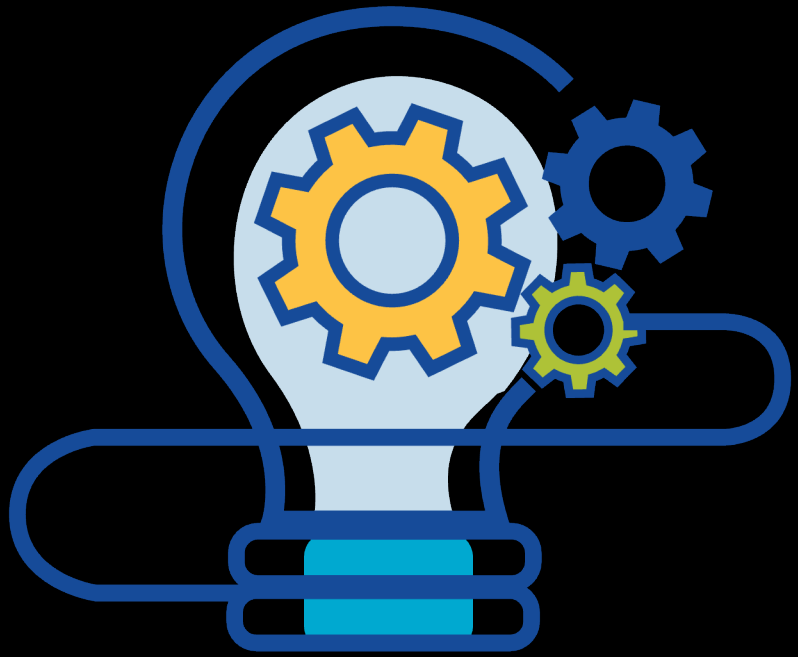


Decision Tree

Mingsheng Long

Tsinghua University

Fall 2025

Outline

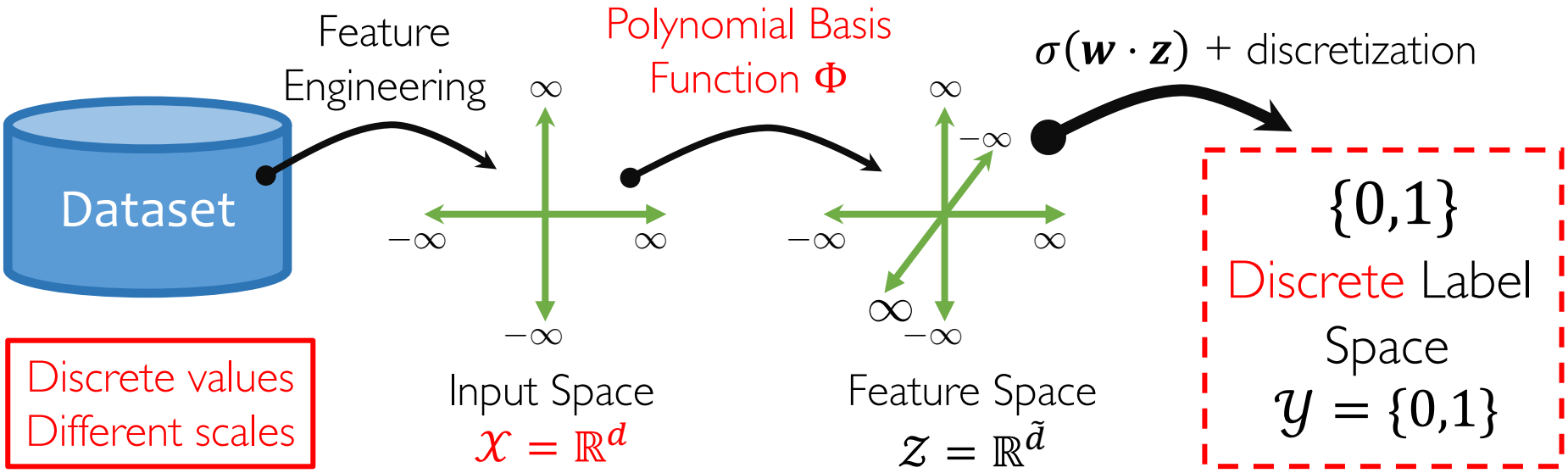
- **Decision Tree**
  - ID3 Algorithm
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - Generalization Bound
- Random Forest
  - Bagging
  - Breiman Algorithm

Linear Model

2-class Classification:

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

How to do with feature heterogeneity and achieve model interpretability?

Human Classifier

- How do people solve this problem?

Do we combine features as in LM?

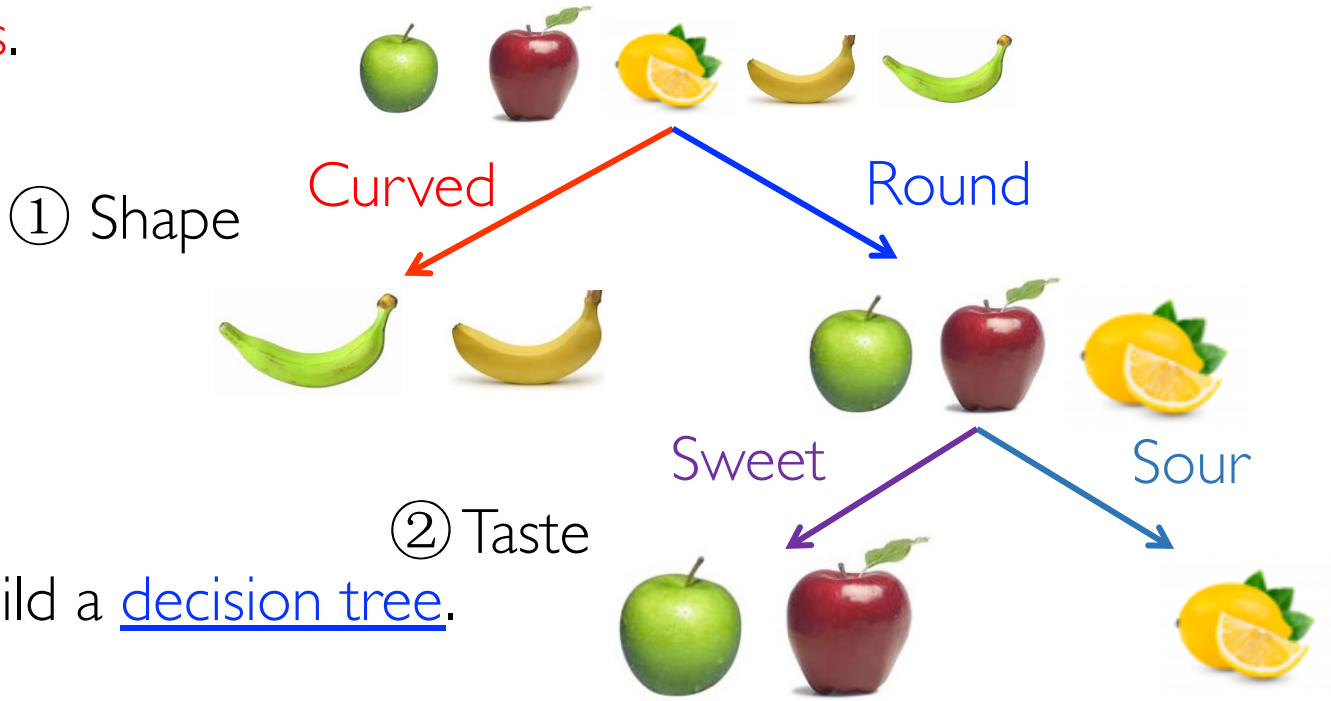
green, round, no leaf, sweet, …	Apple
red, round, leaf, sweet, …	Apple
yellow, round, leaf, sour, …	Lemon
yellow, curved, no leaf, sweet, …	Banana
green, curved, no leaf, sweet, …	Banana

Feature

Label

Human Classifier

- We find a most useful feature, such as *shape*: curved or round.
  - Split the dataset: If the fruit is curved, then it is a banana.
- After the first split, select next best feature until each node contains only one class.



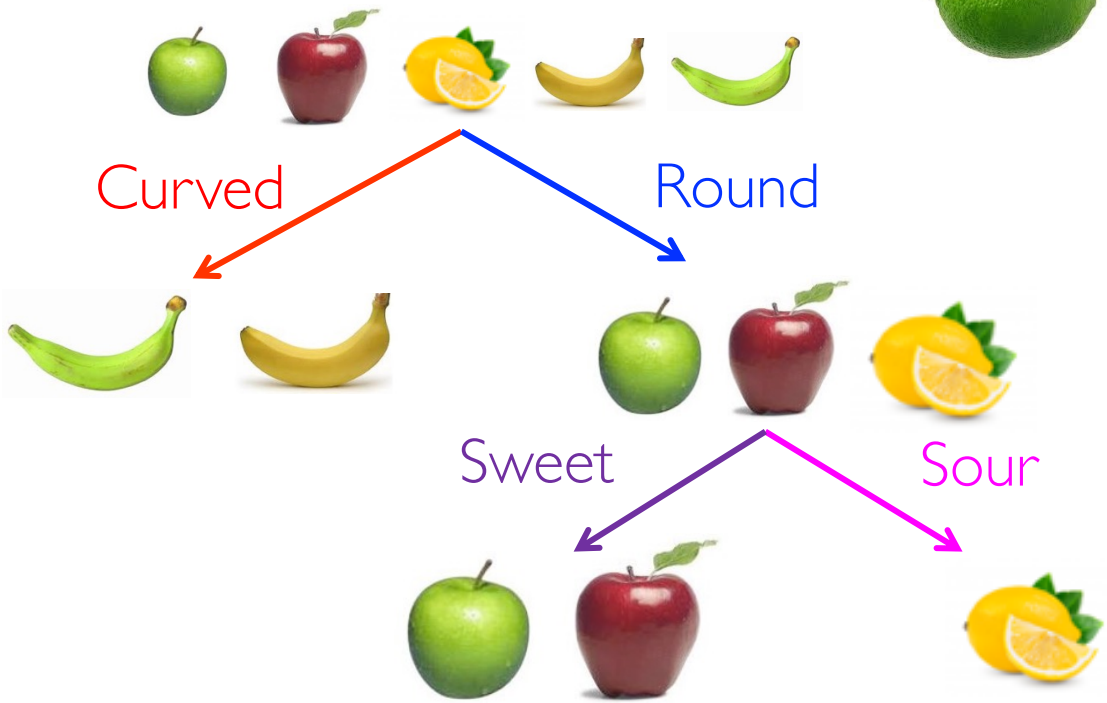
- At last, we build a decision tree.

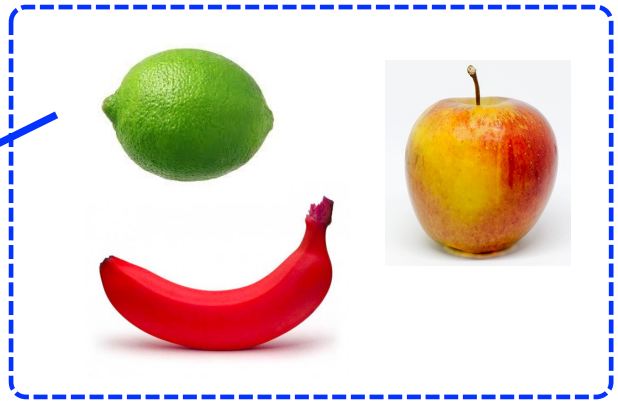
Decision Tree

Interpretability ✓

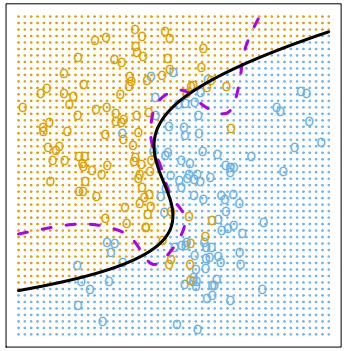
Generalize to unseen data ✓

- During test:
  - Why do you think this fruit is lemon?
  - Because it is round and sour.





Compared with LM:

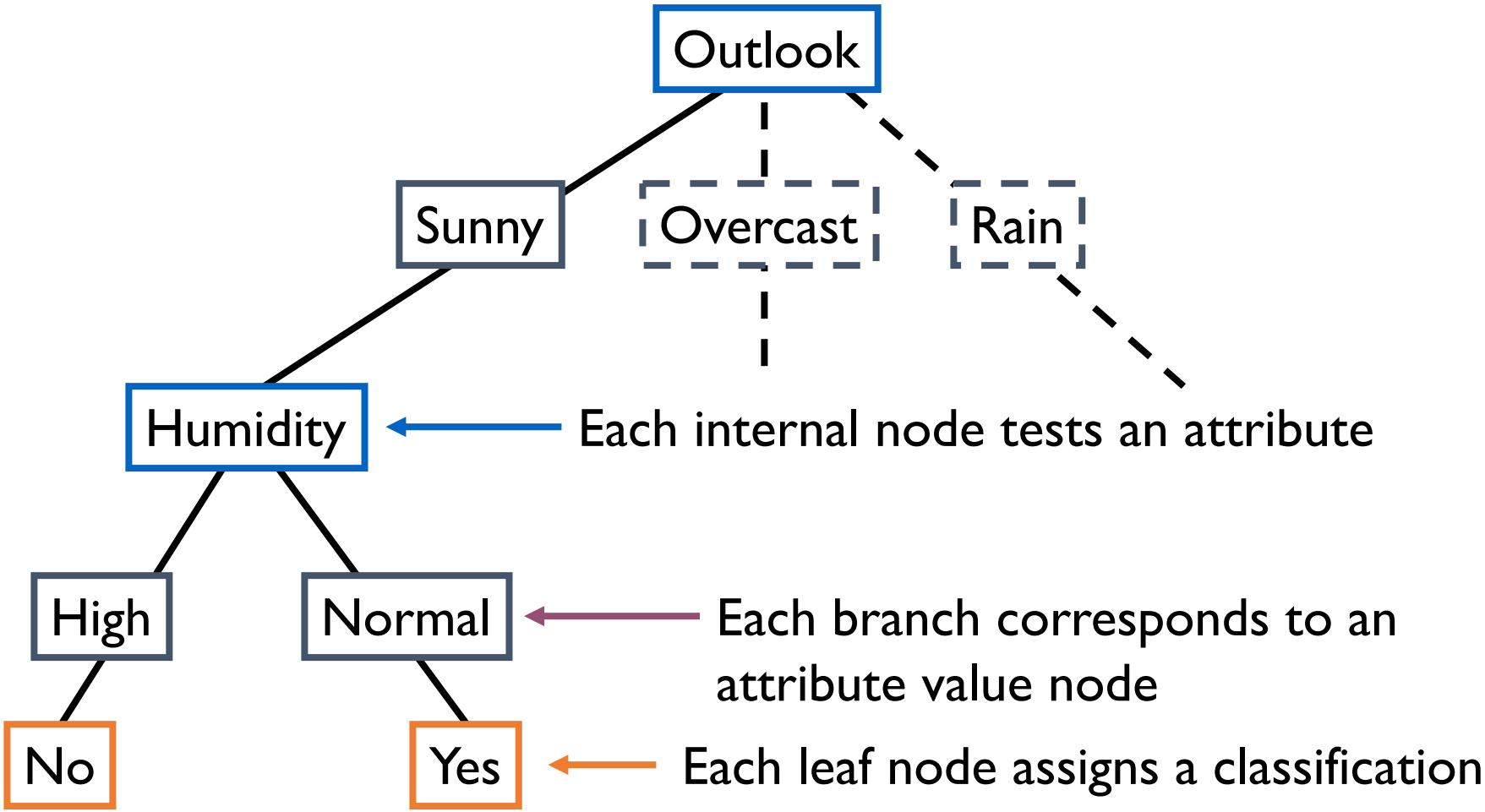


Decision Tree

- When we face many features, it is hard for us to build tree by hand.
- Can we teach machine to find it? What are the difficulties?
  - How to find the most useful feature on each node?
  - When should we stop growing the tree?
  - What if some features are missing or continuous-valued?

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

Decision Tree

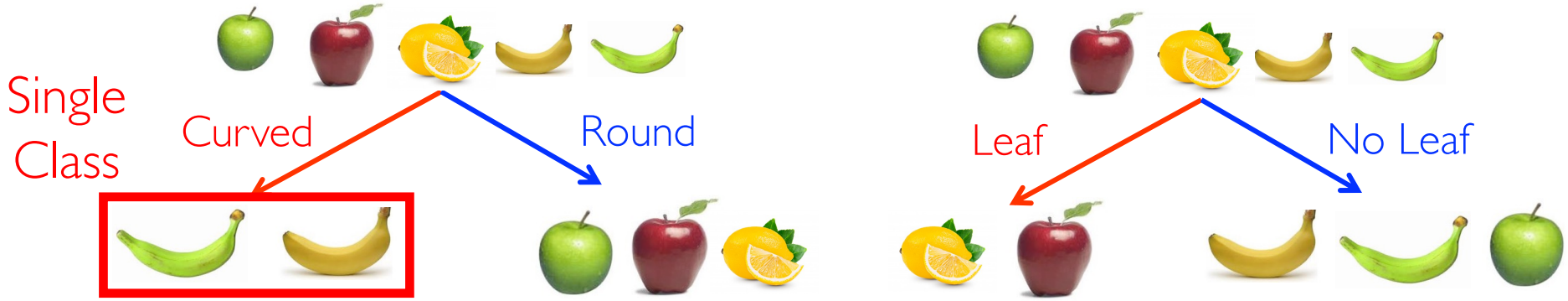


Outline

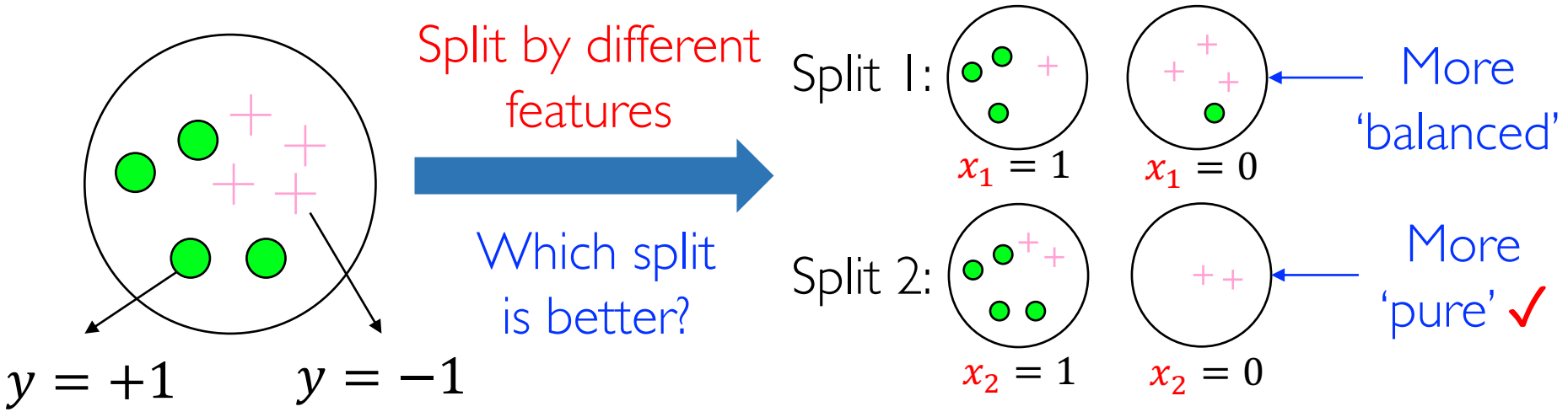
- **Decision Tree**
  - **ID3 Algorithm**
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - Generalization Bound
- Random Forest
  - Bagging
  - Breiman Algorithm

Node Splitting

- Which split is better?



