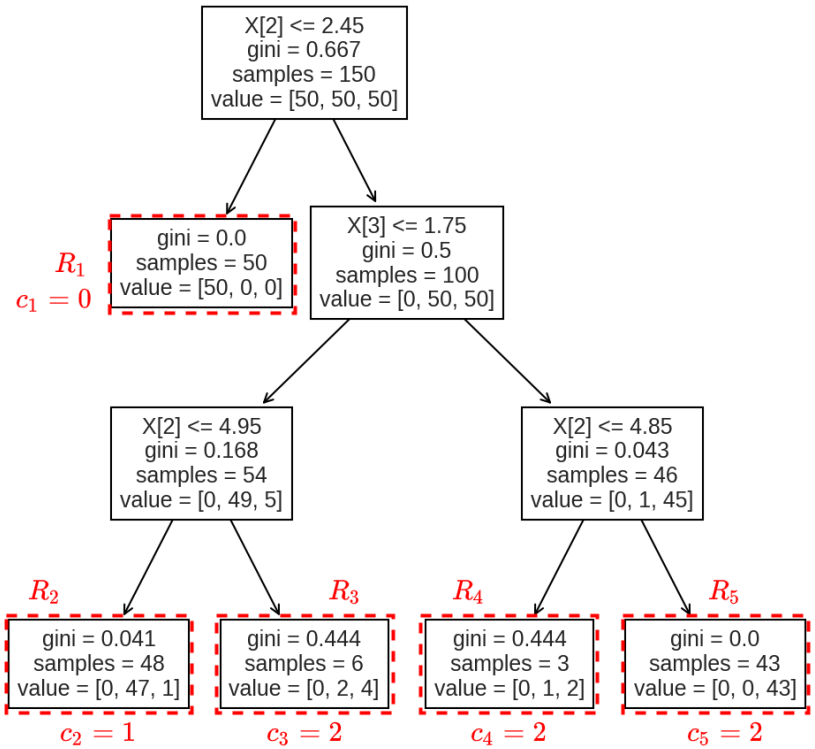
where $\boldsymbol{I}_{R_m}$ is the indicater function:

$$\boldsymbol{I}_{R_m}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_m \\ 0, & \text{otherwise} \end{cases}$$

and $c_m \in \{0, \cdots, C-1\}$ is the class index and R_m are the decision regions (leaf nodes)
- **Parameter**:
  - All non-leaf nodes
  - Decision regions R_m (leaf nodes) for $m = 1, \cdots, M$
  - A class index c_m for each region R_m
- **Parameter estimation**: **classification and regression tree (CART)**, Iterative Dichotomiser 3 (ID3), C4.5, C5.0
- **Hyperparameter**: stopping criteria (e.g. maximum depth, minimum number of data points assigned to each region), hyperparameters for pruning, etc

