Lec11 Machine Learning
11-3 Tree Model

Yang Shu

School of Data Science and Engineering

yshu@dase.ecnu.edu.cn

[Acknowledgement: Slides are adapted from Machine Learning Course, Mingsheng Long, THU]

Lec11-3 Tree Model

- **Decision Tree**
  - ID3 Algorithm
  - C4.5 Algorithm
- Random Forest

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

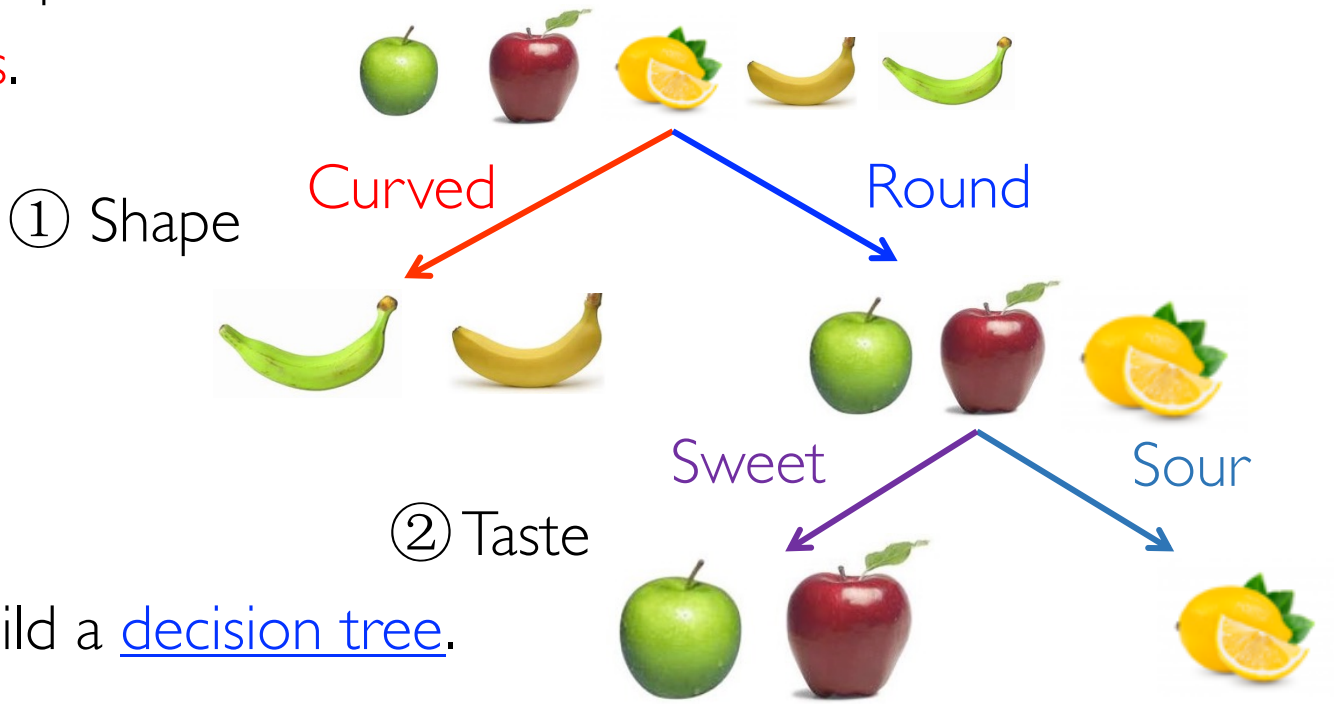
green, curved, no leaf, sweet, … — Banana

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.