- As example, we visualize splits in a binary classification problem:

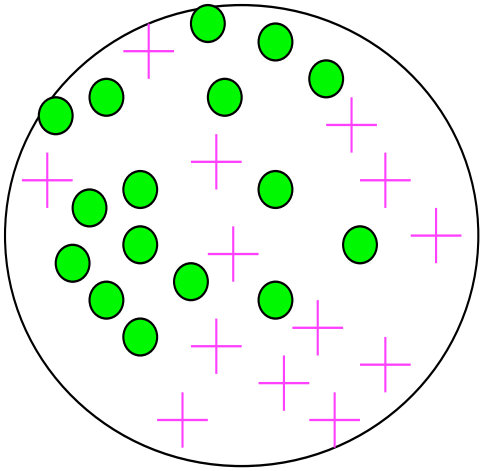


How to Measure a Node

- We want pure leaf nodes, as close to a single class as possible:
  - Lead to a lower classification error.
  - The classification error is $\min(\frac{|\mathcal{C}_1|}{|\mathcal{D}|}, \frac{|\mathcal{C}_2|}{|\mathcal{D}|})$.

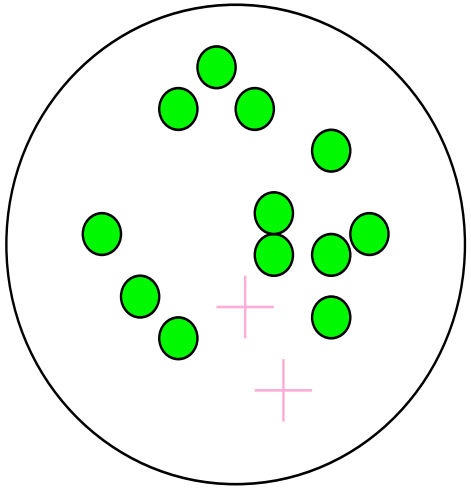
Number of samples in each class

- We'll choose the splitting feature that minimizes impurity measure.



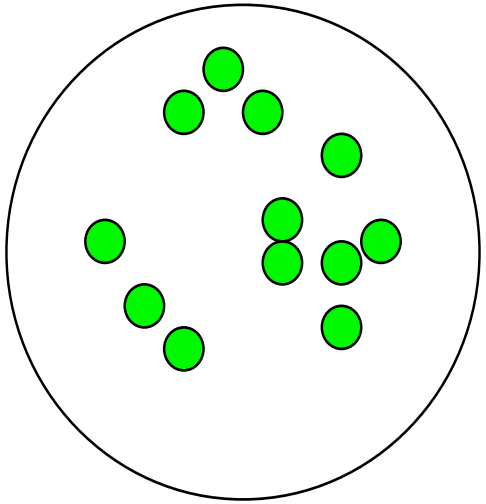
Very impure group

Error: $13/29$



Less impure

Error: $2/14$



Minimum impurity

Error: 0

Node Impurity Measures

- Three standard node impurity measures:
  - Misclassification error:

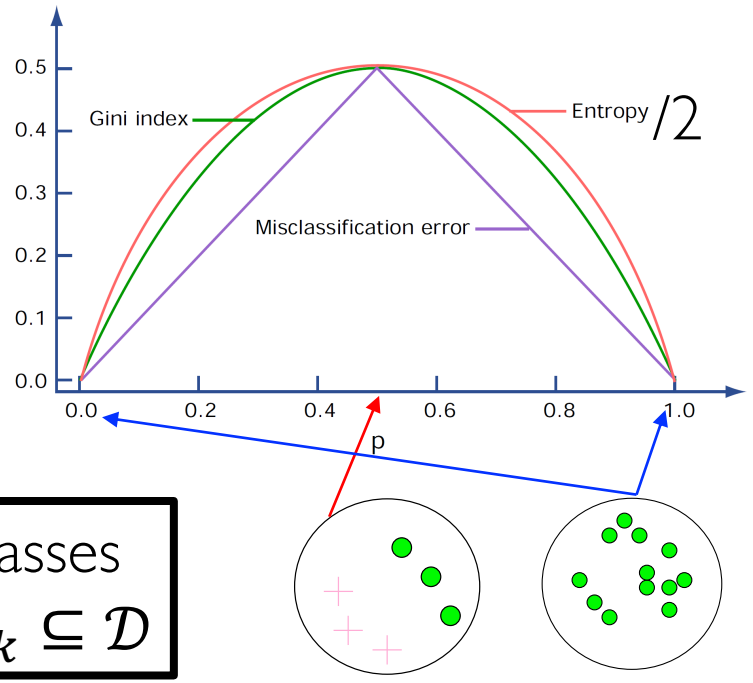
$$\mathrm{Err}(\mathcal{D}) = 1 - \max_{1 \le k \le K} \left(\frac{|\mathcal{C}_k|}{|\mathcal{D}|} \right)$$

  - Entropy (used in ID3 and C4.5):

$$H(\mathcal{D}) = -\sum_{k=1}^{K} \frac{|\mathcal{C}_k|}{|\mathcal{D}|} \log \frac{|\mathcal{C}_k|}{|\mathcal{D}|}$$

  - Gini index (used in CART):

$$\mathrm{Gini}(\mathcal{D}) = 1 - \sum_{k=1}^{K} \left(\frac{|\mathcal{C}_k|}{|\mathcal{D}|} \right)^2$$



K #classes
Each $\mathcal{C}_k \subseteq \mathcal{D}$

Max: $\frac{|\mathcal{C}_k|}{|\mathcal{D}|} = 0.5$

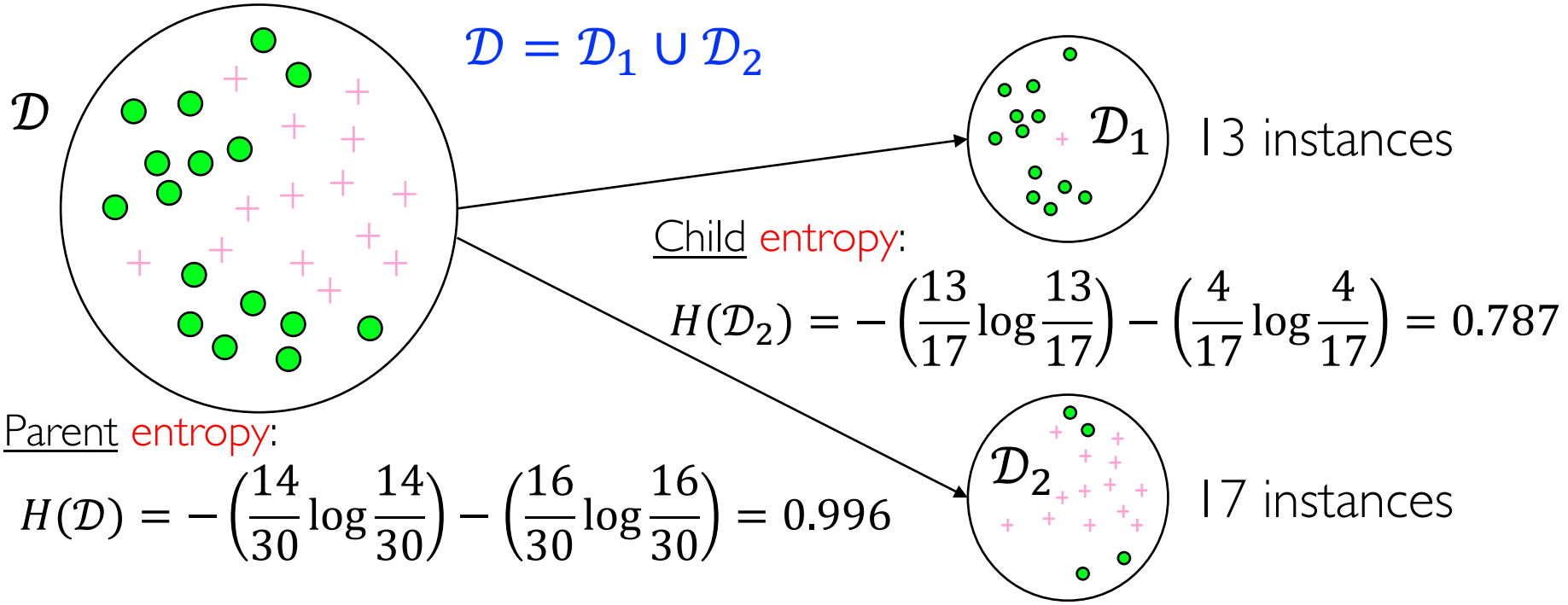
Min: $\frac{|\mathcal{C}_k|}{|\mathcal{D}|} = 0$

$(0 \log 0 = 0)$

How to Measure a Split

Entire dataset (30 instances)

$$H(\mathcal{D}_1) = -\left(\frac{1}{13}\log\frac{1}{13}\right) - \left(\frac{12}{13}\log\frac{12}{13}\right) = 0.391$$



- We want to measure this split – how this split minimizes entropy.
  - How about $H(\mathcal{D}) - (H(\mathcal{D}_1) + H(\mathcal{D}_2))/2$?
  - But the instance number on each node is different → Weighting!

Information Gain (IG)

- A good split gives minimal weighted average of node impurities:

$$\frac{|\mathcal{D}_1|}{|\mathcal{D}|} H(\mathcal{D}_1) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|} H(\mathcal{D}_2)$$

- Equivalent to maximizing the Information Gain (IG):

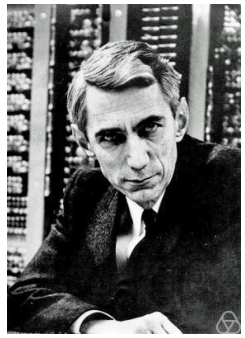
Child nodes

Parent node

$$H(\mathcal{D}_1 \cup \mathcal{D}_2) - \frac{|\mathcal{D}_1|}{|\mathcal{D}|} H(\mathcal{D}_1) - \frac{|\mathcal{D}_2|}{|\mathcal{D}|} H(\mathcal{D}_2)$$

- In information theory, entropy is the expected number of bits needed to encode a randomly drawn value of X.
- Larger entropy, less information. That's why we call it IG.

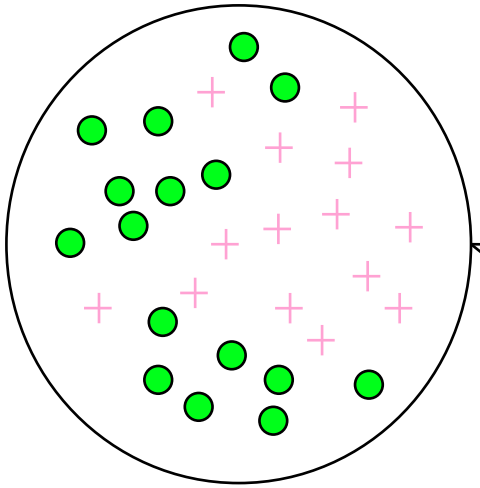
https://en.wikipedia.org/wiki/Entropy_(information_theory)



Claude Shannon

Calculating Information Gain (IG)

Entire population (30 instances)

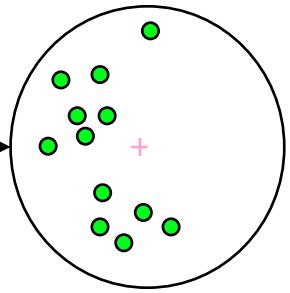


Parent entropy:

$$-\left(\frac{14}{30}\log\frac{14}{30}\right)-\left(\frac{16}{30}\log\frac{16}{30}\right)=0.996$$

Child entropy:

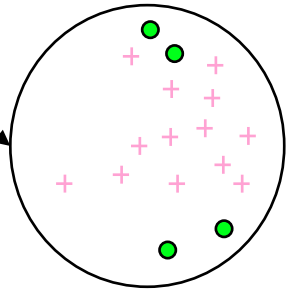
$$-\left(\frac{1}{13}\log\frac{1}{13}\right)-\left(\frac{12}{13}\log\frac{12}{13}\right)=0.391$$



13 instances

Child entropy:

$$-\left(\frac{13}{17}\log\frac{13}{17}\right)-\left(\frac{4}{17}\log\frac{4}{17}\right)=0.787$$

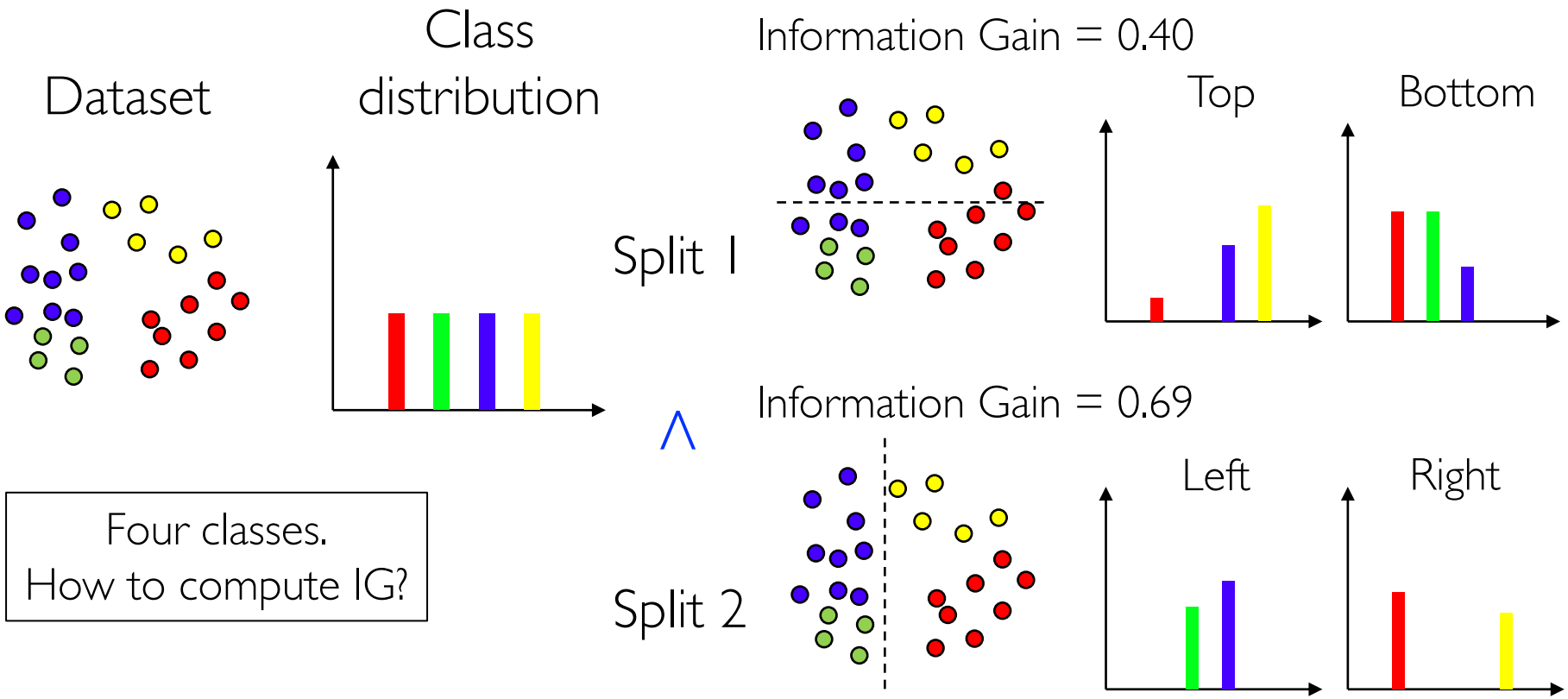


17 instances

Information Gain (IG):

$$0.996-\frac{13}{30}\cdot 0.391-\frac{17}{30}\cdot 0.787=0.38$$

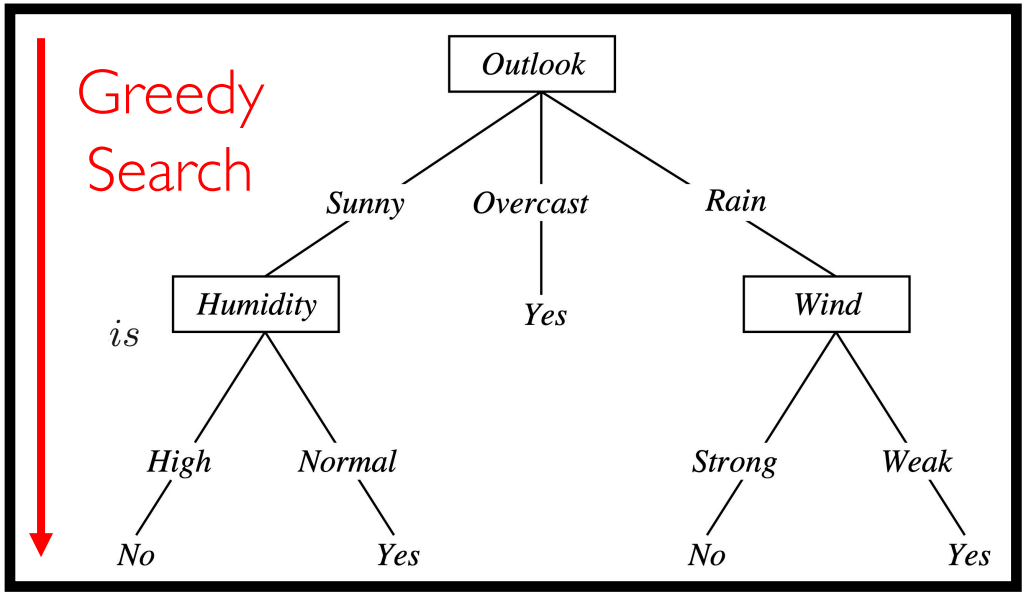
Split Search



- Compute IG for all splits induced by every feature.
  - If IGs of all features are small (e.g. $< \epsilon$): stop.
  - Else: find the feature that maximizes the Information Gain (IG).

ID3 Algorithm

Ross Quinlan



- Start
  - Create the root node.
  - Assign all examples to root.
- Main Loop
  1. $A \leftarrow$ the best decision attribute for next node.
  2. For each value of A, create a new descendant of node.
  3. Sort training examples to leaf nodes.
  4. If training examples well classified, then STOP; else iterate over new leaf nodes.