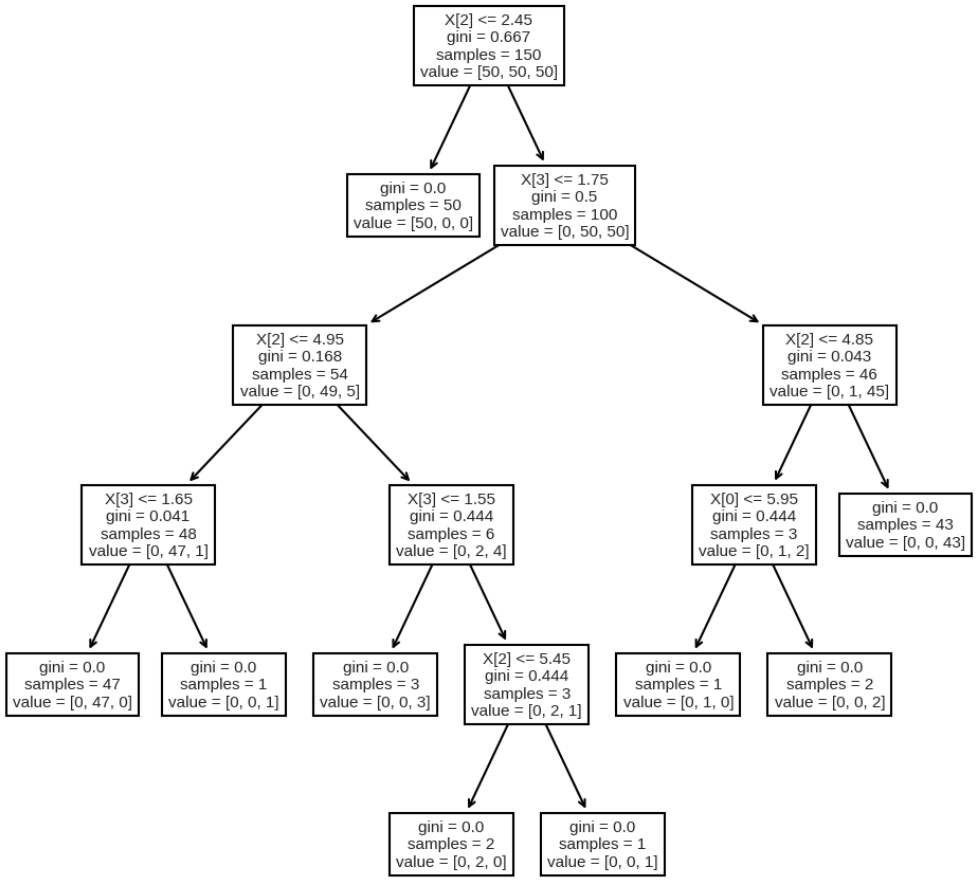
Visualization - maximum depth 3

Example visualization using scikit-learn on the data set iris

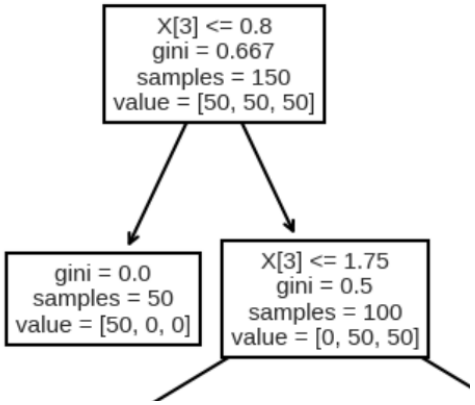


$R_1, \cdots, R_5$ are the decision regions (leaf nodes)

Visualization - maximum depth 5

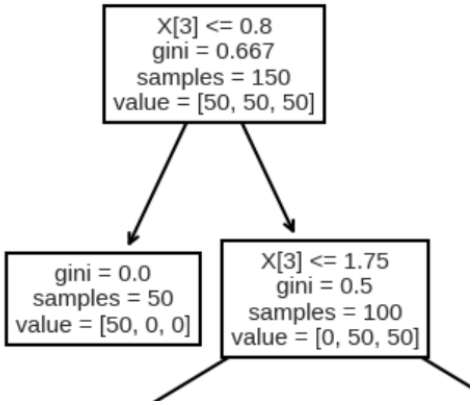


Parameter estimation (CART)



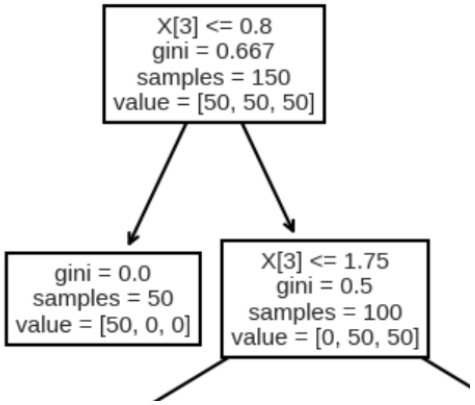
- To use a decision tree for classification, we need to construct the tree, i.e. parameter estimation or training

Parameter estimation (CART)



- To use a decision tree for classification, we need to construct the tree, i.e. parameter estimation or training
- Once a tree is **trained**, an unseen data point $\boldsymbol{x}$ will be assigned to a region R_m (leaf node) by the decision tree to identify the class $\hat{y}$

Parameter estimation (CART)



- To use a decision tree for classification, we need to construct the tree, i.e. parameter estimation or training
- Once a tree is **trained**, an unseen data point $\boldsymbol{x}$ will be assigned to a region R_m (leaf node) by the decision tree to identify the class $\hat{y}$
- To **train a tree**, we need to estimate the parameters, i.e. all the nodes and a class c_m associated with each leaf node R_m for $m = 1, \cdots, M$

Parameter estimation (CART) (cont.)

1. What does a node do during training?

Parameter estimation (CART) (cont.)

1. What does a node do during training?

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated
   - Given the **feature index** i and the **splitting criterion**, some training data will be assigned to the left child node and the rest will be assigned to the right child node

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated
   - Given the **feature index** i and the **splitting criterion**, some training data will be assigned to the left child node and the rest will be assigned to the right child node
2. What does a node do during prediction?

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated
   - Given the **feature index** i and the **splitting criterion**, some training data will be assigned to the left child node and the rest will be assigned to the right child node
2. What does a node do during prediction?
   - Categorical x_i: Is x_i from **category** c?
     - Yes: go to the left child node
     - No: go to the right child node
   - Numerical x_i: Is $x_i \leq$ **t**?
     - Yes: go to the left child node
     - No: go to the right child node

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated
   - Given the **feature index** i and the **splitting criterion**, some training data will be assigned to the left child node and the rest will be assigned to the right child node
2. What does a node do during prediction?
   - Categorical x_i: Is x_i from **category** c?
     - Yes: go to the left child node
     - No: go to the right child node
   - Numerical x_i: Is $x_i \leq$ **t**?
     - Yes: go to the left child node
     - No: go to the right child node
3. Is it possible to have multi-way splits instead of binary (left/right)?

Parameter estimation (CART) (cont.)