Time Complexity (n #examples, d #features): $O(dn \cdot \text{depth})$

ID3 Algorithm

Class label

ID3(*Examples*, *Target_attribute*, *Attributes*)

- create a *Root* node for the tree; assign all *Examples* to *Root*;
- if all *Examples* are positive, return the single-node tree *Root*, with label=+;
- if all *Examples* are negative, return the single-node tree *Root*, with label=−;
- if *Attributes* is empty, return the single-node tree *Root*, with label = the most common value of *Target_attribute* in *Examples*;
- otherwise // **Main loop:**

  $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*;
  the decision attribute for *Root* $\leftarrow A$;
  for each possible value v_i of A

  add a new tree branch below *Root*, corresponding to the test $A = v_i$;
  let $Examples_{v_i}$ be the subset of *Examples* that have the value v_i for A;
  if $Examples_{v_i}$ is empty

  below this new branch add a leaf node with label = the most common value of *Target_attribute* in *Examples*;

  else

  below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes*\{A}$);$

- return *Root*;

Stop Criteria

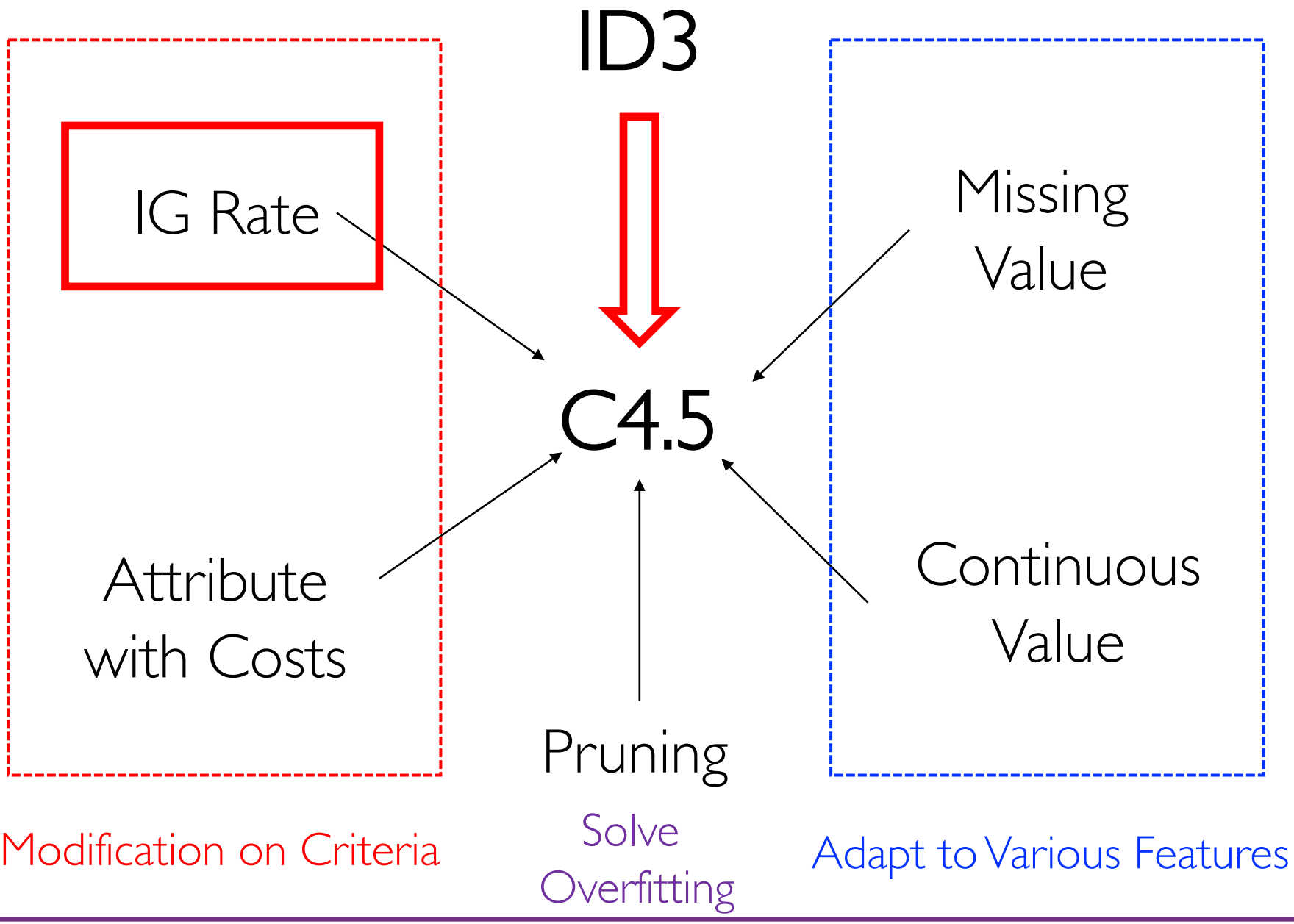
$O(dn)$ in each layer

depth $\min(d, \log n)$

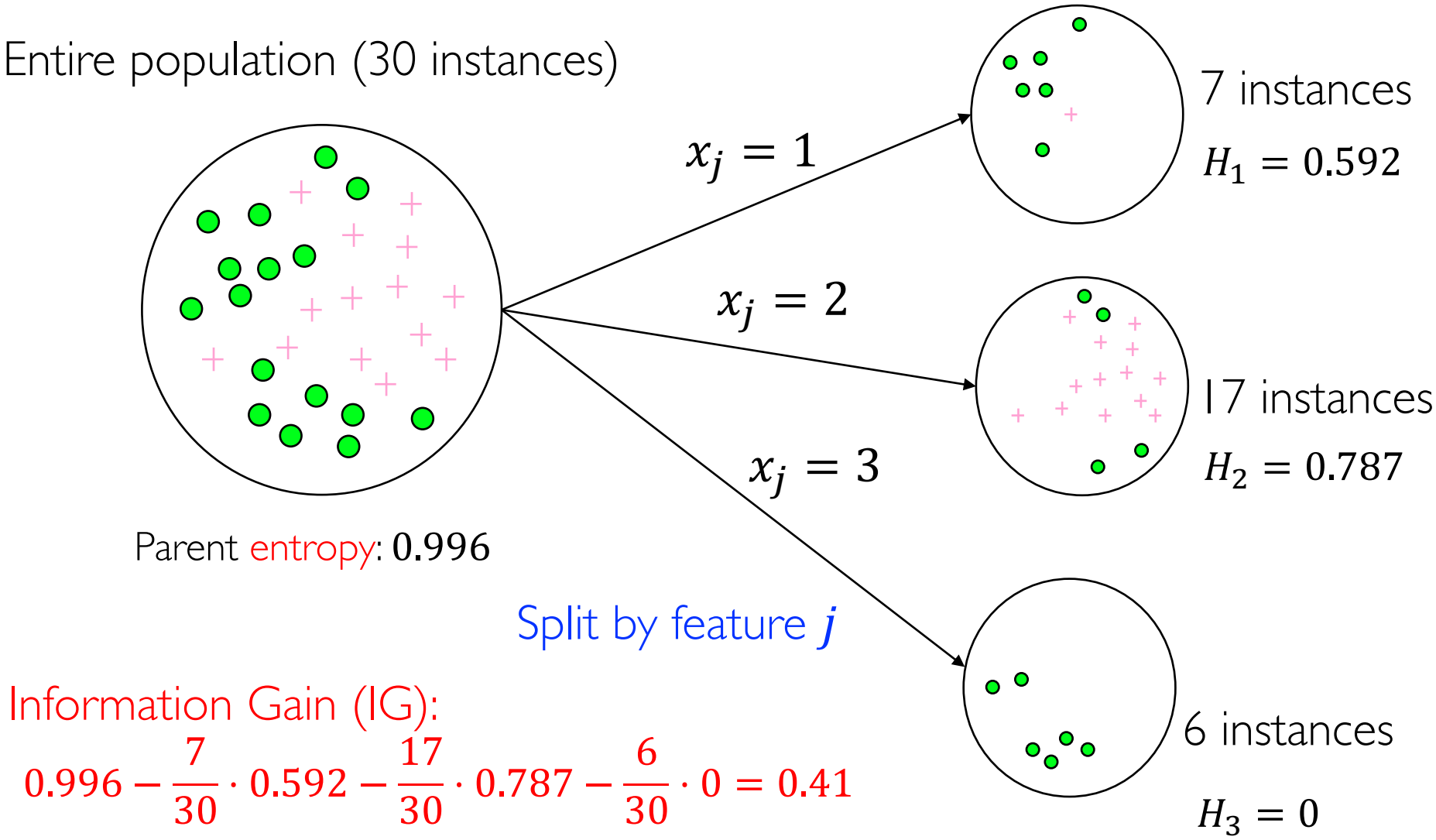
* The *best* attribute is the one with the highest *information gain*.

Outline

- **Decision Tree**
  - ID3 Algorithm
  - **C4.5 Algorithm**
  - Classification and Regression Tree (CART)
  - Generalization Bound
- Random Forest
  - Bagging
  - Breiman Algorithm


ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Modification on Criteria
Solve Overfitting
Adapt to Various Features

Information Gain for Multivalued Features



Information Gain (IG):

$$0.996 - \frac{7}{30} \cdot 0.592 - \frac{17}{30} \cdot 0.787 - \frac{6}{30} \cdot 0 = 0.41$$

Information Gain for Multivalued Features

- Consider **Day** (primary key of the table) as a feature, what is the IG?

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

- There are only one instance in one node, so the entropy is zero.
  - Max Information Gain, but useless for classification. Why?
- Information Gain is highly biased to multivalued features.
  - This will cover up useful features with fewer numbers of values.

Gain Ratio

- By penalizing multivalued rate, IG is improved to Gain Ratio (GR):

$$\text{GR} = \frac{\text{Information Gain}}{\text{multivalued rate}}$$

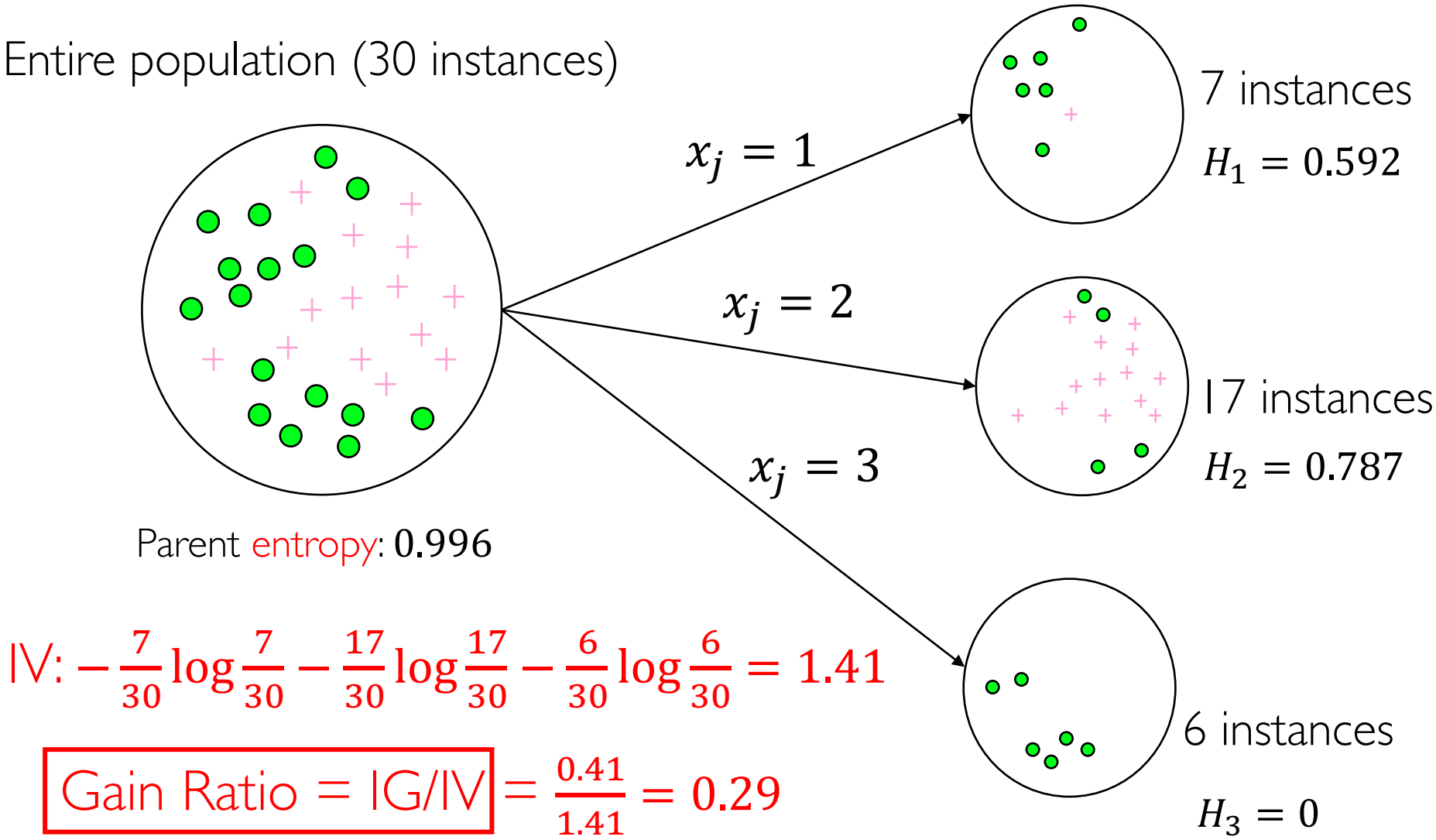
- C4.5 Algorithm measures multivalued rate by Intrinsic Value (IV):

$$\text{IV}(f) = -\sum_{i=1}^{|V|} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \log \frac{|\mathcal{D}_i|}{|\mathcal{D}|}$$

  - When $\mathcal{D}$ is split by the feature f to $\mathcal{D}_i$ if feature value is i.
  - Which is the entropy of the value probability of the feature f.

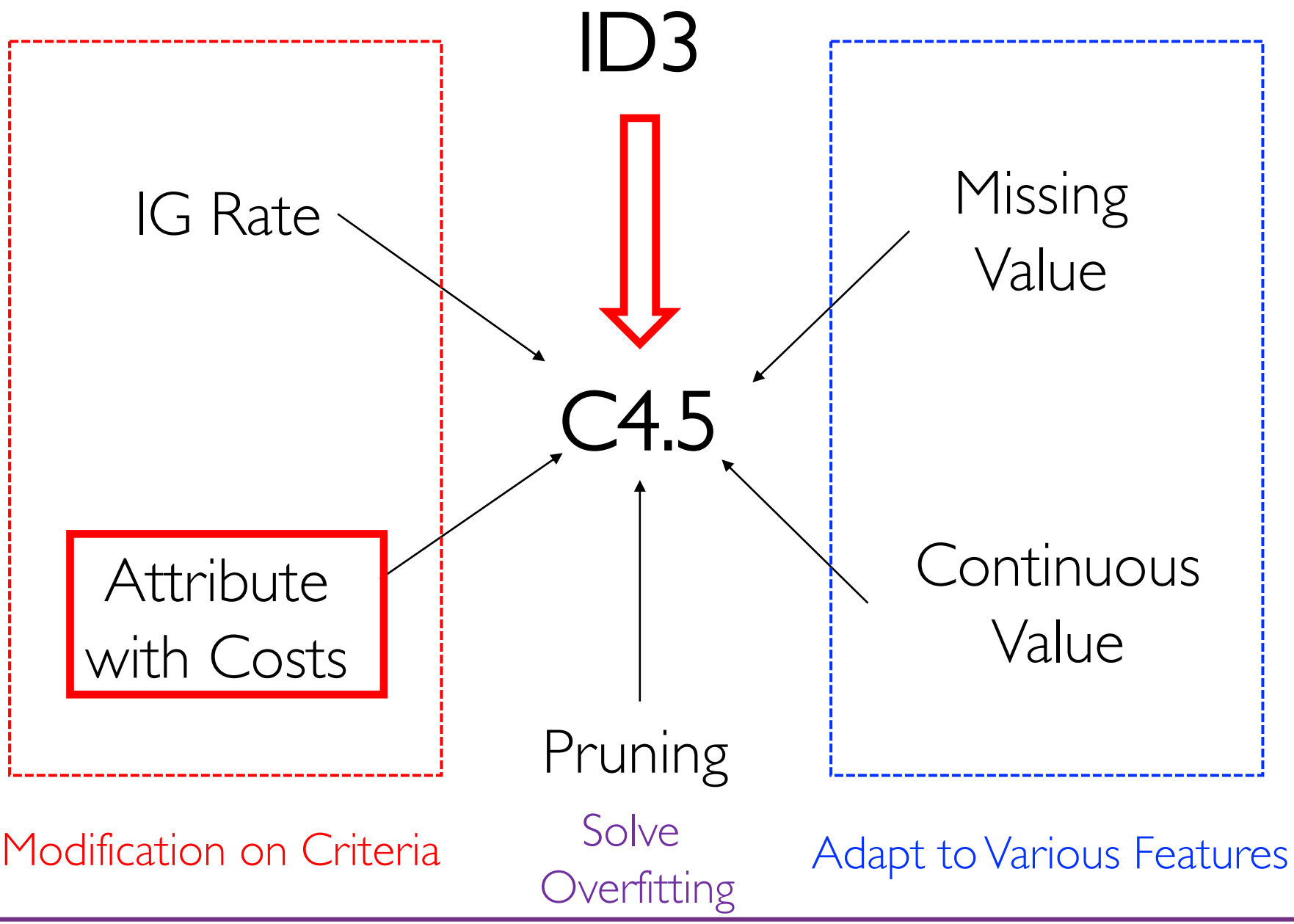
- $\text{GR(Day)} = \frac{\text{IG(Day)}}{\log|\mathcal{D}|}$, it is no longer the largest one.

Information Gain for Multivalued Features



$$\text{IV: } -\frac{7}{30}\log\frac{7}{30} - \frac{17}{30}\log\frac{17}{30} - \frac{6}{30}\log\frac{6}{30} = 1.41$$

$$\text{Gain Ratio} = \text{IG/IV} = \frac{0.41}{1.41} = 0.29$$

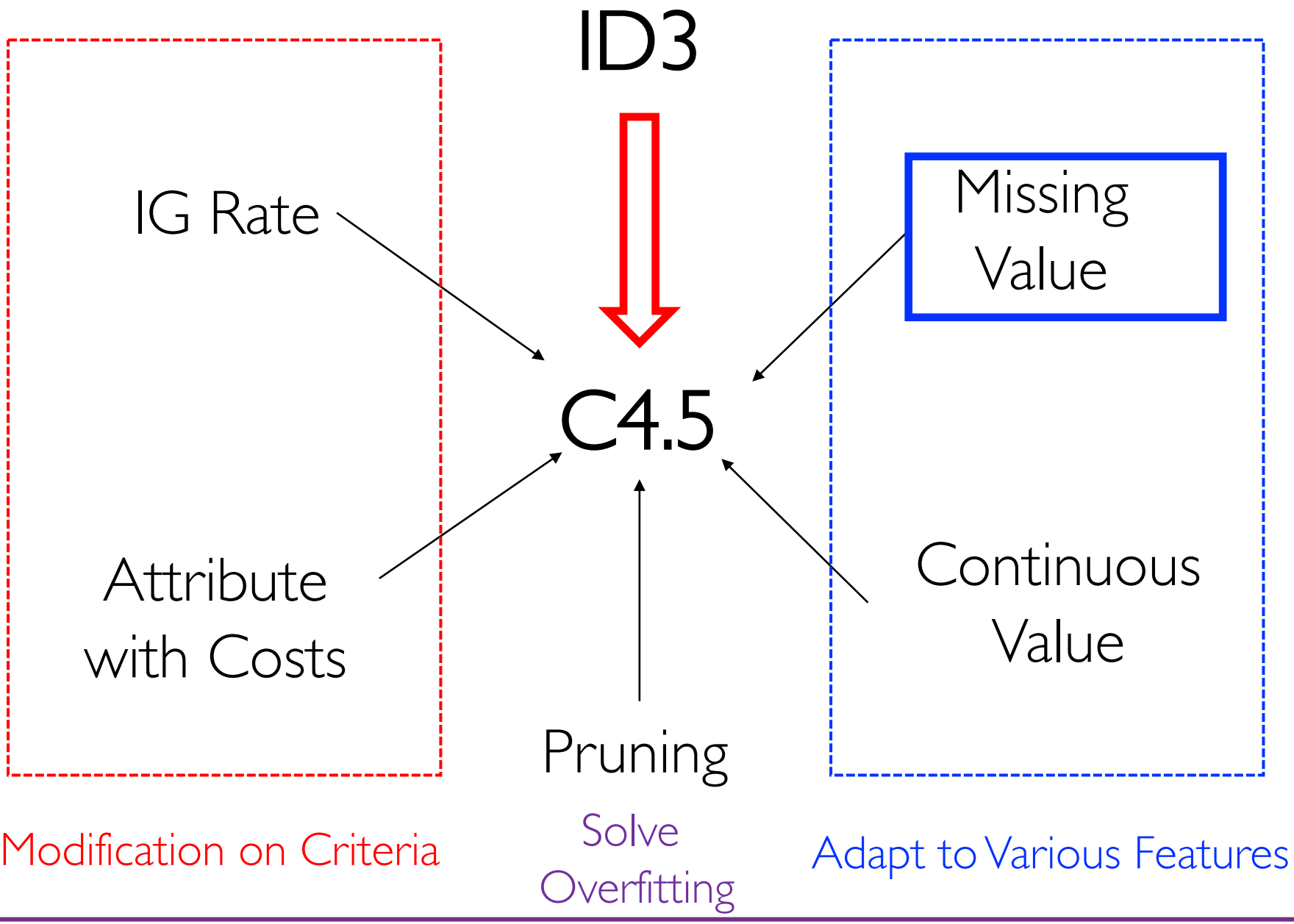

ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Modification on Criteria
Solve Overfitting
Adapt to Various Features

Attributes with Different Costs

- To collect different attributes may incur different costs:
  - In medical diagnosis, **Age** is very cheap; But **MRI** result costs $500.
  - In robotics, measuring the **width of the space** needs ~20s
- Can we avoid non-necessary cost in a decision tree?
  - Penalize on the Cost of this feature.
- **C4.5** introduces a new criterion that could penalize the cost:

$$\frac{(\text{Gain Ratio})^2}{\text{Cost}}$$

- How to define Cost?
  - We still need to define it by hand (feature engineering).


ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Modification on Criteria
Solve Overfitting
Adapt to Various Features

Missing Values

- What if there are missing values in the dataset?

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	—	Hot	High	Weak	No
D2	Sunny	—	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	—	Yes
D5	Rain	Cool	Normal	Weak	Yes

- Delete features or samples with the missing value:
  - May lose useful information.
- C4.5: Improve the criteria (IG) to adapt missing values in training.
  - Reweight samples with missing values when building decision tree.
- During test: supplement missing values with the most common one.

We will discuss this in detail

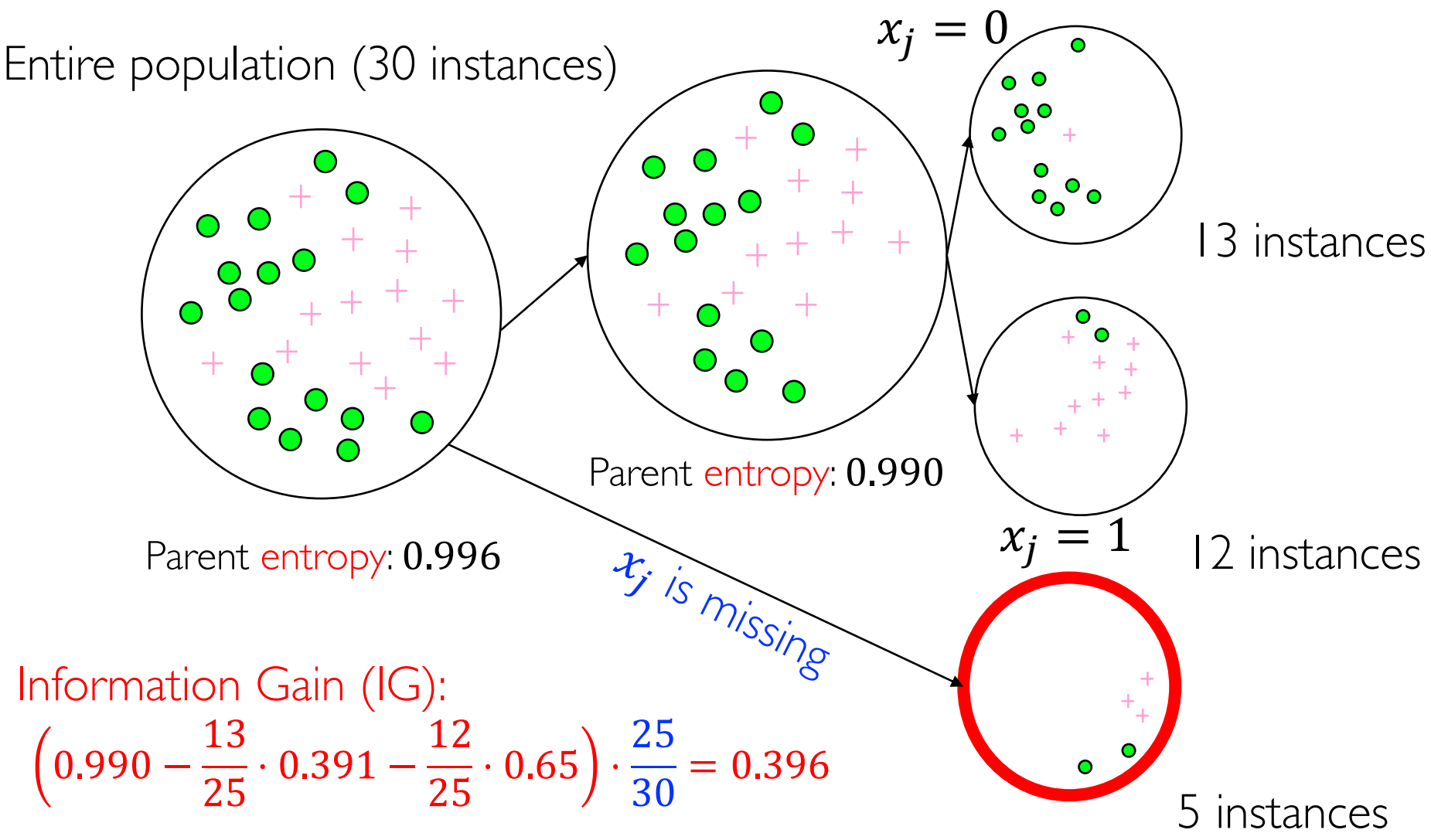
Information Gain with Missing Values

- In the ID3 algorithm, there are two steps:
  - Computing Information Gain for every feature.
    - How to define IG for features with missing values?
  - Assign samples on this node to its child nodes split by best feature.
    - How to assign samples with missing value on this feature?

Two problems we need to solve

- **Step 1:** When computing IG, for a feature with missing values:
  - Compute the ratio of samples with missing value: $|\bar{\mathcal{D}}|/|\mathcal{D}| = \rho$.
  - Compute the IG on $\mathcal{D}\backslash\bar{\mathcal{D}}$ (all samples with values):
  - The IG for this feature is computed by: $(1-\rho)\cdot$ IG.
    - Then compute Gain Ratio based on this new Information Gain.

Information Gain with Missing Values

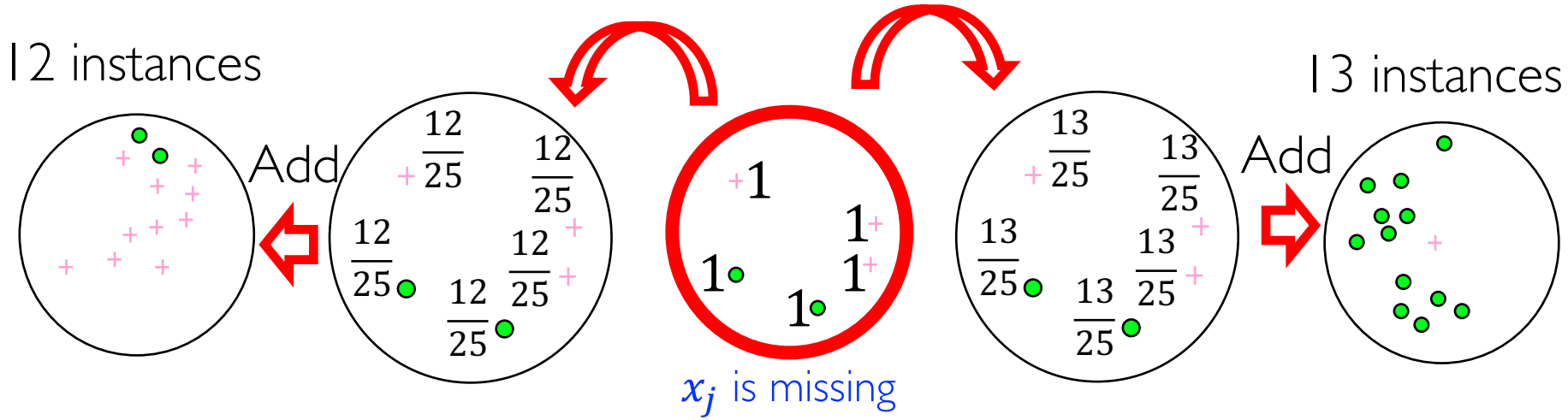


Information Gain (IG):

$$\left(0.990 - \frac{13}{25} \cdot 0.391 - \frac{12}{25} \cdot 0.65\right) \cdot \frac{25}{30} = 0.396$$

Assign Samples with Missing Values

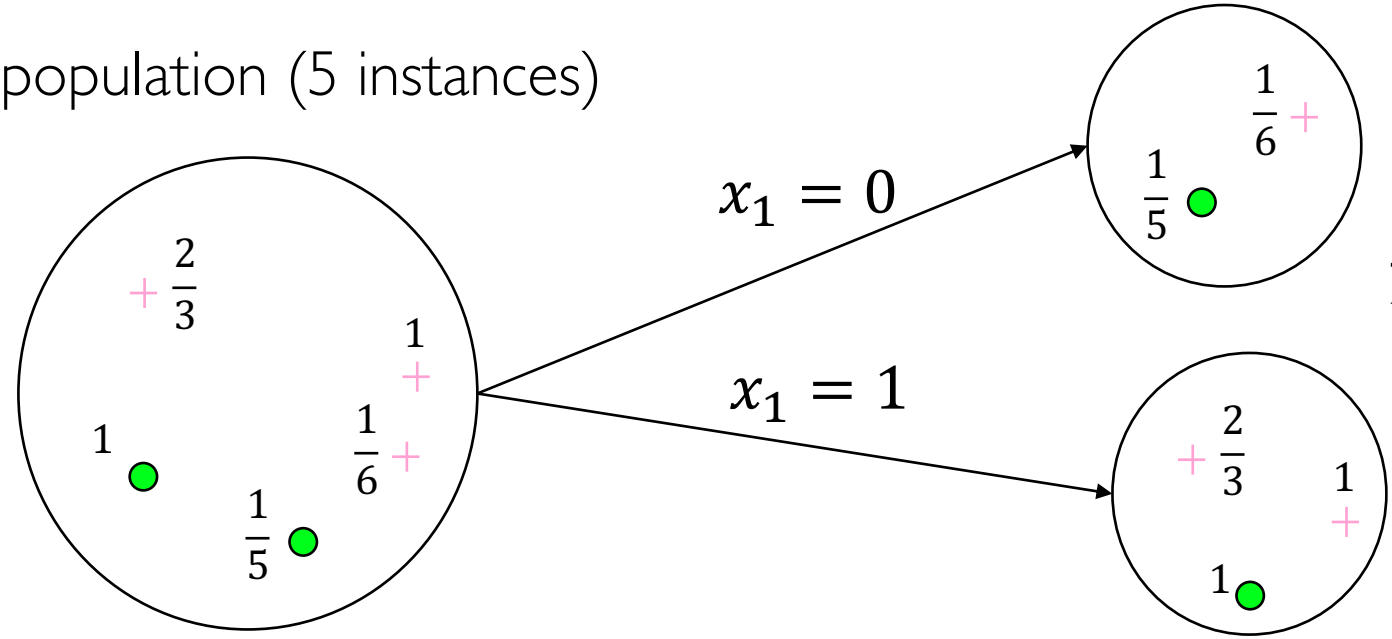
- **Step 2:** How to assign samples with missing value on this feature?
- Assign them to all nodes, with weights proportional to node sizes.



- Each sample is weighted as 1 at the root node.
  - For each split on the missing value, multiply the weight.
- Computing Weighted Information Gain on the child node.

Information Gain on Weighted Samples

Entire population (5 instances)



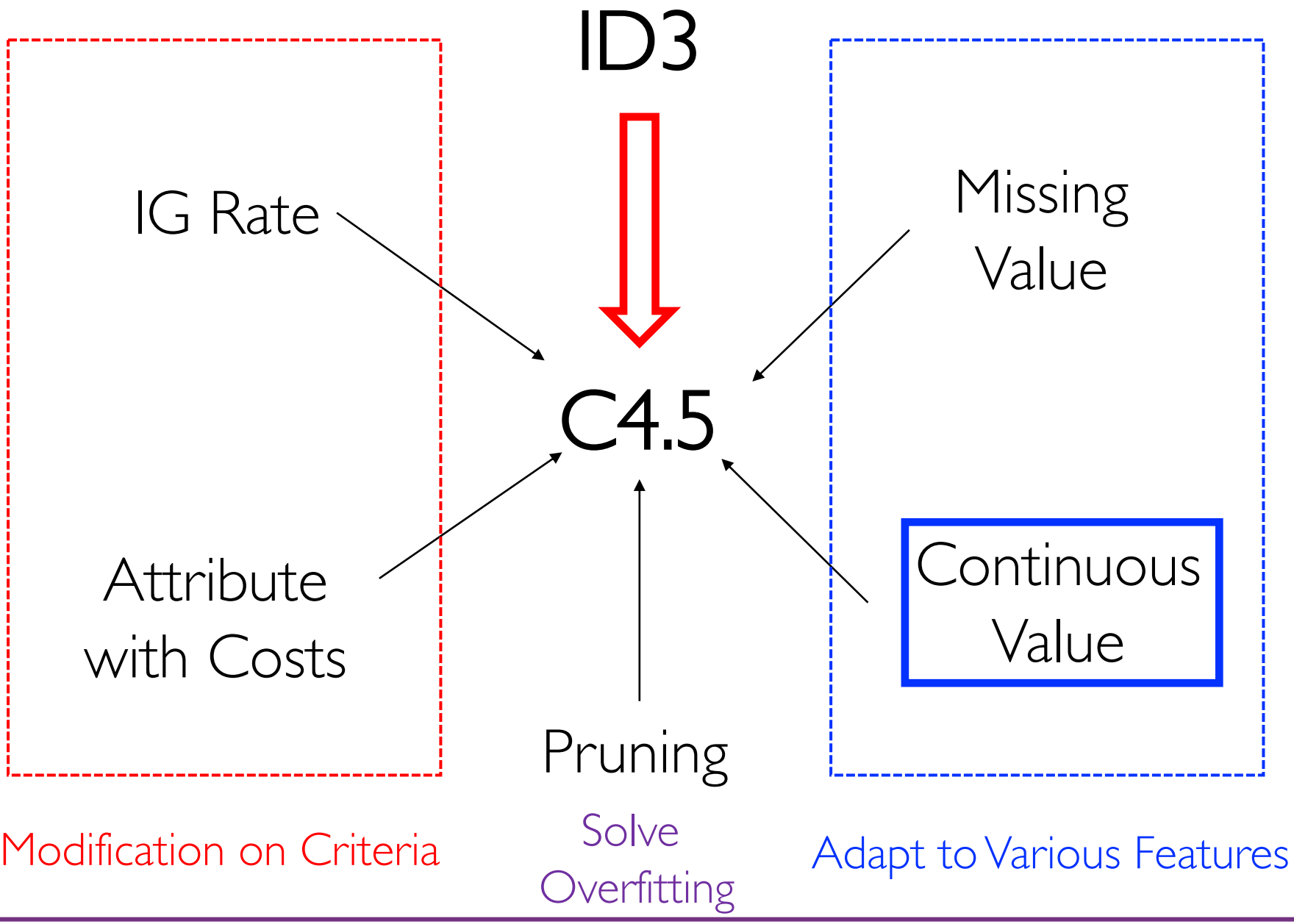
Parent entropy:

$$-\left(\frac{\frac{2}{3}+\frac{1}{6}+1}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}}\right)\log\left(\frac{\frac{2}{3}+\frac{1}{6}+1}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}}\right)-\left(\frac{\frac{1}{5}+1}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}}\right)\log\left(\frac{\frac{1}{5}+1}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}}\right)$$

Information Gain:

$$H(\mathcal{D}_1 \cup \mathcal{D}_2) - \frac{\frac{1}{5}+\frac{1}{6}}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}} H(\mathcal{D}_1) - \frac{1+1+\frac{2}{3}}{1+1+\frac{1}{5}+\frac{2}{3}+\frac{1}{6}} H(\mathcal{D}_2)$$

Let's leave it to computer.


ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Modification on Criteria
Solve Overfitting
Adapt to Various Features

Continuous Variables

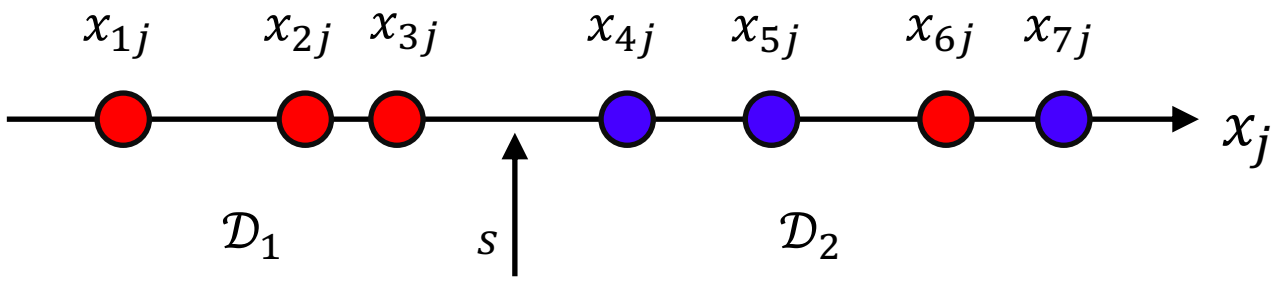
- What if there are continuous values in the dataset?
- Consider previous Tennis example with a temperature feature (°F):

Temperature:	**40**	**48**	**60**	**72**	**80**	**90**
EnjoyTennis:	**No**	**No**	**Yes**	**Yes**	**Yes**	**No**

- Add threshold to split:

$$\{\boldsymbol{x}|x_j \leq s\} \cup \{\boldsymbol{x}|x_j > s\}$$

Red: +
Blue: -



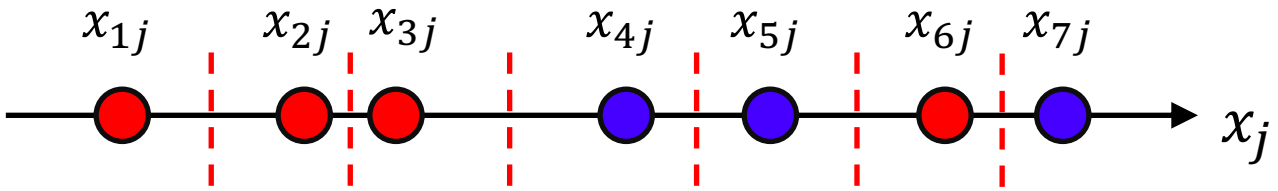
Finding the Split Point

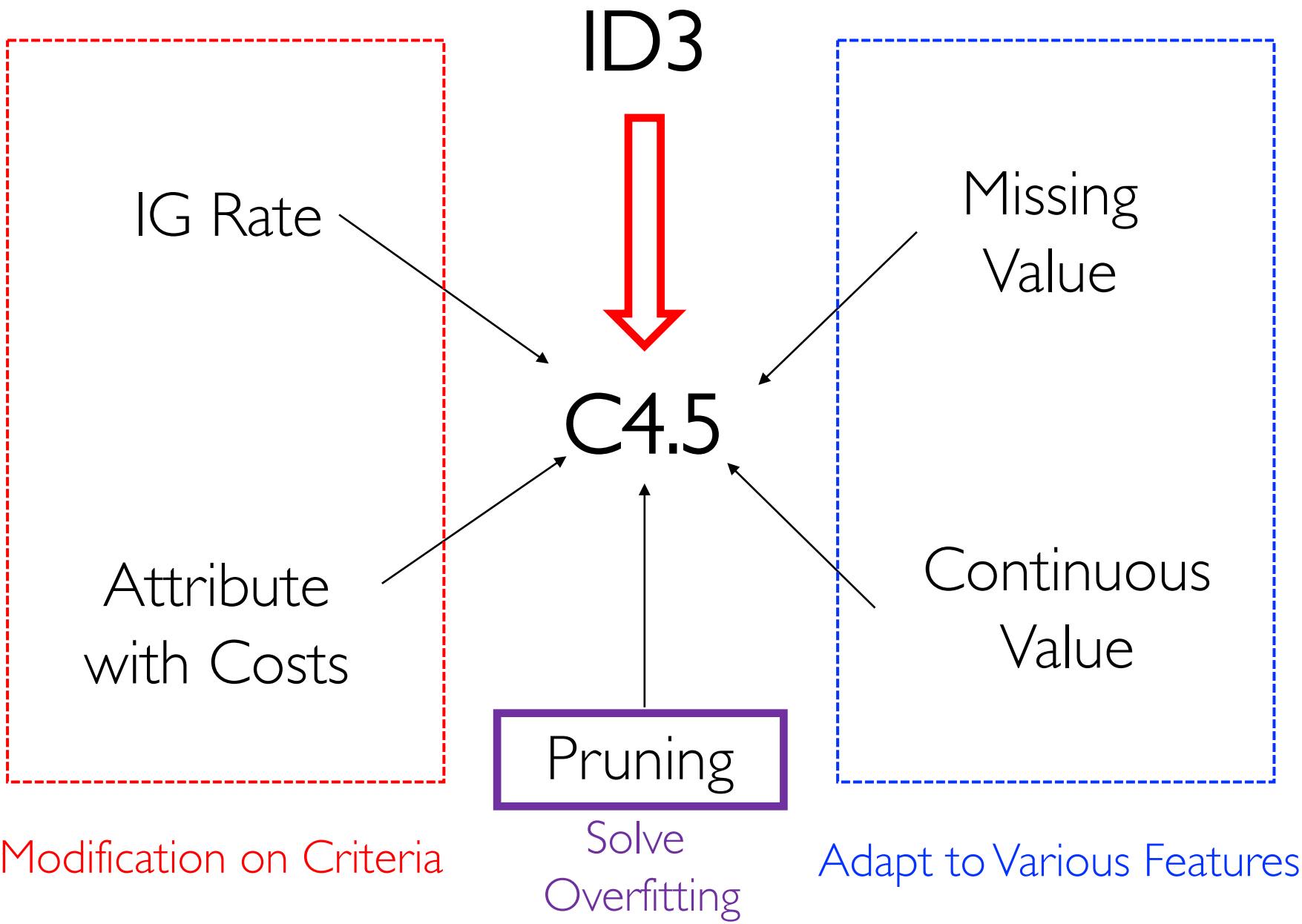
- Consider splitting on the jth feature x_j.
- If $x_{1j}, \dots, x_{nj}$ are the sorted values of the jth feature for n instances.
  - We only need to check split points between adjacent values
  - Traditionally take split points halfway between adjacent values:

$$s_j \in \left\{ \frac{1}{2}\left(x_{rj} + x_{(r+1)j}\right) \middle| \, r = 1, \dots, n-1 \right\}$$

- Check performance of $n-1$ splits and find the one with highest IG.

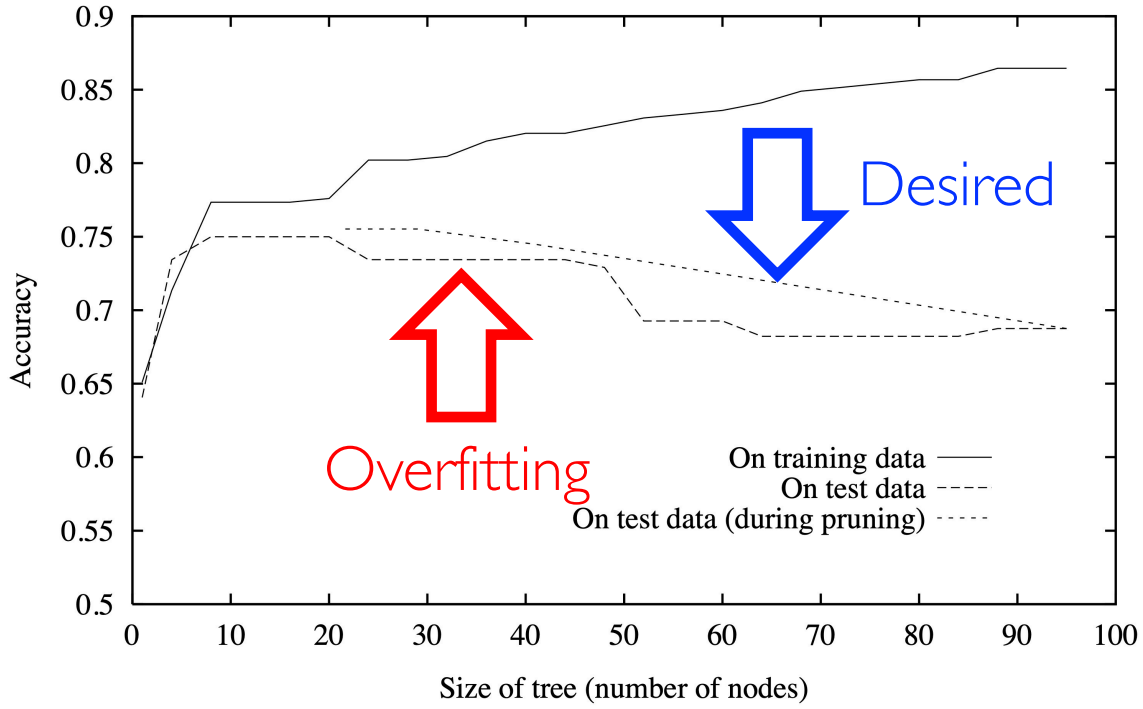
Red: +
Blue: -




ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Modification on Criteria
Solve Overfitting
Adapt to Various Features

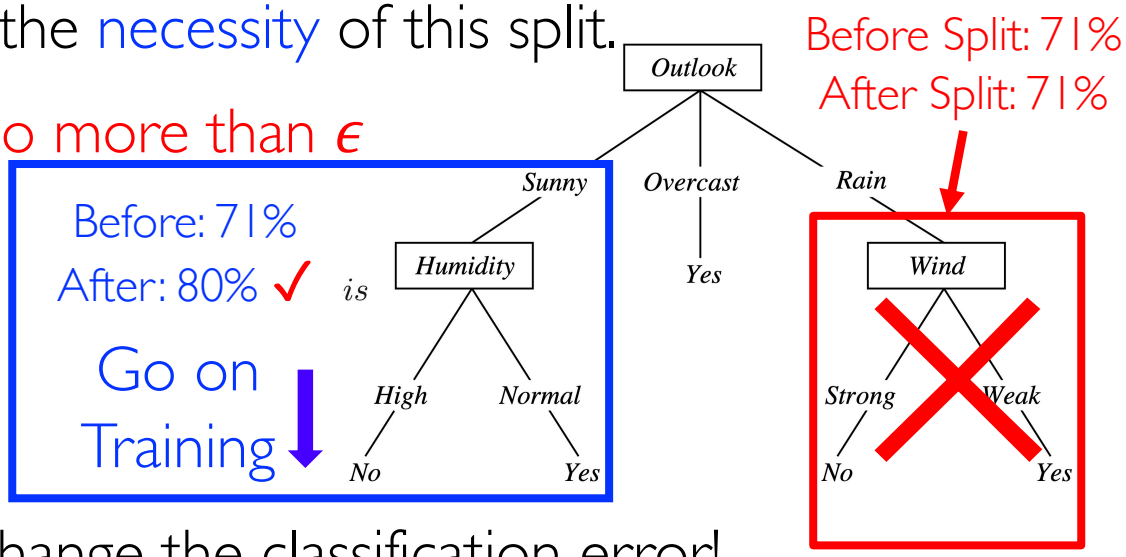
Overfitting in Decision Tree

- Decision tree has the ability to fit all data:
  - Except noisy samples with same features and different labels.
- Overfitting also appears in decision tree:



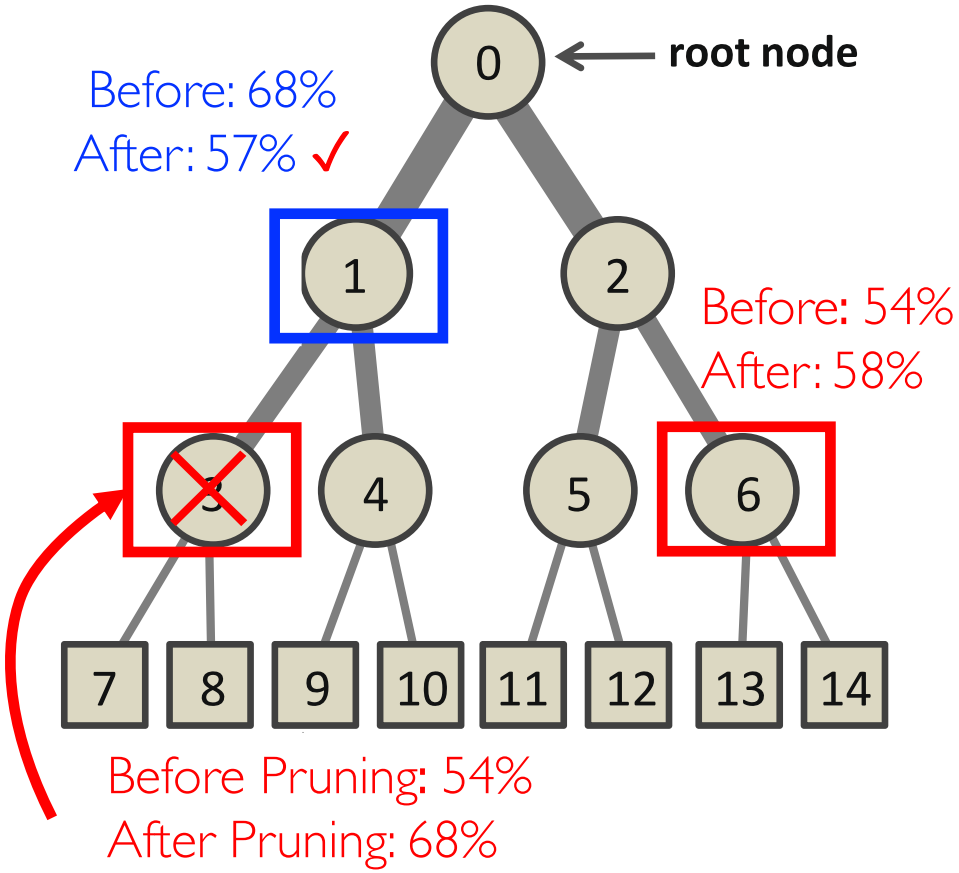
Pre-Pruning

- Tree Complexity is measured by numbers of layers and branches.
  - To prevent overfitting: Control number of layers – Pruning.
- A simplest way is Pre-Pruning:
- **During Training:** for every split, use the change of accuracy on validation set to measure the necessity of this split.
  - If the accuracy grows no more than ϵ or even decreases:
    - Stop the split.
- May cause underfitting!
  - A good split may not change the classification error!



Post-Pruning

- Pre-Pruning may lead to underfitting.
- Safe idea is Post-Pruning:
  - Pruning the tree after training
- From leaf node to root node:
  - Use the change of accuracy on validation set to measure the necessity of the pruning on this node.
  - Remove node most improving validation set accuracy.



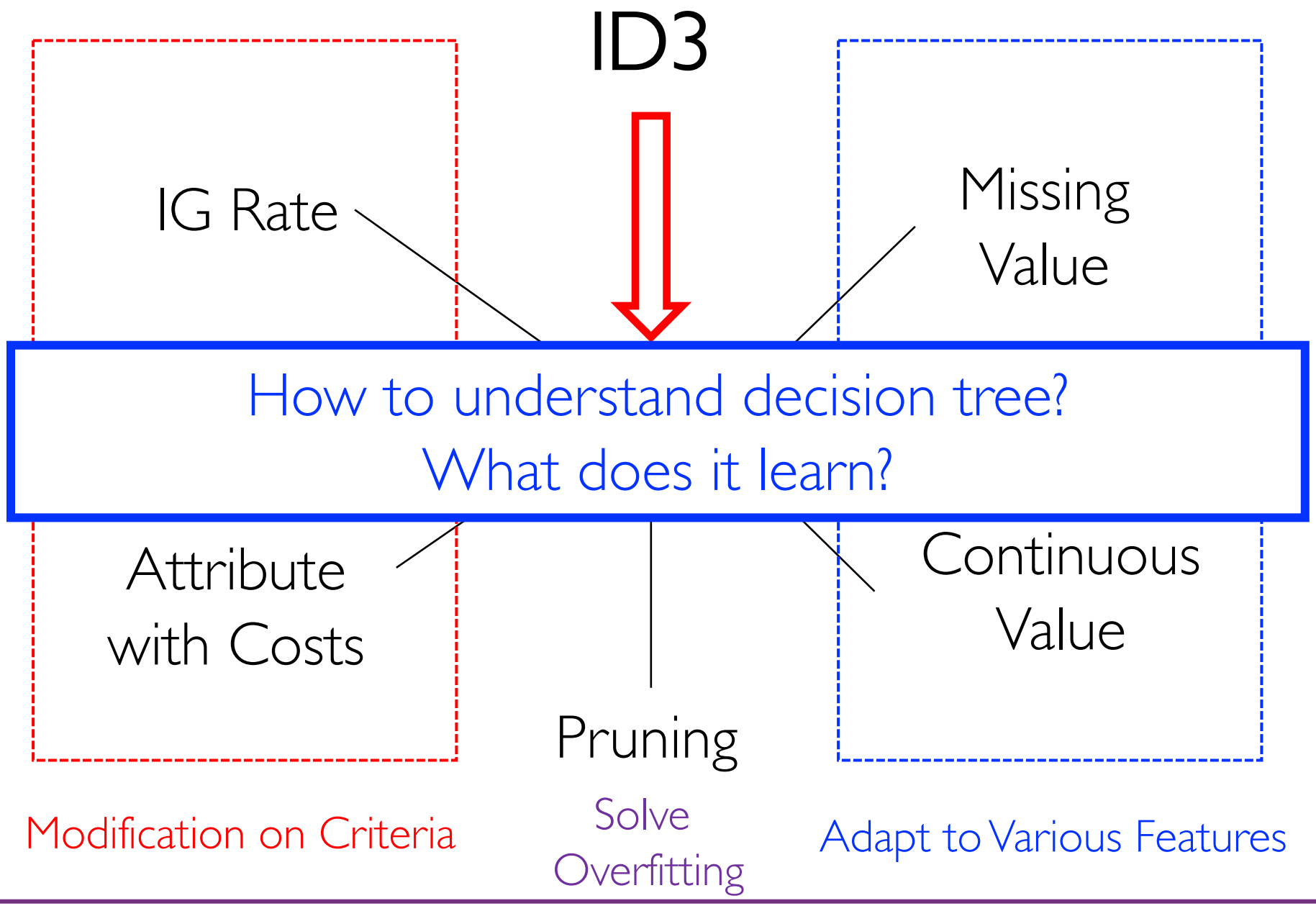
Tree Complexity Regularization

- The cost complexity criterion with parameter α for tree T is

$$C_\alpha(T) = \hat{\mathcal{E}}(T) + \alpha|T| = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha|T|$$

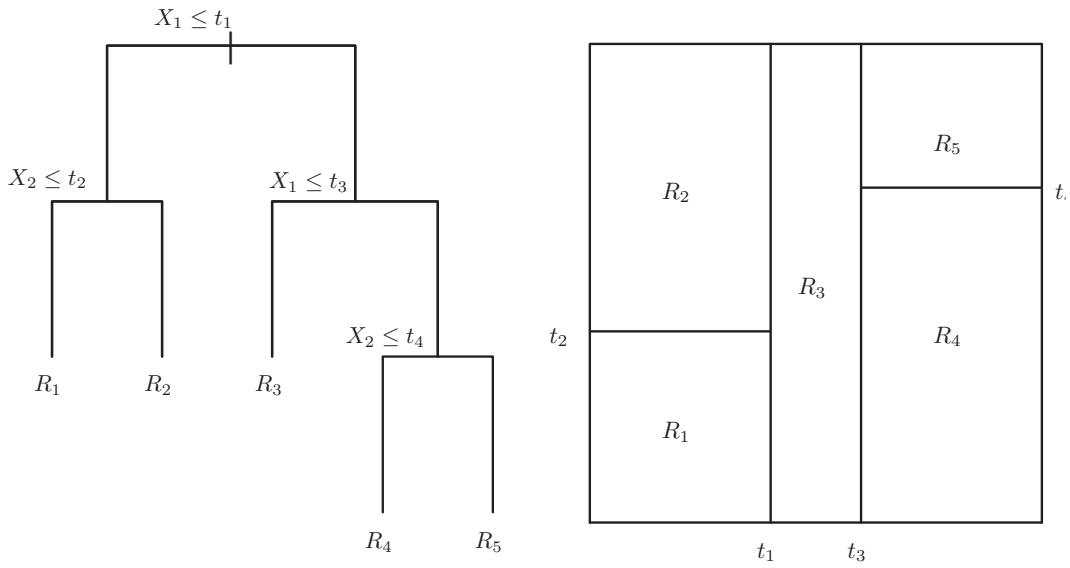
$$= -\sum_{t=1}^{|T|}\sum_{k=1}^{K} N_{tk} \log \frac{N_{tk}}{N_t} + \alpha|T|$$

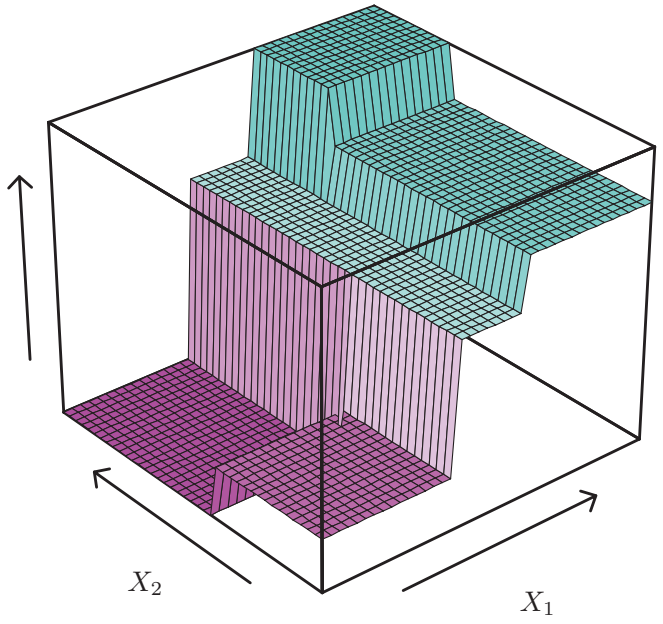
  - $|T|$: #leaf nodes, N_{tk} is the #examples of class k in leaf node t
  - $\hat{\mathcal{E}}(T)$ is the empirical error of the tree on training data.
- Cost Complexity Pruning: (Given tree T_0 and parameter α)
  - Compute the empirical entropy $H_t(T)$ of each node
  - Recursively shrink from leaf nodes to internal nodes
    - If $C_\alpha(T_B) \leq C_\alpha(T_A)$: prune leaf A and use parent B as new leaf