1. What does a node do during training?
   - A node contains a subset of the training data set
   - The following parameters need to be estimated from the (sub) training set
     - Which **feature index** i from $x_1, \cdots, x_d$ should be used for the split?
     - What **splitting criterion** to use?
       - If x_i is categorical, a **category** c needs to be estimated
       - If x_i is numerical, a **threshold** t needs to be estimated
   - Given the **feature index** i and the **splitting criterion**, some training data will be assigned to the left child node and the rest will be assigned to the right child node
2. What does a node do during prediction?
   - Categorical x_i: Is x_i from **category** c?
     - Yes: go to the left child node
     - No: go to the right child node
   - Numerical x_i: Is $x_i \leq$ **t**?
     - Yes: go to the left child node
     - No: go to the right child node
3. Is it possible to have multi-way splits instead of binary (left/right)?
   - Yes, but they are in general more complex. We focus on binary splits in this lecture

Parameter estimation (CART) for categorical variables

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by splitting each node until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by splitting each node until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This split will create two child nodes (groups): left and right

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the **impurity**

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by splitting each node until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This split will create two child nodes (groups): left and right
- Each child node will contain a (sub) training set
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to split a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the impurity
- Impurity: after a split, how "mixed" the (sub) training sets are in terms of their classes

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the **impurity**
- **Impurity:** after a split, how "mixed" the **(sub) training sets** are in terms of their classes
- Example (three classes in total $C = 3$): after a split, the left child node contains 20 data points from class 0, 15 from class 1, 5 from class 2,

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the **impurity**
- **Impurity:** after a split, how "mixed" the **(sub) training sets** are in terms of their classes
- Example (three classes in total $C = 3$): after a split, the left child node contains 20 data points from class 0, 15 from class 1, 5 from class 2, i.e. left=[20, 15, 5];

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the **impurity**
- **Impurity:** after a split, how "mixed" the **(sub) training sets** are in terms of their classes
- Example (three classes in total $C = 3$): after a split, the left child node contains 20 data points from class 0, 15 from class 1, 5 from class 2, i.e. left=[20, 15, 5]; similarly, right=[0, 35, 0];

Parameter estimation (CART) for categorical variables

- Start with the root node that contains the entire training set
- **Grow the tree** by **splitting each node** until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a child node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain **a (sub) training set**
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l == c\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r != c\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting category c

  by minimizing the **impurity**
- **Impurity:** after a split, how "mixed" the **(sub) training sets** are in terms of their classes
- Example (three classes in total $C = 3$): after a split, the left child node contains 20 data points from class 0, 15 from class 1, 5 from class 2, i.e. left=[20, 15, 5]; similarly, right=[0, 35, 0]; the right child node is more pure than the left child node because the left child node is more "mixed" in terms of the classes

Impurity

- There are alternative impurity measures
  - **Gini score (used by CART):** $\sum_{b=1}^{C} \hat{p}_b(1-\hat{p}_b)$
  - **Entropy:** $-\sum_{b=1}^{C} \hat{p}_b \log \hat{p}_b$

  where $\hat{p}_b$ is estimated by the proportion (count) in each class b in the node

Impurity

- There are alternative impurity measures
  - **Gini score (used by CART):** $\sum_{b=1}^{C} \hat{p}_b(1-\hat{p}_b)$
  - **Entropy:** $-\sum_{b=1}^{C} \hat{p}_b \log \hat{p}_b$

  where $\hat{p}_b$ is estimated by the proportion (count) in each class b in the node
- Previous example: left=[20, 15, 5] and right=[0, 35, 0]; in total, $N_{\text{left}} = 20+15+5 = 40$ and $N_{\text{right}} = 0+35+0 = 35$
  - $G_{\text{left}} = \frac{20}{40}(1-\frac{20}{40}) + \frac{15}{40}(1-\frac{15}{40}) + \frac{5}{40}(1-\frac{5}{40}) = 0.59$
  - $G_{\text{right}} = \frac{0}{35}(1-\frac{0}{35}) + \frac{35}{35}(1-\frac{35}{35}) + \frac{0}{35}(1-\frac{0}{35}) = 0$

Impurity

- There are alternative impurity measures
  - **Gini score (used by CART):** $\sum_{b=1}^{C} \hat{p}_b(1-\hat{p}_b)$
  - **Entropy:** $-\sum_{b=1}^{C} \hat{p}_b \log \hat{p}_b$

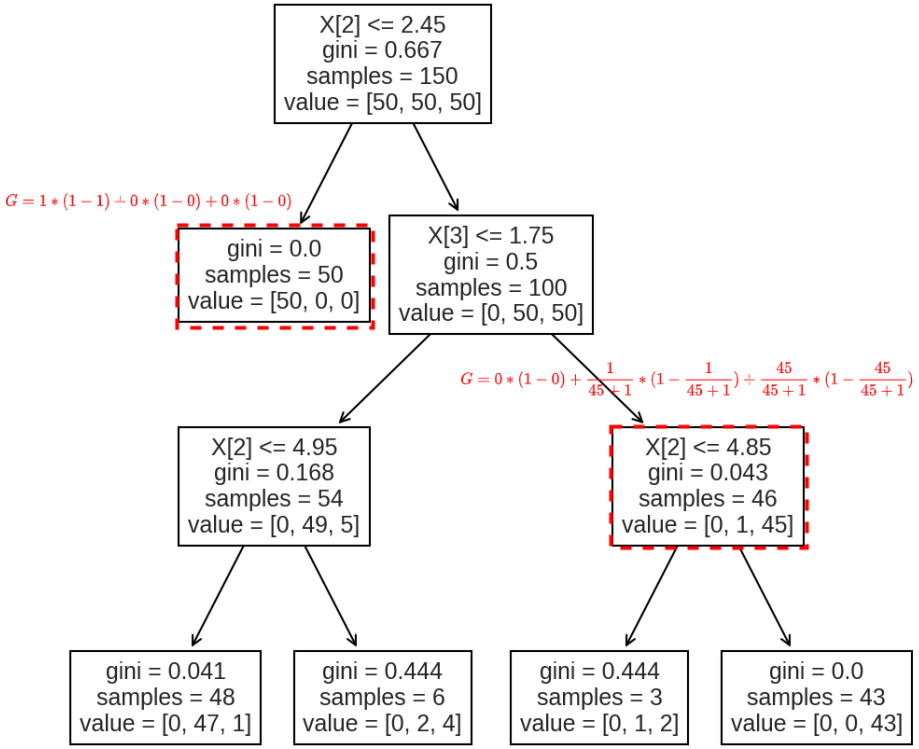
  where $\hat{p}_b$ is estimated by the proportion (count) in each class b in the node
- Previous example: left=[20, 15, 5] and right=[0, 35, 0]; in total, $N_{\text{left}} = 20+15+5 = 40$ and $N_{\text{right}} = 0+35+0 = 35$
  - $G_{\text{left}} = \frac{20}{40}(1-\frac{20}{40}) + \frac{15}{40}(1-\frac{15}{40}) + \frac{5}{40}(1-\frac{5}{40}) = 0.59$
  - $G_{\text{right}} = \frac{0}{35}(1-\frac{0}{35}) + \frac{35}{35}(1-\frac{35}{35}) + \frac{0}{35}(1-\frac{0}{35}) = 0$
- Impurity of the split: $G = \frac{N_{\text{left}}}{N_{\text{left}}+N_{\text{right}}} G_{\text{left}} + \frac{N_{\text{right}}}{N_{\text{left}}+N_{\text{right}}} G_{\text{right}}$