ID3
IG Rate
Missing Value
How to understand decision tree?
What does it learn?
Attribute with Costs
Pruning
Continuous Value
Modification on Criteria
Solve Overfitting
Adapt to Various Features

Understanding Decision Tree

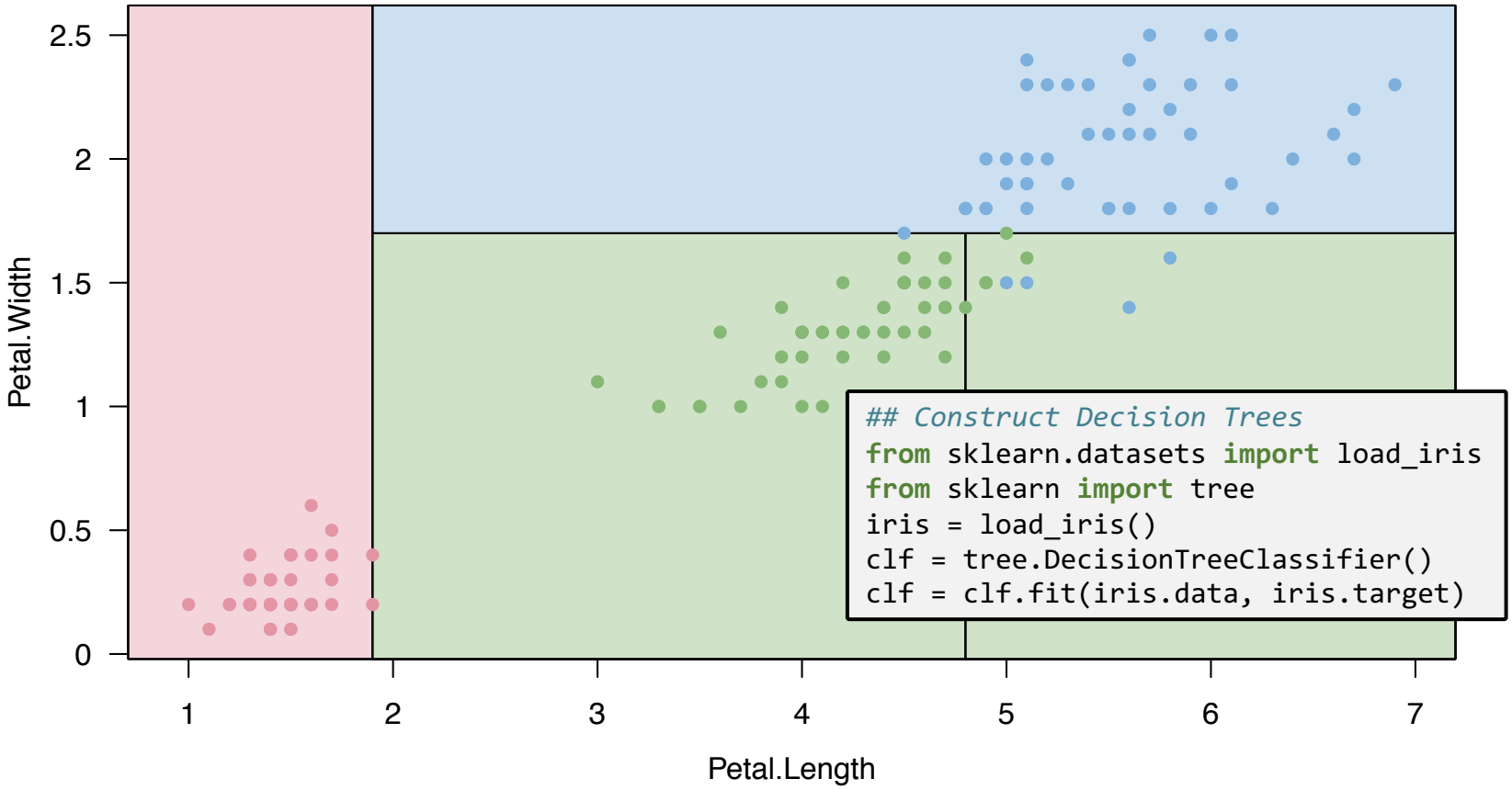
- **Hypothesis space** of decision tree:
  - Decision trees divide the feature space into axis-parallel rectangles.
  - Each rectangular region is labeled with a specific label.
  - Curse of dimensionality.





Understanding Decision Tree

- Decision Tree on the Iris Data. What phenomena can you find?



Nonlinear Feature Extraction

- Suppose decision tree T gives a partition with regions $R_1, \ldots, R_m$.
- Then the decision tree can be written as:

c_i is the label of R_i

$$h(\boldsymbol{x}) = \sum_{i=1}^{m} c_i \mathbf{1}\{\boldsymbol{x} \in R_i\}$$

- Each region can be viewed as extracting a feature for every sample:

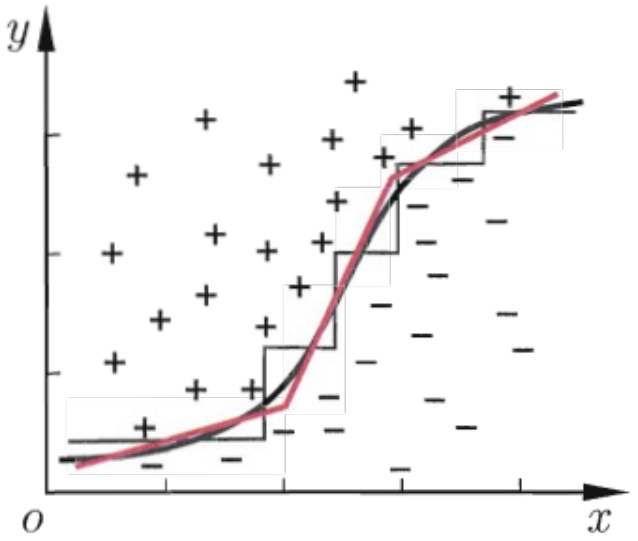
$$\boldsymbol{x} \to \boldsymbol{\Phi}(\boldsymbol{x}) = (\mathbf{1}\{\boldsymbol{x} \in R_1\}, .., \mathbf{1}\{\boldsymbol{x} \in R_m\})$$

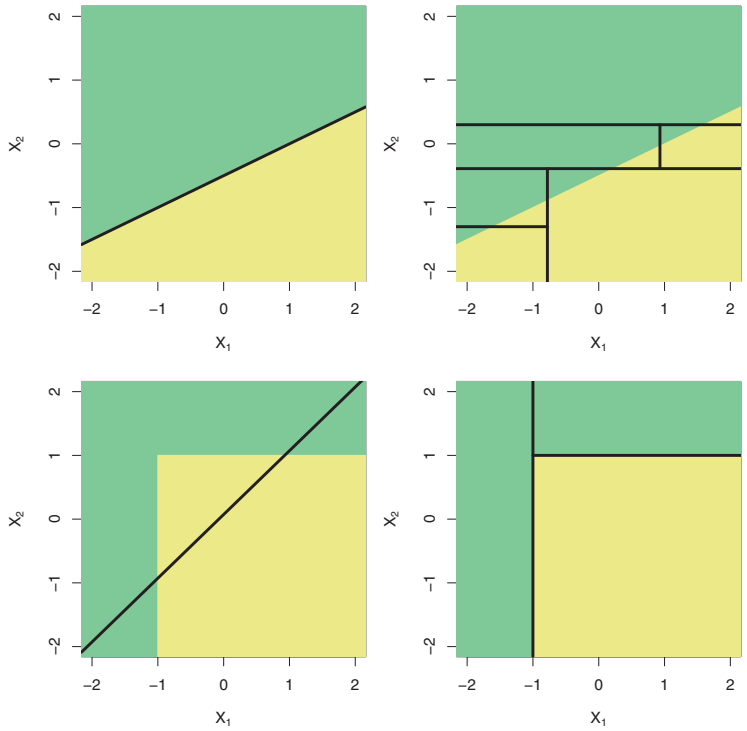
- $h(\boldsymbol{x}) = \boldsymbol{c} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$ is a linear classifier on the nonlinear features $\boldsymbol{\Phi}$.

- These nonlinear feature maps are non-differential functions.
- Is it the ability of feature learning? Can it be applied by layering trees?

Multivariate Decision Tree

- Decision Tree is not good at settings that the separating hyperplane is not parallel to the axis.



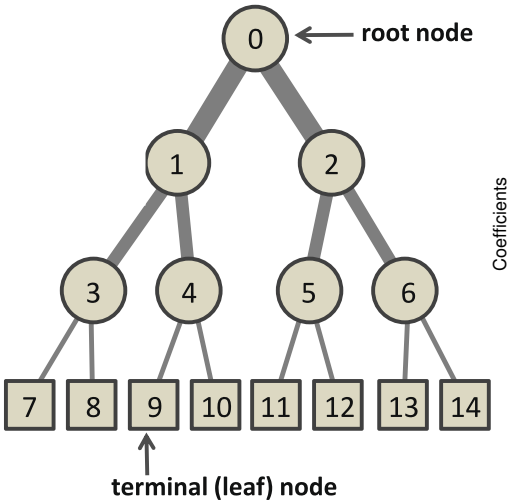


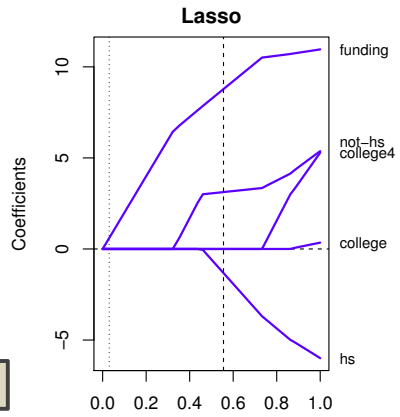
- Choose multiple features and train a linear classifier at each node!
  - An algorithm: Oblique Classifier 1 – not commonly used. Why?

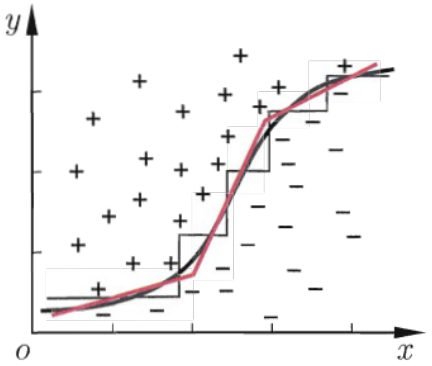
Expressivity vs. Interpretability

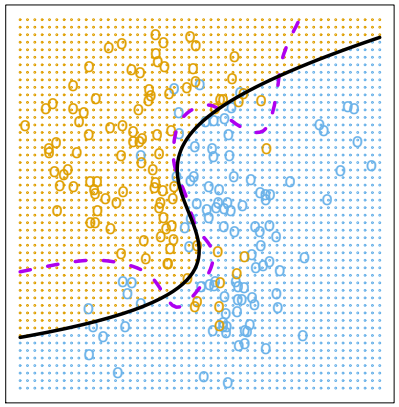
- Multivariate decision tree: good expressivity but weak interpretability.
- Are Expressivity and Interpretability in conflict?

Stronger expressivity?









Weaker interpretability

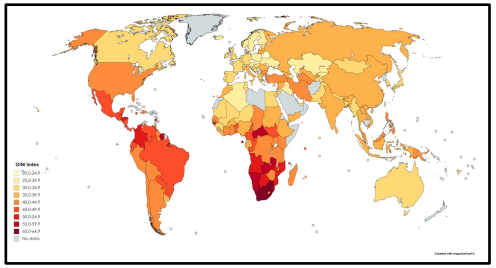
Outline

- **Decision Tree**
  - ID3 Algorithm
  - C4.5 Algorithm
  - **Classification and Regression Tree (CART)**
  - Generalization Bound
- Random Forest
  - Bagging
  - Breiman Algorithm

Classification Tree: Gini Index

- Gini index (uncertainty of samples in $\mathcal{D}$):

$$\text{Gini}(\mathcal{D}) = 1 - \sum_{k=1}^{K} \left(\frac{|\mathcal{C}_k|}{|\mathcal{D}|} \right)^2$$



- If $\mathcal{D}$ is split into $\mathcal{D}_1$ and $\mathcal{D}_2$ by whether attribute $A(x) = a$:

$$\mathcal{D}_1 = \{(x, y) \in \mathcal{D} : A(x) = a\}, \mathcal{D}_2 = \mathcal{D} - \mathcal{D}_1$$

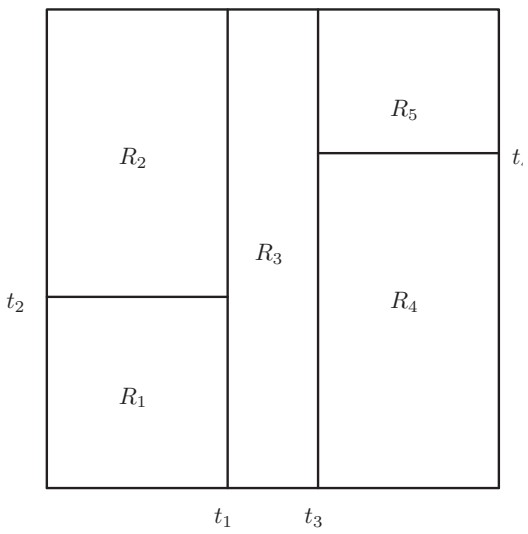
- Then under the condition of attribute A, the Gini index of $\mathcal{D}$ is

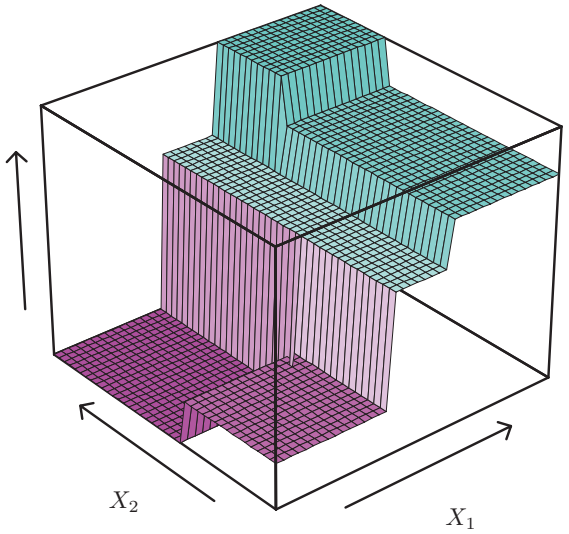
$$\text{Gini}(\mathcal{D}, A) = \frac{|\mathcal{D}_1|}{|\mathcal{D}|} \text{Gini}(\mathcal{D}_1) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|} \text{Gini}(\mathcal{D}_2)$$

- Gini index $\text{Gini}(\mathcal{D}, A)$ is used to evaluate node split in CART, which is a binary tree where each node is split into $A(x) = a$ and $A(x) \neq a$

Regression Tree

- Can we use decision trees in regression task?
- Recall the visualization of decision tree prediction landscape:





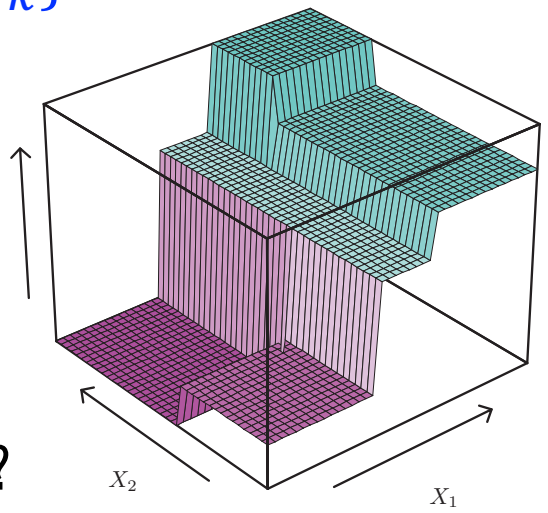
Discrete Label
⇒
Continuous Target

- We assign a class label for each of the regions in classification.
- In regression: for each region, assign continuous target (response).

Regression Tree

- Given the partition $\{R_1, \ldots, R_m\}$, the final prediction is

$$\hat{y}(\boldsymbol{x}) = h(\boldsymbol{x}) = \sum_{k=1}^{m} v_k \mathbf{1}\{\boldsymbol{x} \in R_k\}$$



- How to choose $v_1, \ldots, v_m$ (of linear regressor)?
- For L2 loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$, the best is

$$\hat{v}_k = \text{avg}(y_i | \boldsymbol{x}_i \in R_k)$$

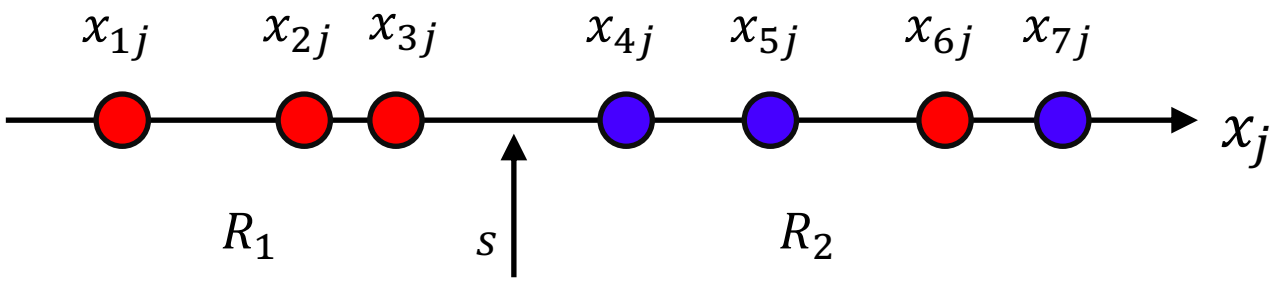
Average. Why?

- In classification tree, we find node of high purity.
- In regression we find node of low L2 loss to its average target value.

Continuous Variables

- Add threshold to split:

$$\{\boldsymbol{x}|x_j \leq s\} \cup \{\boldsymbol{x}|x_j > s\}$$



- In classification, we try to find best feature and split to ensure best IG.
- In regression, we want to find feature j and splitting threshold s_j:
  - To minimize L2 loss on left node and right node of the split:

$$(\text{avg}(y_i|\boldsymbol{x}_i \in R) - y)^2$$

Node Splitting

- For each splitting variable j and splitting point s
  - Assume that the region R is split by j and s to R_1 and R_2.

$$\hat{v}_1(j,s) = \text{avg}(y_i | \boldsymbol{x}_i \in R_1(j,s))$$
$$\hat{v}_2(j,s) = \text{avg}(y_i | \boldsymbol{x}_i \in R_2(j,s))$$

- Find j, s that minimize the L2 loss

$$\ell(j,s) = \underbrace{\sum_{i:\boldsymbol{x}_i \in R_1(j,s)} \left(y_i - \hat{v}_1(j,s)\right)^2}_{\text{L2 loss for samples in } R_1(j,s)} + \underbrace{\sum_{i:\boldsymbol{x}_i \in R_2(j,s)} \left(y_i - \hat{v}_2(j,s)\right)^2}_{\text{L2 loss for samples in } R_2(j,s)}$$

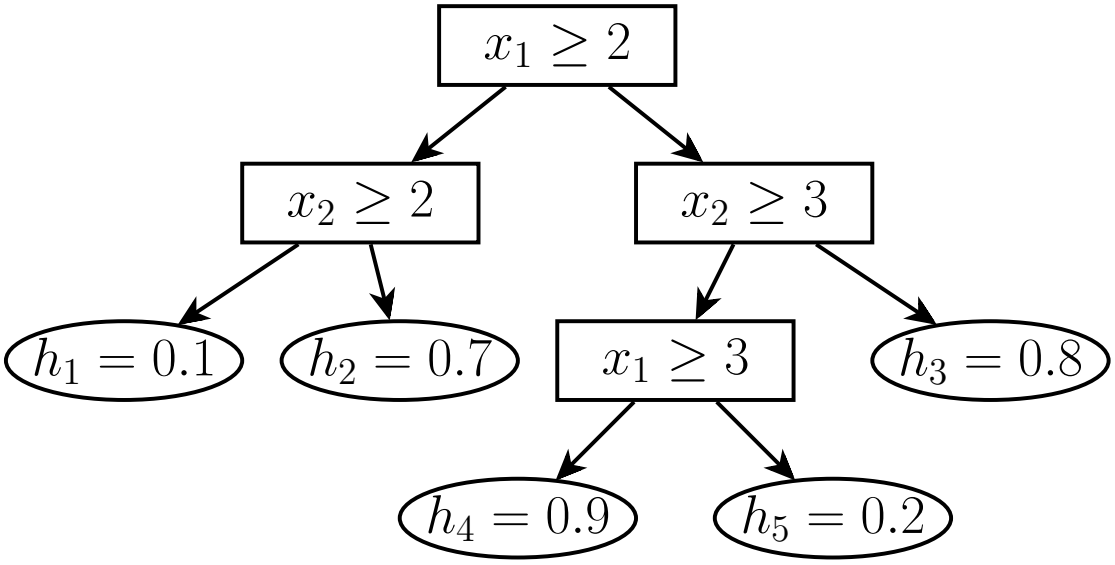
- As we did in classification, search on all (j, s)'s and find a best one.

CART Algorithm

- We have determined R_1 and R_2 from root node R.
- **Repeat** until done
  - 1. Find best split for points in R_1
  - 2. Find best split for points in R_2

Greedy Search

All previous techniques work in regression tree.



Outline

- **Decision Tree**
  - ID3 Algorithm
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - **Generalization Bound**
- Random Forest
  - Bagging
  - Breiman Algorithm

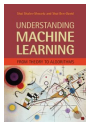
VC-Dimension Bound

- Theorem: Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$ of VC-dimension d, then for any $\delta > 0$, with probability at least $1 - \delta$, for all $h \in \mathcal{H}$:

$$\mathcal{E}_D(h) \leq \hat{\mathcal{E}}_{\mathcal{D}_n}(h) + \sqrt{\frac{2d \log \frac{en}{d}}{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}$$

- **Splitting rule:** thresholding the value of a feature $\mathbf{1}_{[x_j < \theta]}, \forall j \in [d]$.
- Decision tree splits the instance space $\mathcal{X} = \mathbb{R}^d$ into regions (leaves).
- A tree with k leaves can shatter a set of k instances.
  - $\mathbf{VCdim}(T) = k$. Become $k = 2^d$ for problems $h: \{0,1\}^d \mapsto \{0,1\}$.
  - Nonparametric! Such an approach can easily lead to overfitting.

Minimum Description Length Bound

- Theorem: (

7.3) Let $\mathcal{H}$ be a hypothesis space and let $p: \mathcal{H} \mapsto \{0,1\}^*$ be a prefix-free description language for $\mathcal{H}$. Denote $|h|$ as the length of $p(h)$. With probability at least $1-\delta$ over choices $\mathcal{D}_n \sim D^n$,

$$\forall h \in \mathcal{H}, \qquad \mathcal{E}_D(h) \leq \hat{\mathcal{E}}_{\mathcal{D}_n}(h) + \sqrt{\frac{|h| + \log\frac{2}{\delta}}{2n}}$$

- Theorem: With probability at least $1-\delta$ over a sample of size n, for every m and every decision tree $h \in \mathcal{H}$ with m nodes it holds that

$$\forall h \in \mathcal{H}, \qquad \mathcal{E}_D(h) \leq \hat{\mathcal{E}}_{\mathcal{D}_n}(h) + \sqrt{\frac{(m+1)\log_2(d+3) + \log\frac{2}{\delta}}{2n}}$$

Minimum Description Length Bound

- Proof: Description for decision tree with m nodes is prefix-free.
  - A tree described in $m+1$ blocks, each of size $\log_2(d+3)$ bits.
  - The first m blocks encode the nodes, in a depth-first order.
  - The last block marks the end of the code.

For each block:

- An internal node of the form $\mathbf{1}_{[x_j<\theta]}$ for some $j \in [d]$.
- A leaf whose value is 1.
- A leaf whose value is 0.
- End of the code.

  - Overall, there are $d+3$ options, and require $\log_2(d+3)$ bits.

- Prior knowledge (inductive bias): prefer small trees over large trees.

Decision Tree

- Advantages:
  - Inexpensive to construct and fast at classifying new samples.
  - Easy to interpret for small-sized trees (caution: unstable!).
- Trees make no use of geometry
  - A nonmetric method, feature scale irrelevant (vs. SVM or KNN).
  - Can simultaneously cope with categorical and continuous variables.
  - Can naturally cope with missing values and noisy data (**why?**)
- Prediction functions are not continuous
  - Not so bad for classification, but may be undesirable for regression.

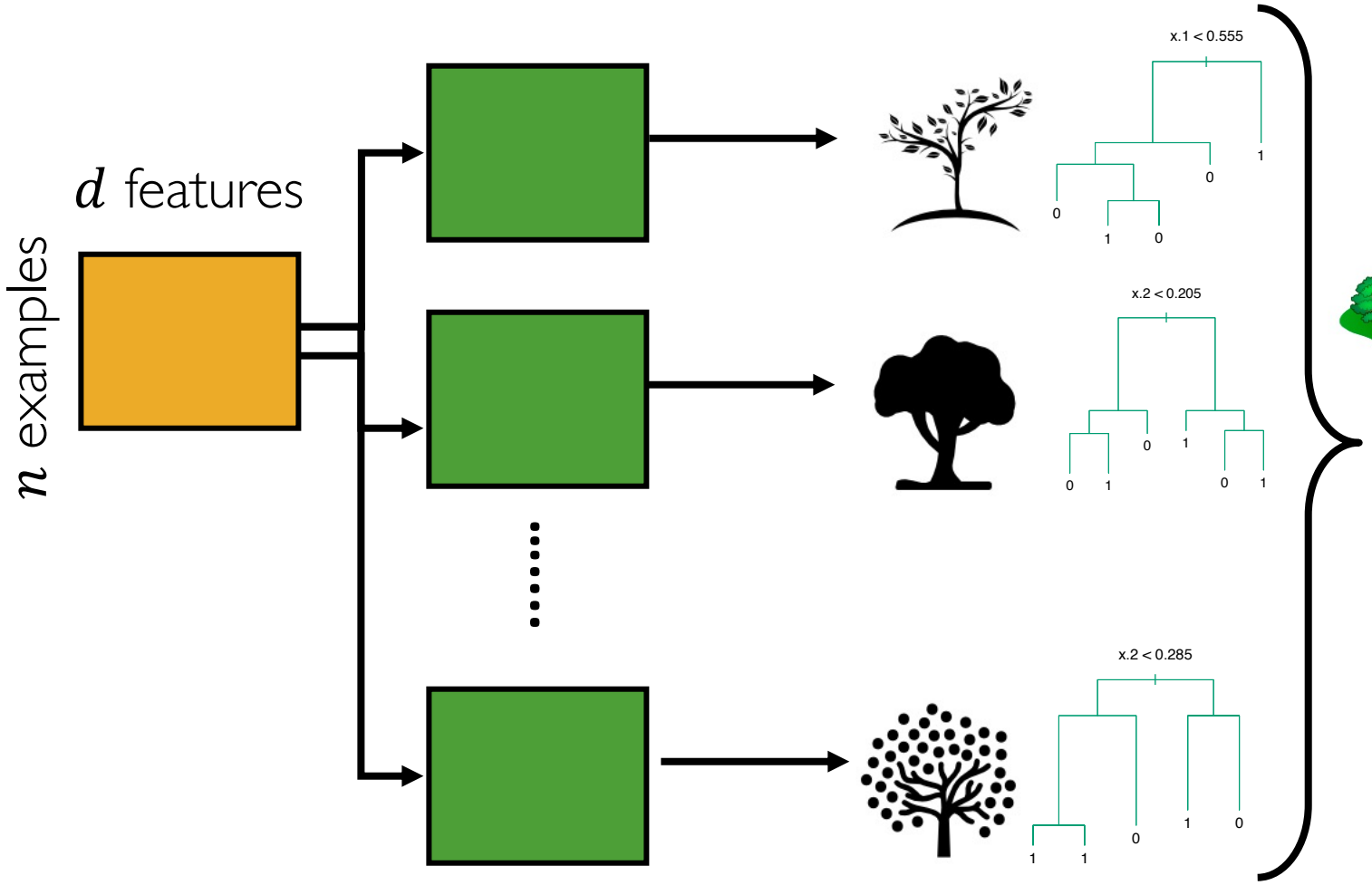
Outline

- Decision Tree
  - ID3 Algorithm
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - Generalization Bound
- **Random Forest**
  - Bagging
  - Breiman Algorithm

Ensemble Learning

- Ensemble methods train multiple learners and combine them for use.

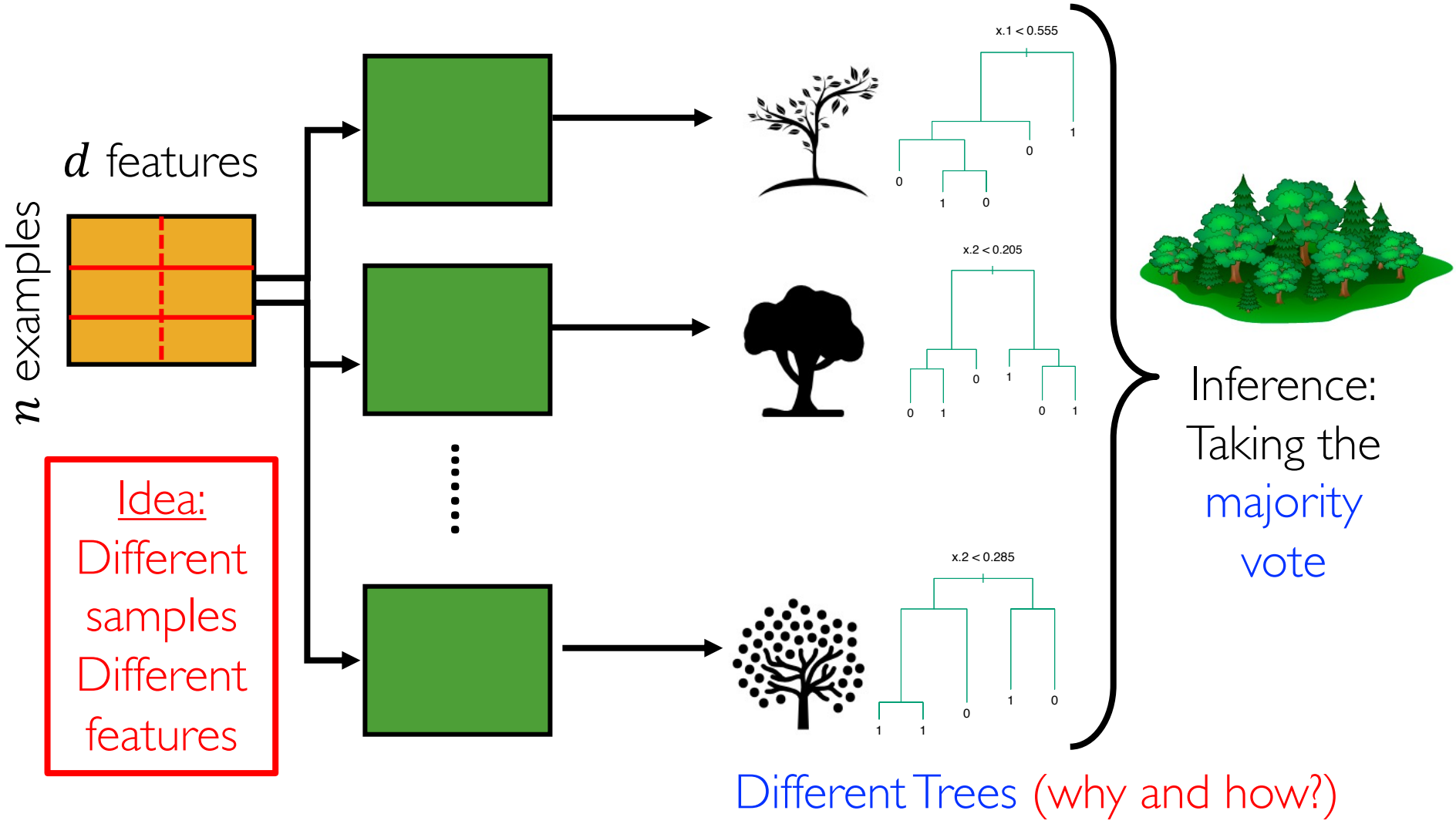
Random Forest



Inference: Taking the majority vote

Different Trees (why and how?)

Random Forest

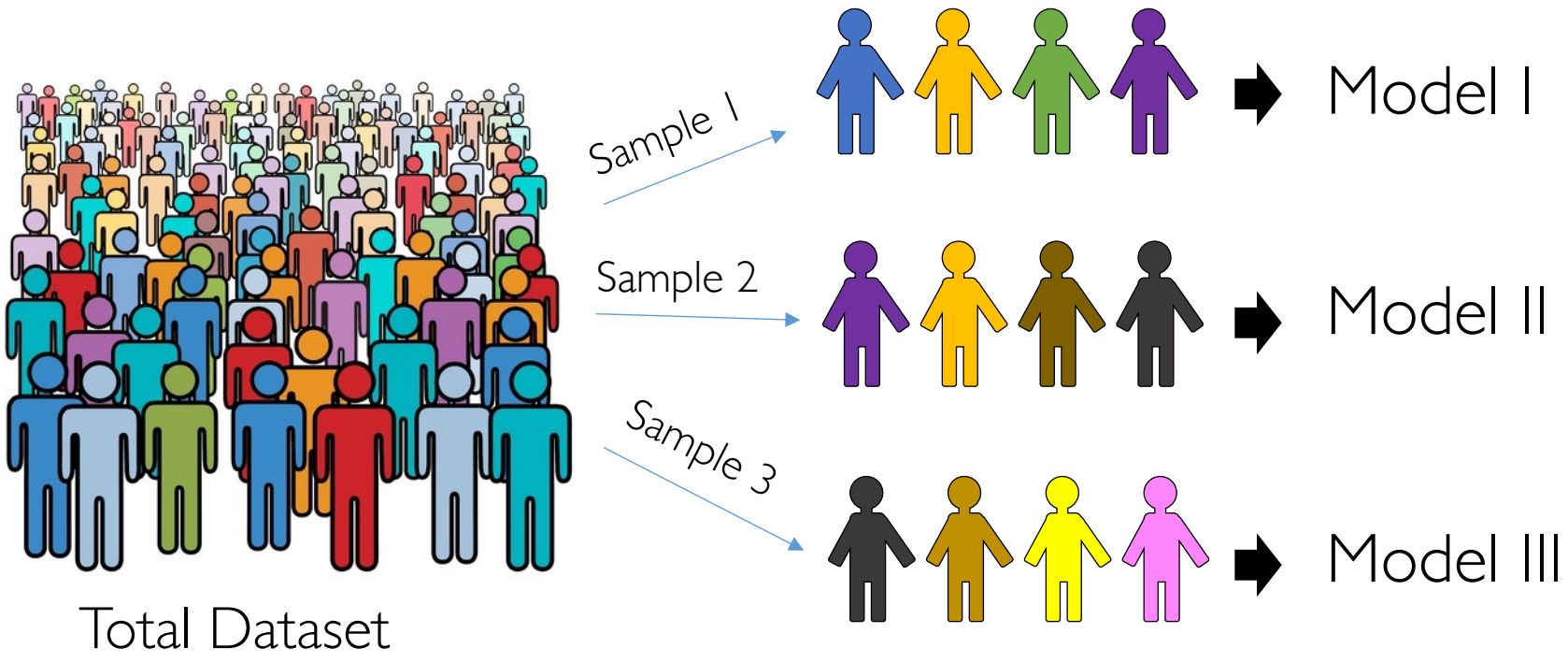


Outline

- Decision Tree
  - ID3 Algorithm
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - Generalization Bound
- **Random Forest**
  - **Bagging**
  - Breiman Algorithm

Bootstrap Sample

- Bootstrap (自助法): randomly draw datasets with replacement from the training data, with the same sample size as the original training set



Bootstrap Sample

- **Definition**: A bootstrap sample from $\mathcal{D}_n$ is a sample of size n drawn with replacement from $\mathcal{D}_n$.
- In a bootstrap sample, some elements of $\mathcal{D}_n$
  - will show up multiple times.
  - some will not show up at all. Will this happen?
- Each X_i has a probability $\left(1 - \frac{1}{n}\right)^n$ of being not selected.
- Recall from calculus that for large n,

$$\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e} \approx 0.368.$$

- So we expect $\sim 63.2\%$ of elements of $\mathcal{D}$ will show up at least once.

Bootstrap Method

- **Definition:** A bootstrap method is when you simulate B independent samples from P by taking B bootstrap samples from the dataset $\mathcal{D}_n$.

- Given original data $\mathcal{D}_n$ compute B bootstrap samples $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^B$
- For each bootstrap sample, compute some function:

$$\phi(\mathcal{D}_n^1), \ldots, \phi(\mathcal{D}_n^B)$$

  - work with these values as though $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^B$ were i.i.d. from P.
  - For example, find some independent classifiers $h_{\mathcal{D}_n^1}, \ldots, h_{\mathcal{D}_n^B}$.

- Amazing fact:
  - Things $\phi(\mathcal{D}_n^1), \ldots, \phi(\mathcal{D}_n^B)$ often come out very close to what we will obtain with independent samples from P.

Bootstrap Aggregation (Bagging)

- Bagging is a general technique to reduce the variance of an estimator.