Impurity

- There are alternative impurity measures
  - **Gini score (used by CART):** $\sum_{b=1}^{C} \hat{p}_b(1-\hat{p}_b)$
  - **Entropy:** $-\sum_{b=1}^{C} \hat{p}_b \log \hat{p}_b$

  where $\hat{p}_b$ is estimated by the proportion (count) in each class b in the node
- Previous example: left=[20, 15, 5] and right=[0, 35, 0]; in total, $N_{\text{left}} = 20+15+5 = 40$ and $N_{\text{right}} = 0+35+0 = 35$
  - $G_{\text{left}} = \frac{20}{40}(1-\frac{20}{40}) + \frac{15}{40}(1-\frac{15}{40}) + \frac{5}{40}(1-\frac{5}{40}) = 0.59$
  - $G_{\text{right}} = \frac{0}{35}(1-\frac{0}{35}) + \frac{35}{35}(1-\frac{35}{35}) + \frac{0}{35}(1-\frac{0}{35}) = 0$
- Impurity of the split: $G = \frac{N_{\text{left}}}{N_{\text{left}}+N_{\text{right}}} G_{\text{left}} + \frac{N_{\text{right}}}{N_{\text{left}}+N_{\text{right}}} G_{\text{right}}$
  - $G = \frac{40}{40+35} G_{\textbf{left}} + \frac{35}{40+35} G_{\textbf{right}} = 0.53 * 0.59 + 0.467 * 0 = 0.3127$

Exercise: compute the Gini score for each node



Gini: $G = \sum_{c=1}^{C} \hat{p}_{mc}(1 - \hat{p}_{mc})$

Choose the best feature i and category c

Use the impurity score to choose the best i and c

- Loop over all $i = 1, \cdots, d$
- For each i, loop over all categories $c = 0, \cdots, C_i - 1$

For each choice in the loop, compute the impurity score and choose the i and c that minimizes the impurity.

Pseudocode for categorical variables (CART)

- For each node:
  - Choose the best feature index i
  - Choose the best category c

```
for i in [1, ··· , d] do
    for c in [0, ··· , C_i − 1] do
        Compute the Gini score
    end for
end for
Find the best i and c corresponding to the smallest Gini score
```

where C is the number of classes and C_i is the number of categories for feature x_i

- After the tree is constructed,

$$\hat{c}_m = \arg\max_c N_m^c, \ \forall c \in \{0, \cdots, C-1\}$$

where N_m^c is the number of data points that are 1) from class c and 2) assigned to region (leaf node) R_m
Example ($C = 3$): in region 2, if $[N_2^0, N_2^1, N_2^2] = [10, 25, 0]$, then $\hat{c}_2 = 1$ since 25 is the largest count.

Parameter estimation (CART) for numerical variables

Similar to categorical variables with minor **modifications**

- Start with the root node that contains the entire training set
- **Grow the tree** by splitting each node until a stopping criterion is met, e.g. maximum depth, minimum number of data assigned to a leaf node, etc; this is a **recursive procedure**
- This **split** will create two child nodes (groups): left and right
- Each child node will contain a (sub) training set
  - $\mathcal{I}_{\text{left}} \leftarrow \{l \mid x_i^l \leq t\}$: all indices assigned to the left child node
  - $\mathcal{I}_{\text{right}} \leftarrow \{r \mid x_i^r > t\}$: all indices assigned to the right child node
- In order to **split** a node into the left and right child nodes, we need to estimate
  - The best feature x_i for the split
  - The splitting threshold t (**modification**)

  by minimizing the **impurity**
- To estimate x_i and t (**modification**)
  - Loop over all $i = 1, \cdots, d$
  - For each feature index i, let $t \leftarrow x_i^j$, loop over all data in the (sub) training sample in the node