- Draw B bootstrap samples $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^B$ from original data $\mathcal{D}_n$.
- Let $h_{\mathcal{D}_n^1}, \ldots, h_{\mathcal{D}_n^B}$ be the prediction functions for each sample.
- The bagged prediction function is a combination of these functions:

$$h_{\text{bag}}(x) = \text{Combine}\left(h_{\mathcal{D}_n^1}(\boldsymbol{x}), \ldots, h_{\mathcal{D}_n^B}(\boldsymbol{x})\right)$$

- How can we combine:
  - Binary/multiclass predictions } Vote
  - Class probability predictions } Average
  - Prediction functions for regression } Average

$$\text{Var}(h_{\text{bag}}) \leq \text{Var}(h_{\mathcal{D}_n})$$

Leo Breiman

Out-of-Bag Validation (OOB)

- Each bagged prediction function is trained on $\sim 63\%$ of the data.
- The remaining 37% are called out-of-bag (OOB) observations.
- For each training point $\boldsymbol{x}_i$, let

$$S_i = \{b | \mathcal{D}_n^b \text{ does not contain } \boldsymbol{x}_i\}$$

- The OOB prediction on $\boldsymbol{x}_i$ is

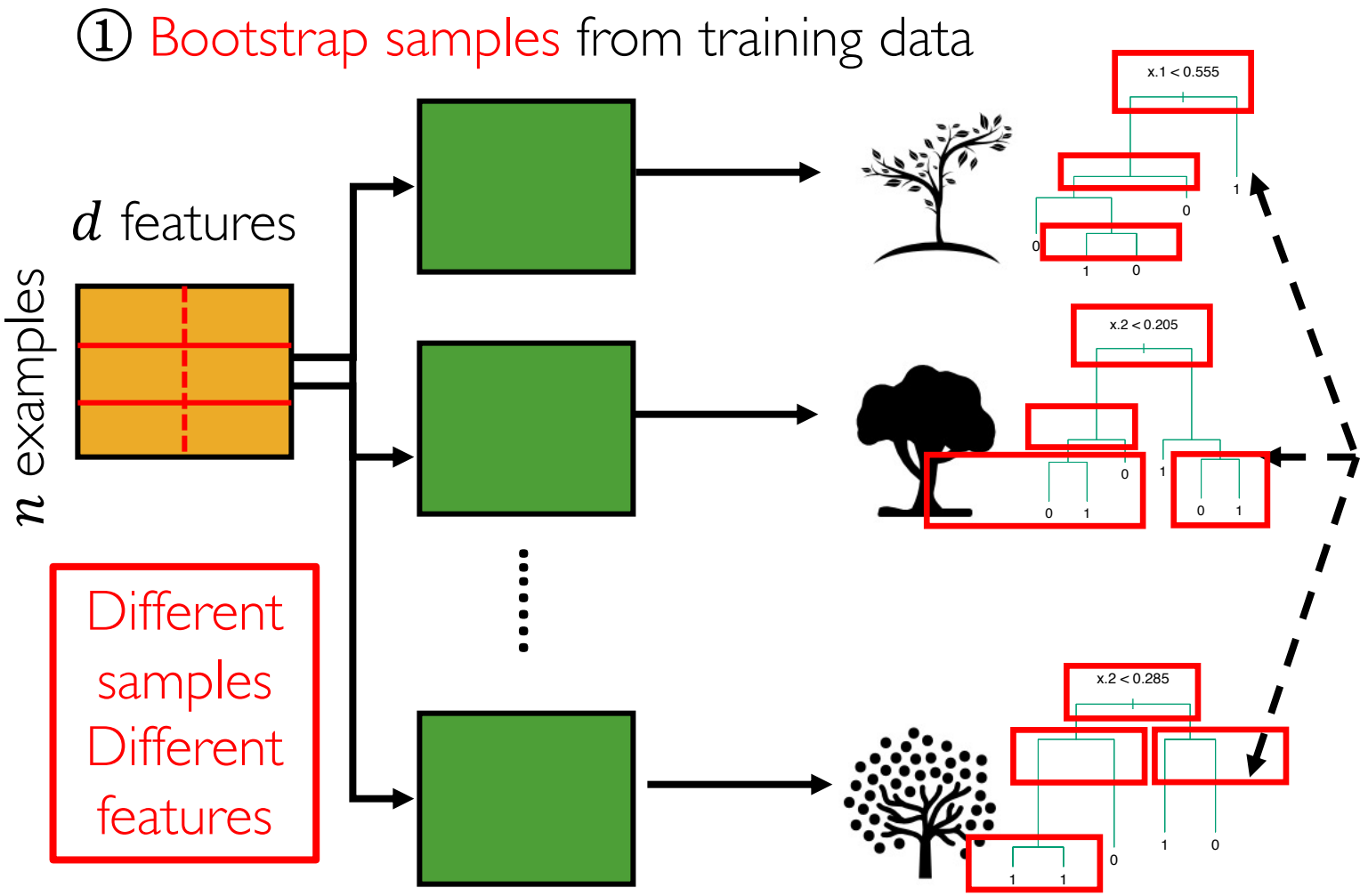
$$h_{\text{OOB}}(\boldsymbol{x}_i) = \frac{1}{|S_i|} \sum_{b \in S_i} h_{\mathcal{D}_n^b}(\boldsymbol{x}_i)$$

- The error of h_{OOB} is a good estimate of the test error.
- OOB error is akin to validation error: computed on training subset.

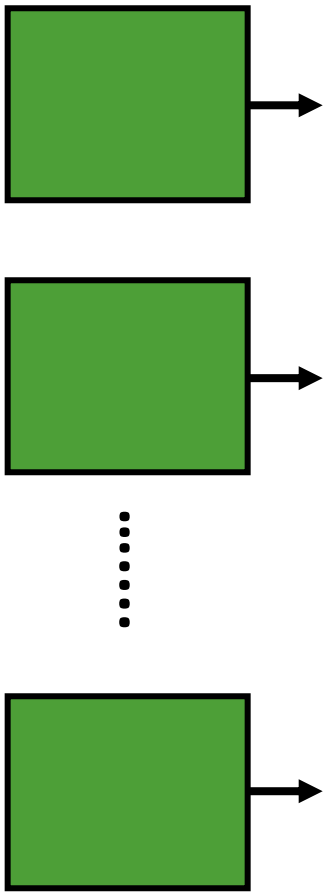
Outline

- Decision Tree
  - ID3 Algorithm
  - C4.5 Algorithm
  - Classification and Regression Tree (CART)
  - Generalization Bound
- **Random Forest**
  - Bagging
  - **Breiman Algorithm**

Random Forest



Breiman Algorithm



Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

each tree

Process:

1. $N \leftarrow$ create a tree node based on D;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K);
11. $N_r \leftarrow$ call the process with parameters (D_r, K);
12. **return** N

$K \approx \sqrt{d}$

Randomized features

Output: A random decision tree

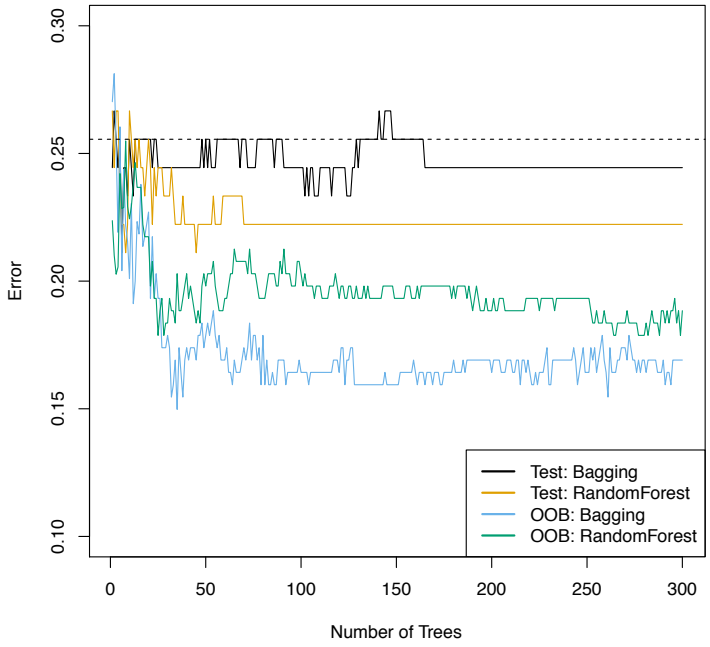
Random forests
L Breiman
Machine learning 45 (1), 5-32

By Leo Breiman

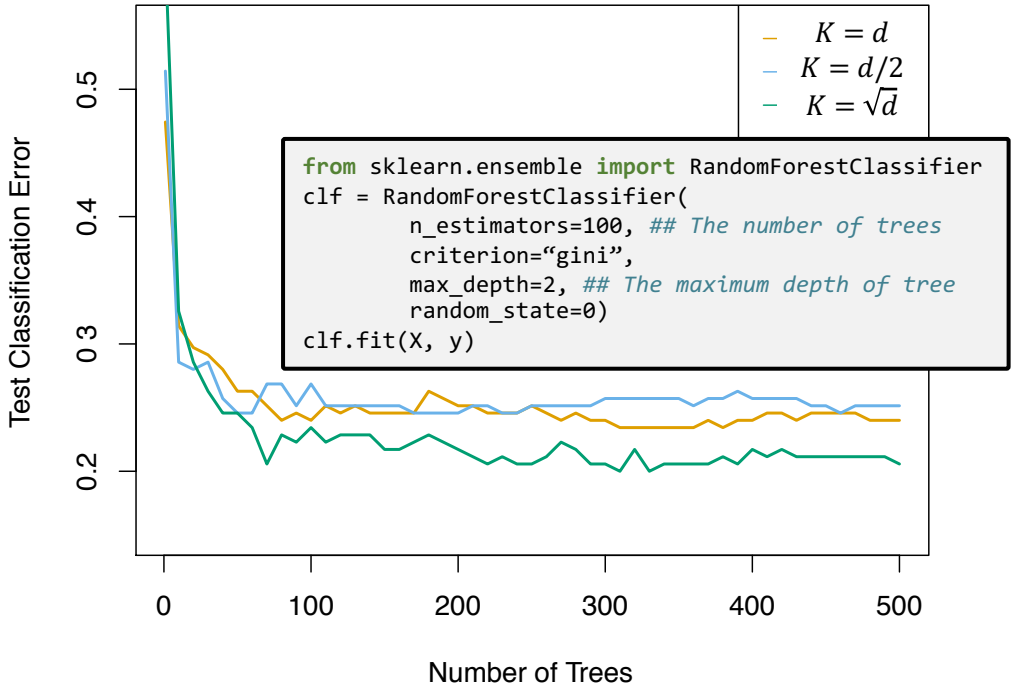
50227 2001

Breiman Algorithm

A widely-used algorithm in practice!



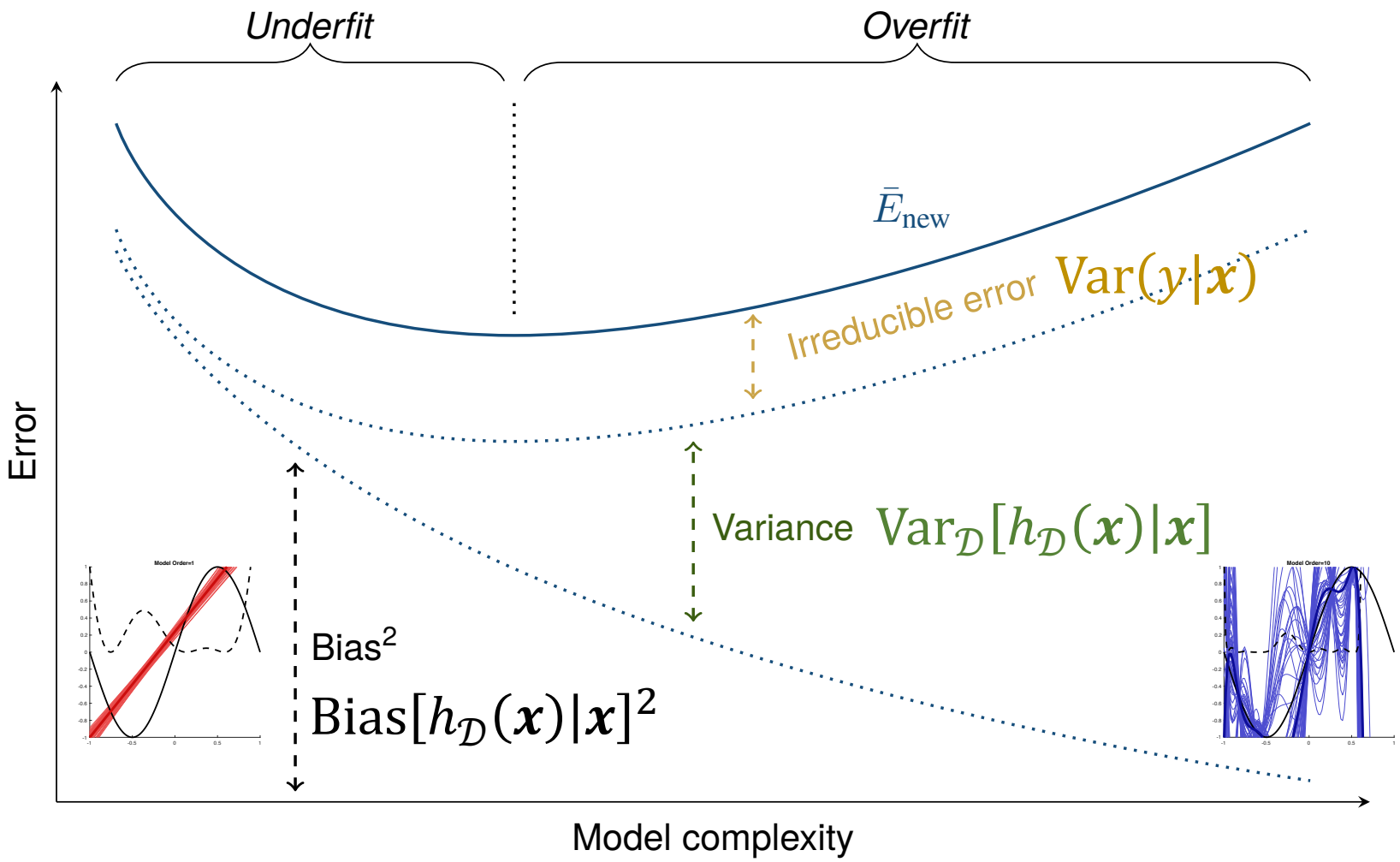
Bootstrap samples




```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(
        n_estimators=100, ## The number of trees
        criterion="gini",
        max_depth=2, ## The maximum depth of tree
        random_state=0)
clf.fit(X, y)
```

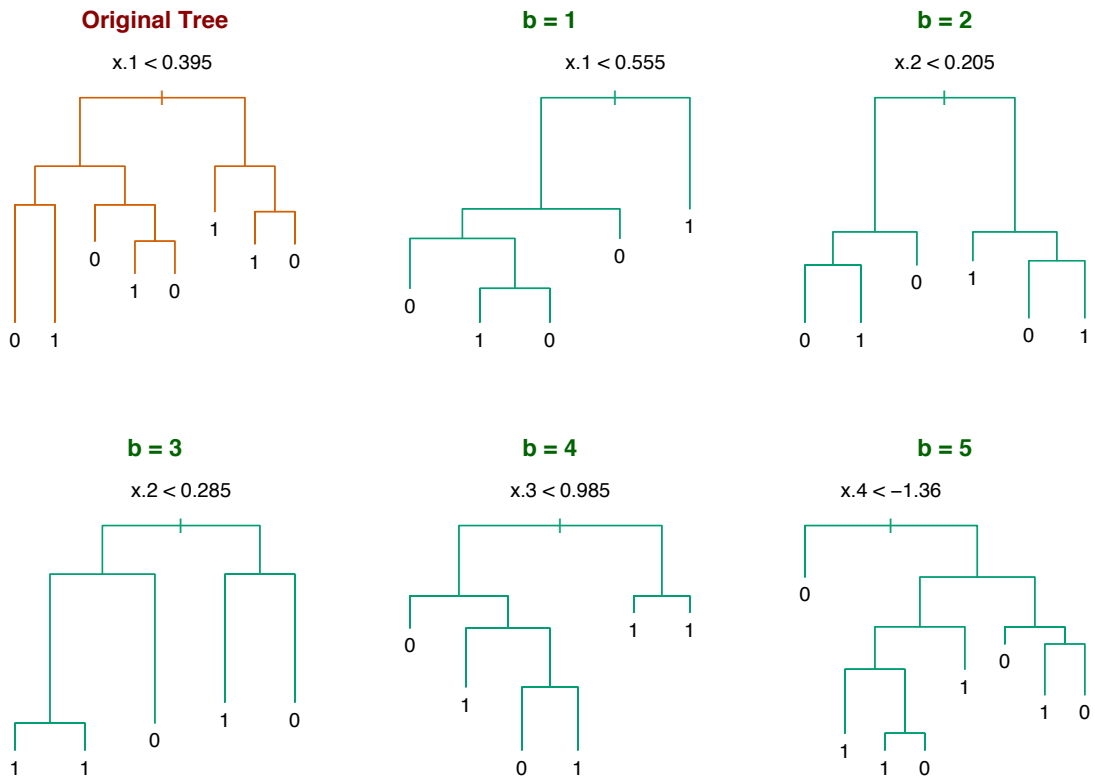
Randomized features

Bias-Variance Decomposition



Decision Trees Have High Variance

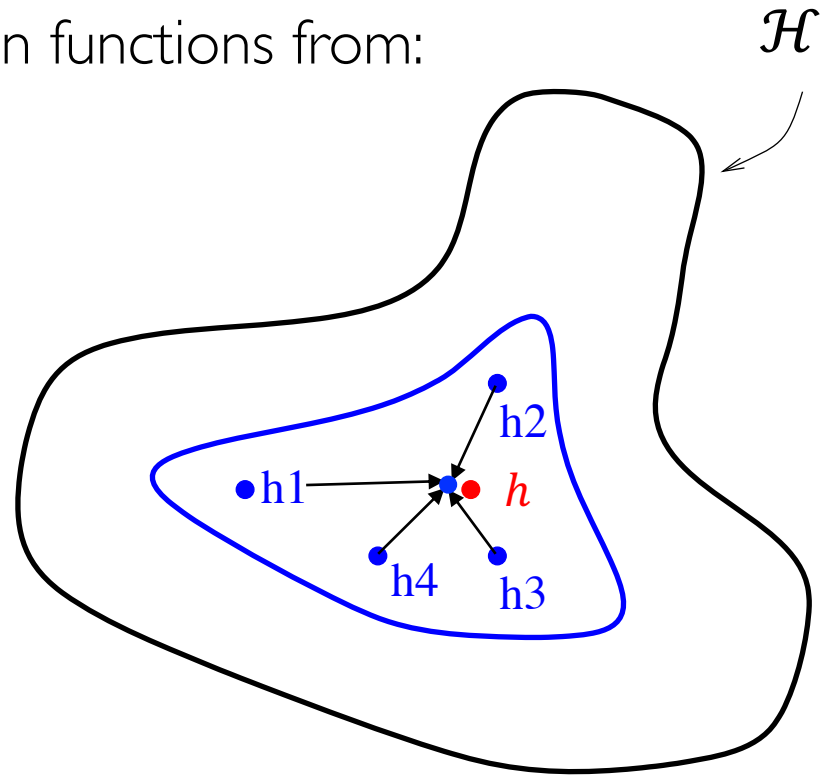
- Input space $\mathcal{X} = \mathrm{R}^5$ and output space $\mathcal{Y} = \{-1,1\}$.



- Sample size $n = 30$
- Each bootstrap tree is quite different.
  - Different splitting variable at the root.
- This shows that tree models suffer from high variance.
- Bagging is needed!

Random Forest

- A usual approach is to build very deep trees
  - Low training error, high generalization error.
- Diversity in individual tree prediction functions from:
  - Bootstrap samples
  - Randomized tree building
- Bagging works better when we are combining a diverse set of prediction functions.
- RF find a way to achieve them.



Thank You

Questions?

Mingsheng Long
mingsheng@tsinghua.edu.cn
http://ise.thss.tsinghua.edu.cn/~mlong
答疑：东主楼11区413室