Pseudocode for numerical variables (CART)

- For each node:
  - Choose the best feature index i
  - Choose the best threshold t

```
for i in [1, ··· , d] do
    for j in [1, ··· , N_node] do
        t ← x_i^j
        Compute the Gini score
    end for
end for
Find the best i and t corresponding to the smallest Gini score
```

where N_{node} is the number of training data points in the current node

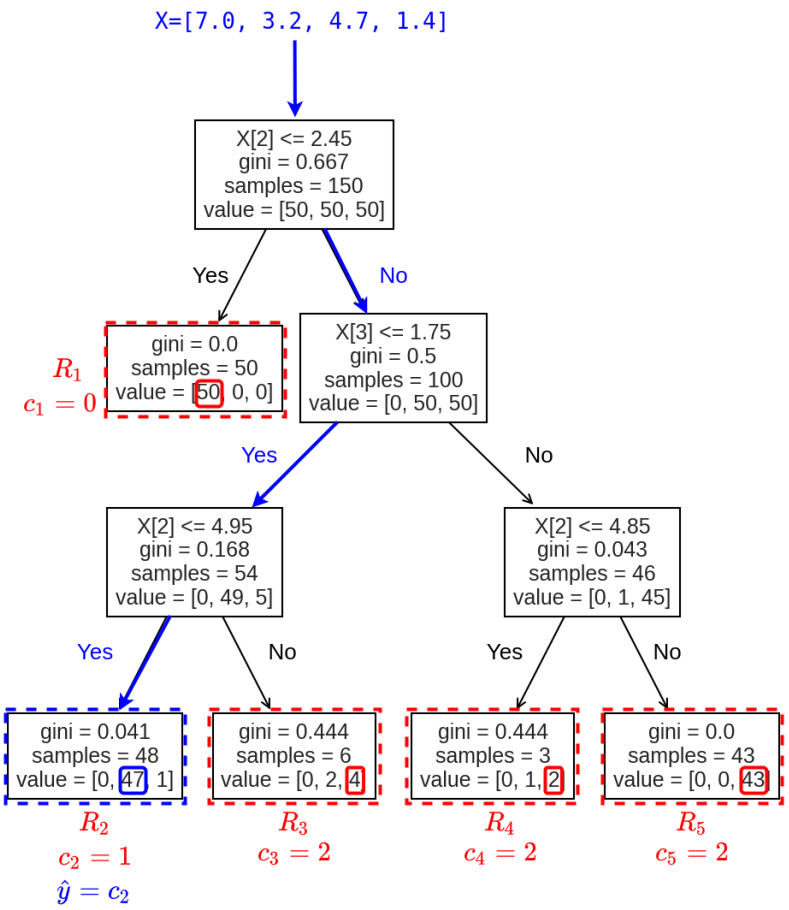
- After the tree is constructed,

$$\hat{c}_m = \arg\max_c N_m^c, \ \forall c \in \{0, \cdots, C-1\}$$

where N_m^c is the number of data points that are 1) from class c and 2) assigned to region (leaf node) R_m

Prediction with a trained decision tree

For a new data point $X = [7.0, 3.2, 4.7, 1.4]$, $\hat{y} = \sum_{m=1}^{5} c_m \boldsymbol{I}_{R_i}(\boldsymbol{x})$



Regression tree

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:
  - y: **continuous numerical**, a scalar or a high dimensional array

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:
  - y: **continuous numerical**, a scalar or a high dimensional array
  - x: typically **continuous numerical**; feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:
  - y: **continuous numerical**, a scalar or a high dimensional array
  - x: typically **continuous numerical**; feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
  - g: **regression model**, e.g. linear regression, regression tree, support vector regression, etc

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:
  - y: **continuous numerical**, a scalar or a high dimensional array
  - x: typically **continuous numerical**; feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
  - g: **regression model**, e.g. linear regression, regression tree, support vector regression, etc
  - θ (parameters) and h (hyperparameters) depend on g

Mathematical modeling for regression

- Modeling for regression

$$y = g(x; \theta \mid h)$$

- Regression:
  - y: **continuous numerical**, a scalar or a high dimensional array
  - x: typically **continuous numerical**; feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
  - g: **regression model**, e.g. linear regression, regression tree, support vector regression, etc
  - θ (parameters) and h (hyperparameters) depend on g
- Parameter estimation: supervised learning

Regression decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$

Regression decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **continuous** scalar

$$y = \sum_{m=1}^{M} a_m \boldsymbol{I}_{R_i}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_i}$ is the indicater function:

$$\boldsymbol{I}_{R_i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_i \\ 0, & \text{otherwise} \end{cases}$$

Regression decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **continuous** scalar

$$y = \sum_{m=1}^{M} a_m \boldsymbol{I}_{R_i}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_i}$ is the indicater function:

$$\boldsymbol{I}_{R_i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_i \\ 0, & \text{otherwise} \end{cases}$$

- **Hyperparameter**: stopping criteria (e.g. maximum depth), hyperparameters for pruning, etc

Regression decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **continuous** scalar

$$y = \sum_{m=1}^{M} a_m \boldsymbol{I}_{R_i}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_i}$ is the indicater function:

$$\boldsymbol{I}_{R_i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_i \\ 0, & \text{otherwise} \end{cases}$$

- **Hyperparameter**: stopping criteria (e.g. maximum depth), hyperparameters for pruning, etc
- **Parameter**:
  - All non-leaf nodes in the tree
  - Regions R_m (leaf nodes) for $m = 1, \cdots, M$
  - Coefficients a_m for $m = 1, \cdots, M$

Regression decision tree

- **Data $\boldsymbol{x}$**: a d dimensional **continuous numerical** feature vector $\boldsymbol{x} = [x_1, \cdots, x_d]$
- **Target y**: a **continuous** scalar

$$y = \sum_{m=1}^{M} a_m \boldsymbol{I}_{R_i}(\boldsymbol{x})$$

where $\boldsymbol{I}_{R_i}$ is the indicater function:

$$\boldsymbol{I}_{R_i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in R_i \\ 0, & \text{otherwise} \end{cases}$$

- **Hyperparameter**: stopping criteria (e.g. maximum depth), hyperparameters for pruning, etc
- **Parameter**:
  - All non-leaf nodes in the tree
  - Regions R_m (leaf nodes) for $m = 1, \cdots, M$
  - Coefficients a_m for $m = 1, \cdots, M$
- **Parameter estimation**: the classification and regression tree (CART) algorithm

Parameter estimation

- Similar to parameter estimation for classification trees
- The difference is the score being calculated
- Instead of using the impurity (e.g. Gini score), which is defined for classification, we use the sum of squared error instead

Pseudocode for regression tree (CART)

- For each node:
  - Choose the best feature i
  - Choose the best threshold t

```
for i in [1, ··· , d] do
    for j in [1, ··· , N_node] do
        t ← x_i^j
        I_left ← {l | x_i^l ≤ t}: all indices assigned to the left child node
        I_right ← {r | x_i^r > t}: all indices assigned to the right child node
        Let a_left ← ȳ_l and a_right ← ȳ_r, where ȳ_l and ȳ_r are the sample means of the left and right child nodes,
        respectively.
        Compute s = Σ_{l∈I_left} (y_l − a_left)^2 + Σ_{r∈I_right} (y_r − a_right)^2
    end for
end for
Find the best i and t corresponding to the lowest score s
```

where N_{node} is the number of training data points in the current node

- After the tree is constructed,

$$\hat{a}_m = \frac{1}{N_m} \sum_{i=1}^{N_m} y_{m_i}$$

where N_m is the number of data points assigned to region (leaf node) R_m and y_{m_i} denotes labels of data points assigned to R_m

Pruning

Overfitting

Without modification, decision trees suffer from overfitting

Overfitting

Without modification, decision trees suffer from overfitting - need to reduce the model complexity using, e.g. pruning

Overfitting

Without modification, decision trees suffer from overfitting - need to reduce the model complexity using, e.g. pruning

- Pre-pruning (top-down, early stopping):
  This can be done using, e.g. cross-validation at each node split during the tree construction process

Overfitting

Without modification, decision trees suffer from overfitting - need to reduce the model complexity using, e.g. pruning

- Pre-pruning (top-down, early stopping):
  This can be done using, e.g. cross-validation at each node split during the tree construction process
- Post-pruning (bottom-up, remove leaves after the tree is constructed):
  - This can be done by, e.g. removing leaves one by one until the **optimal** tree is found
  - The **optimality** is defined by a loss function L, e.g.

$$L = \text{score} + \alpha * \text{complexity}$$

Overfitting

Without modification, decision trees suffer from overfitting - need to reduce the model complexity using, e.g. pruning

- Pre-pruning (top-down, early stopping):
  This can be done using, e.g. cross-validation at each node split during the tree construction process
- Post-pruning (bottom-up, remove leaves after the tree is constructed):
  - This can be done by, e.g. removing leaves one by one until the **optimal** tree is found
  - The **optimality** is defined by a loss function L, e.g.

$$L = \text{score} + \alpha * \text{complexity}$$

Typically, there are additional hyperparameters (e.g. α) to tune using, e.g. cross-validation; a commonly used technique is **cost complexity pruning**, where complexity is defined as the number of leaf nodes.

ID3, C4.5, C5.0, CART

Comparison of parameter estimation algorithms

They are parameter estimation algorithms in the literature.

Algorithm	ID3	C4.5	C5.0	CART
Variable type	Categorical	Continuous and categorical	Continuous and categorical	Continuous and categorical
Target type	Categorical	Categorical	Categorical	Continuous and categorical
Splitting score	Information gain	Information gain	Information gain	Gini impurity
Computational efficiency	Low	Okay	High	Good

Comparison of parameter estimation algorithms

They are parameter estimation algorithms in the literature.

Algorithm	ID3	C4.5	C5.0	CART
Variable type	Categorical	Continuous and categorical	Continuous and categorical	Continuous and categorical
Target type	Categorical	Categorical	Categorical	Continuous and categorical
Splitting score	Information gain	Information gain	Information gain	Gini impurity
Computational efficiency	Low	Okay	High	Good

- In summary, NO BIGGY! Read the documentation (and the literature) before using them

Comparison of parameter estimation algorithms

They are parameter estimation algorithms in the literature.

Algorithm	ID3	C4.5	C5.0	CART
Variable type	Categorical	Continuous and categorical	Continuous and categorical	Continuous and categorical
Target type	Categorical	Categorical	Categorical	Continuous and categorical
Splitting score	Information gain	Information gain	Information gain	Gini impurity
Computational efficiency	Low	Okay	High	Good

- In summary, NO BIGGY! Read the documentation (and the literature) before using them
- For example, the scikit-learn documentation (December 2021): `https://scikit-learn.org/stable/modules/tree.html`

Comparison of parameter estimation algorithms

They are parameter estimation algorithms in the literature.

Algorithm	ID3	C4.5	C5.0	CART
Variable type	Categorical	Continuous and categorical	Continuous and categorical	Continuous and categorical
Target type	Categorical	Categorical	Categorical	Continuous and categorical
Splitting score	Information gain	Information gain	Information gain	Gini impurity
Computational efficiency	Low	Okay	High	Good

- In summary, NO BIGGY! Read the documentation (and the literature) before using them
- For example, the scikit-learn documentation (December 2021): `https://scikit-learn.org/stable/modules/tree.html`
- The most important thing is to understand the different components in the algorithms, e.g. splitting score, splitting criteria, maximum tree depth, etc

Summary

- Pros:
  - Interpretable, visualizable, intuitive
  - Do not require preprocessing
  - Handle categorical and continuous variables
  - Fairly simple to implement
- Cons:
  - High variance, overfitting
  - Not robust

Today

1. Decision trees
   - Classification tree
   - Regression tree
   - Pruning
   - ID3, C4.5, C5.0, CART
2. Ensemble methods
   - Bias-variance trade-off
   - Bagging
   - Random forest
3. Summary

Bias-variance trade-off

Bias-variance trade-off

What are bias and variance in this context?

Bias-variance trade-off

What are bias and variance in this context? - loosely speaking,

Bias-variance trade-off

What are bias and variance in this context? - loosely speaking,

- Bias indicates how far-off the estimated target is compared to the true target

Bias-variance trade-off

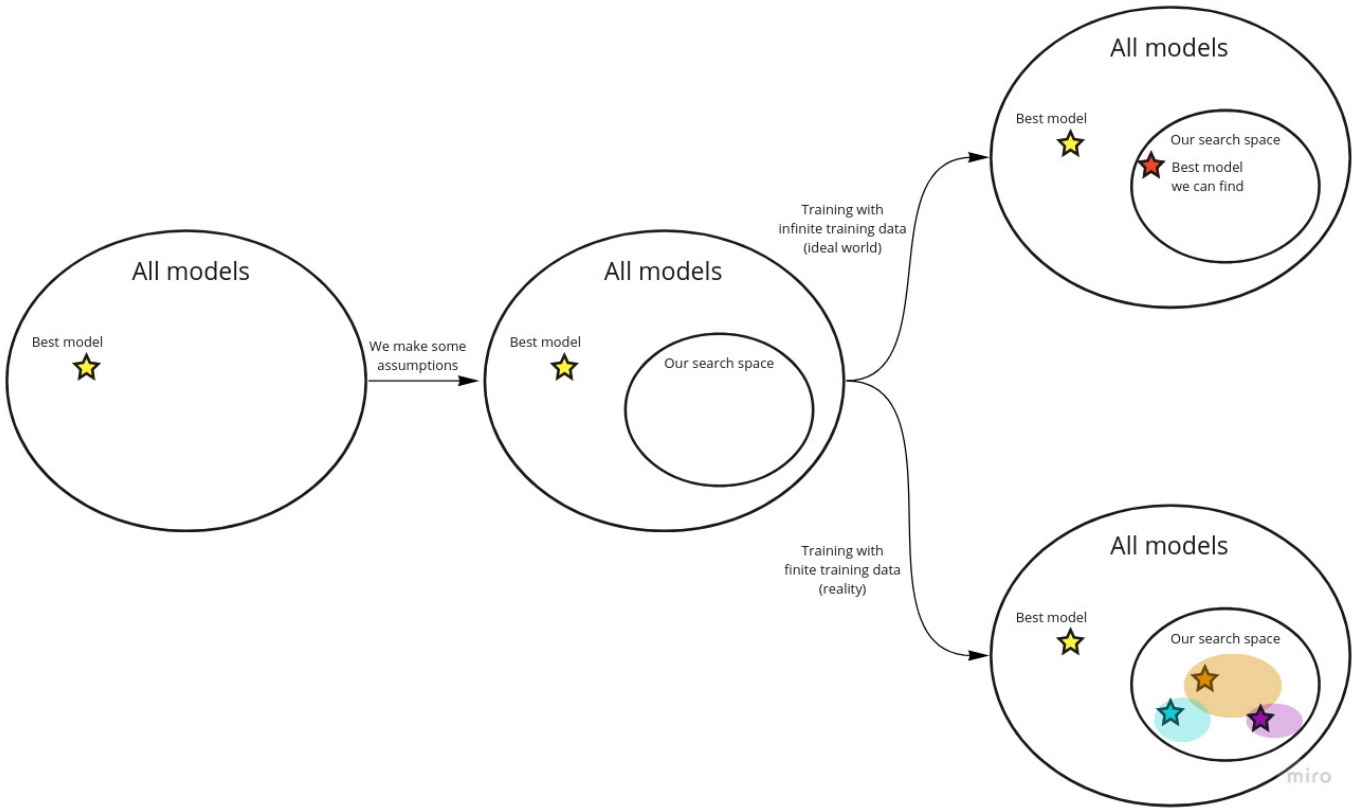
What are bias and variance in this context? - loosely speaking,

- Bias indicates how far-off the estimated target is compared to the true target
- Variance indicates how much the estimated target changes if a different training set is used

Bias-variance trade-off

What are bias and variance in this context? - loosely speaking,

- Bias indicates how far-off the estimated target is compared to the true target
- Variance indicates how much the estimated target changes if a different training set is used



Bias-variance trade-off (cont.)

- Complex models tend to have low bias and high variance

Bias-variance trade-off (cont.)

- Complex models tend to have low bias and high variance
- Simple models tend to have high bias and low variance

Bias-variance trade-off (cont.)

- Complex models tend to have low bias and high variance
- Simple models tend to have high bias and low variance
- Ensemble methods combine multiple models to produce a better result - with reduced bias or variance

Bagging

Bagging

- Purpose: to reduce variance

Bagging

- Purpose: to reduce variance
- General idea behind bagging

Bagging

- Purpose: to reduce variance
- General idea behind bagging
  - Let $\hat{Y}$ be the random variable that describes $\hat{y}$ with variance σ^2

Bagging

- Purpose: to reduce variance
- General idea behind bagging
  - Let $\hat{Y}$ be the random variable that describes $\hat{y}$ with variance σ^2
  - If we use the sample mean $\hat{Y} \leftarrow \bar{Y}$ with sample size, say, 100, the variance for $\hat{Y}$ is $\frac{\sigma^2}{100}$ - the variance is reduced (central limit theorem)

Bagging

- Purpose: to reduce variance
- General idea behind bagging
  - Let $\hat{Y}$ be the random variable that describes $\hat{y}$ with variance σ^2
  - If we use the sample mean $\hat{Y} \leftarrow \bar{Y}$ with sample size, say, 100, the variance for $\hat{Y}$ is $\frac{\sigma^2}{100}$ - the variance is reduced (central limit theorem)
  - Note: this variance cannot be reduced to zero

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample - we have seen this (L6) 

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample - we have seen this (L6) 
  Draw N data pairs $(\boldsymbol{x}_i, y_i)$ **with replacement** from the training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample - we have seen this (L6)
  Draw N data pairs $(\boldsymbol{x}_i, y_i)$ **with replacement** from the training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$
  For example, given a training data set

$$\{(1,0),(2,0),(3,1),(4,1),(5,1)\},$$

  a bootstrapped sample could be

$$\{(1,0),(3,1),(1,0),(1,0),(5,1)\}$$

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample - we have seen this (L6) 
  Draw N data pairs $(\boldsymbol{x}_i, y_i)$ **with replacement** from the training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$
  For example, given a training data set

$$\{(1,0),(2,0),(3,1),(4,1),(5,1)\},$$

  a bootstrapped sample could be

$$\{(1,0),(3,1),(1,0),(1,0),(5,1)\}$$

- Step 2: Build a tree based on the bootstrapped sample

Bagging: parameter estimation (training)

- Step 1: Create a bootstrapped sample - we have seen this (L6) 
  Draw N data pairs $(\boldsymbol{x}_i, y_i)$ **with replacement** from the training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$
  For example, given a training data set

$$\{(1,0),(2,0),(3,1),(4,1),(5,1)\},$$

  a bootstrapped sample could be

$$\{(1,0),(3,1),(1,0),(1,0),(5,1)\}$$

- Step 2: Build a tree based on the bootstrapped sample
- Step 3: Repeat step 1 and 2 B times, which results in B trees

Bagging: prediction

Aggregate the predictions from all B trees

- Regression tree (sample mean):

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b^*$$

- Classification tree (majority vote):

$$\hat{y} = \arg\max_c \text{count}(\hat{y}_b^* == c)$$

Random forest

Random forest

Similar to bagging with one addition +

- Step 1: Create a bootstrapped sample
- Step 2: Build a tree based on the bootstrapped sample
  + Randomly choose $\tilde{d} \approx \sqrt{d}$ at each node and only use $\tilde{d}$ features for the split (**decorrelation**)
- Step 3: Repeat step 1 and 2 B times, which results in B trees

Compared to decision trees, random forest results in significantly lower variance with slightly increased bias; less interpretable

Today

1. Decision trees
   - Classification tree
   - Regression tree
   - Pruning
   - ID3, C4.5, C5.0, CART
2. Ensemble methods
   - Bias-variance trade-off
   - Bagging
   - Random forest
3. Summary

Summary

- Data types and data containers
- Descriptive data analysis: descriptive statistics, visualization
- Probability distributions, events, random variables, PMF, PDF, parameters
- CDF, Q-Q plot, how to compare two distributions (data vs theoretical, data vs data)
- Modeling
- Parameter estimation: maximum likelihood estimation (MLE) and maximum a posteriori estimation (MAP)
- Classification, multinomial naive Bayes classifier, Gaussian naive Bayes classifier
- Central limit theorem, interval estimation
- Hypothesis tests, comparison of two classifiers
- Clustering, cluster tendency, k-means
- SSE and Silhouette score for cluster evaluation, one dimensional Gaussian Mixture Models, AIC/BIC, the EM algorithm, clustering validation
- Decision tree for classification and regression, random forest

Stay safe, happy, curious and enthusiastic!