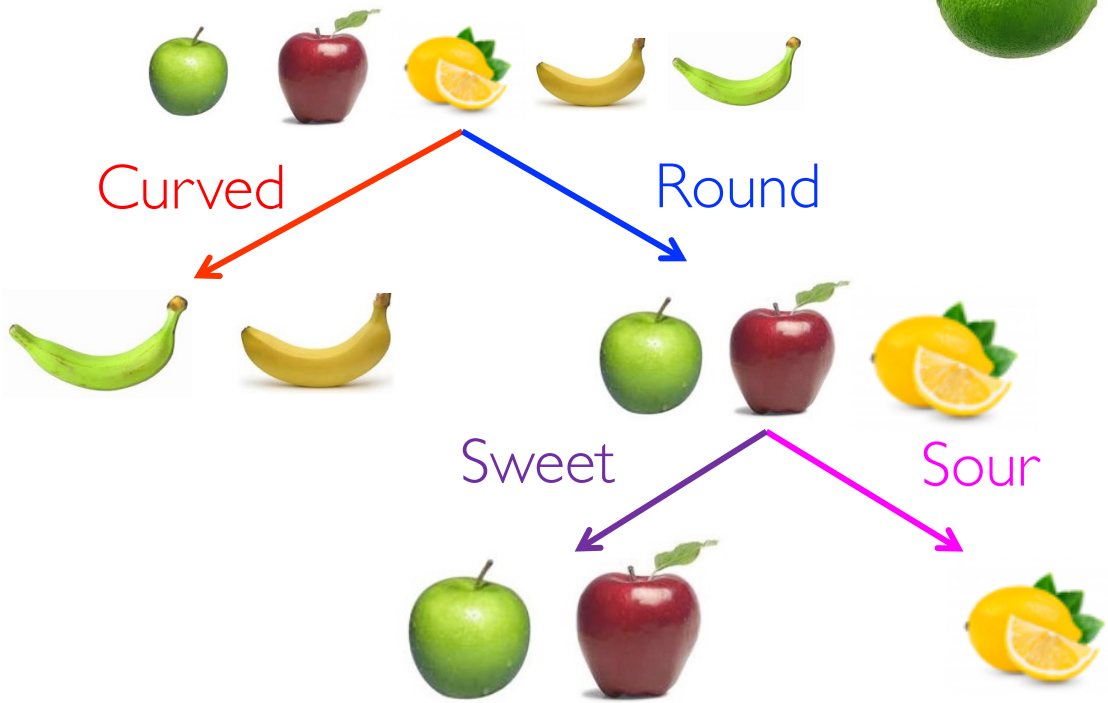
- At last, we build a decision tree.

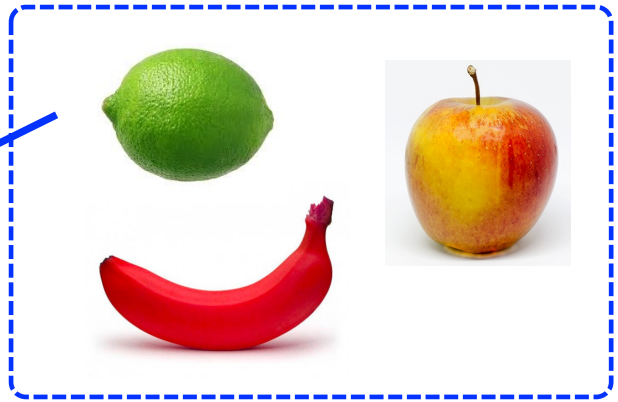
Decision Tree

Interpretability ✓

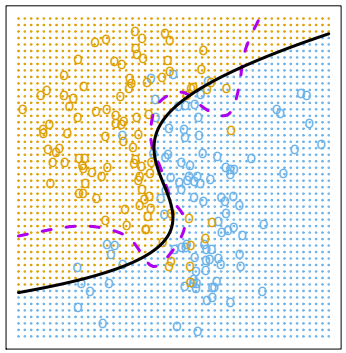
Generalize to unseen data ✓

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





Compared with LM:

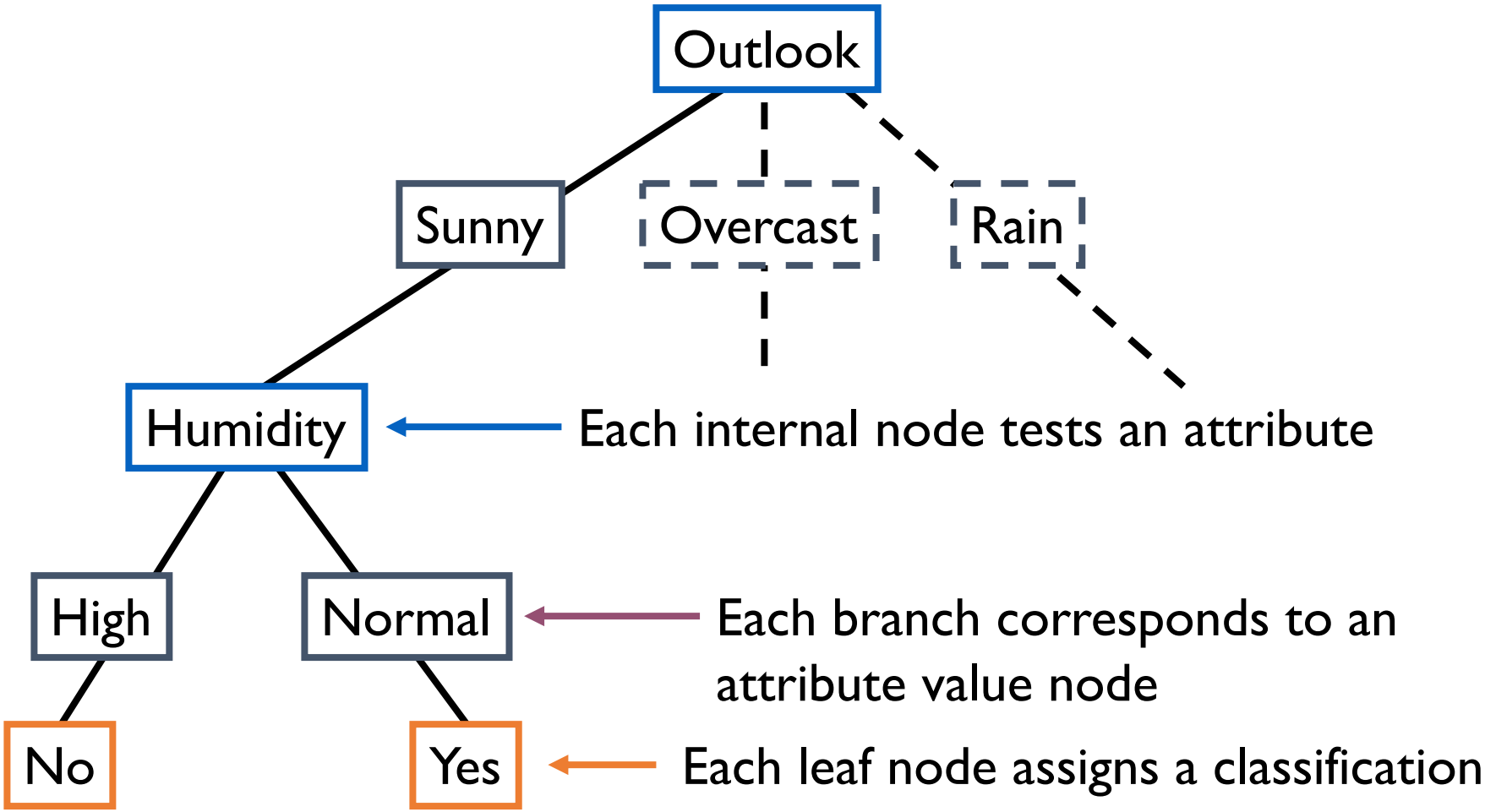


Decision Tree

- 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

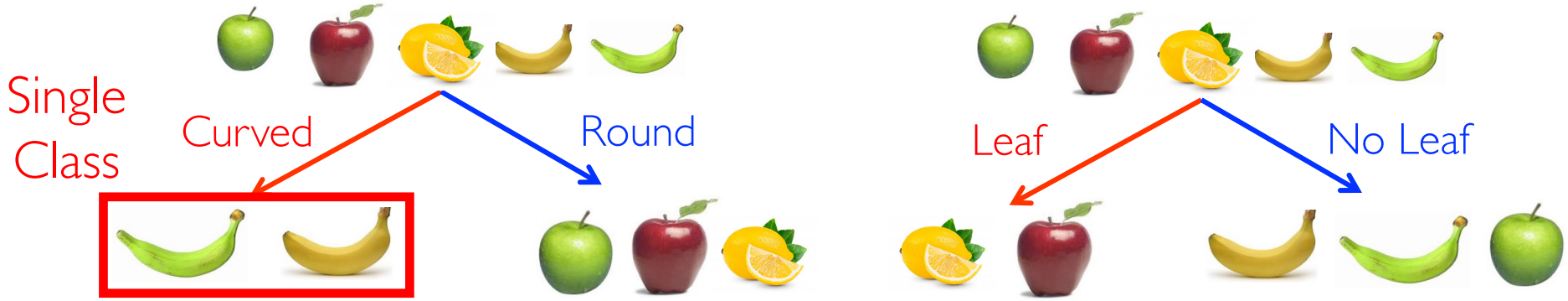


Outline

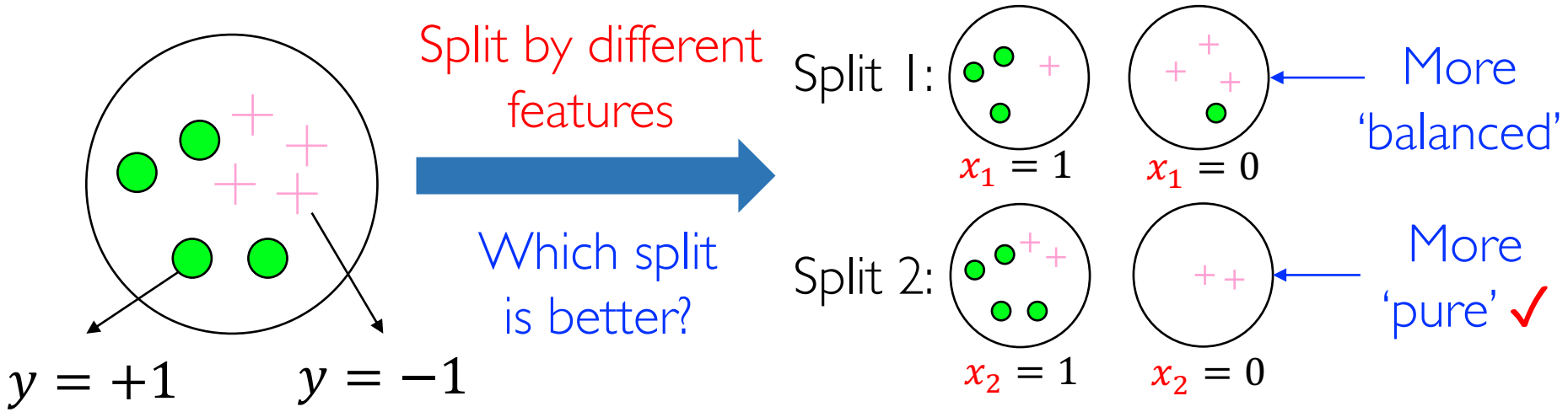
- **Decision Tree**
  - **ID3 Algorithm**
  - C4.5 Algorithm
- Random Forest

Node Splitting

- Which split is better?



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

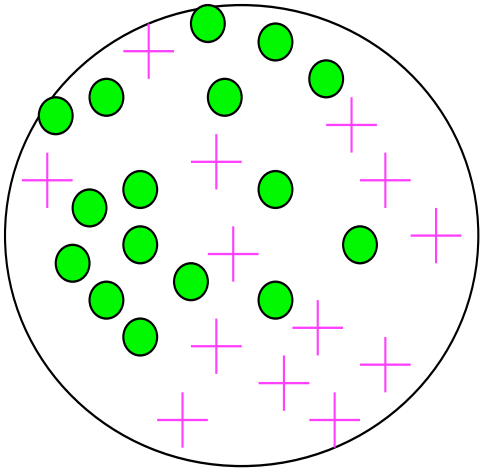


How to Measure a Node?

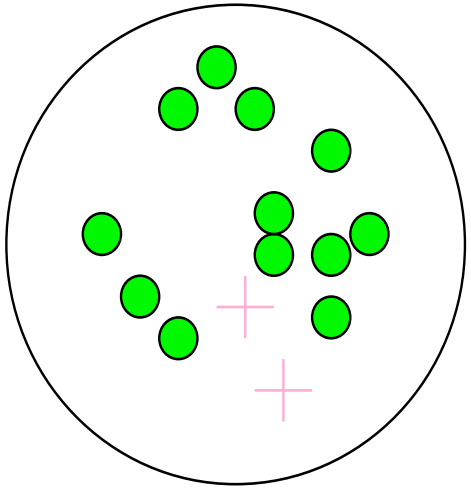
- We want pure leaf nodes, as close to a single class as possible:
  - Lead to a lower classification error.
  - The classification error is $\min(\frac{|\mathcal{C}_1|}{|\mathcal{D}|}, \frac{|\mathcal{C}_2|}{|\mathcal{D}|})$.

Number of samples in each class

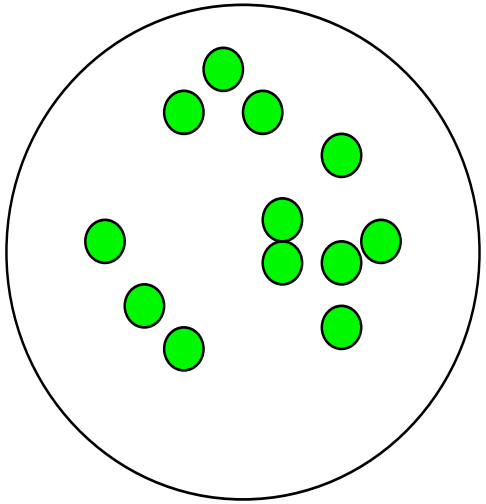
- We'll choose the splitting feature that minimizes impurity measure.



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:

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

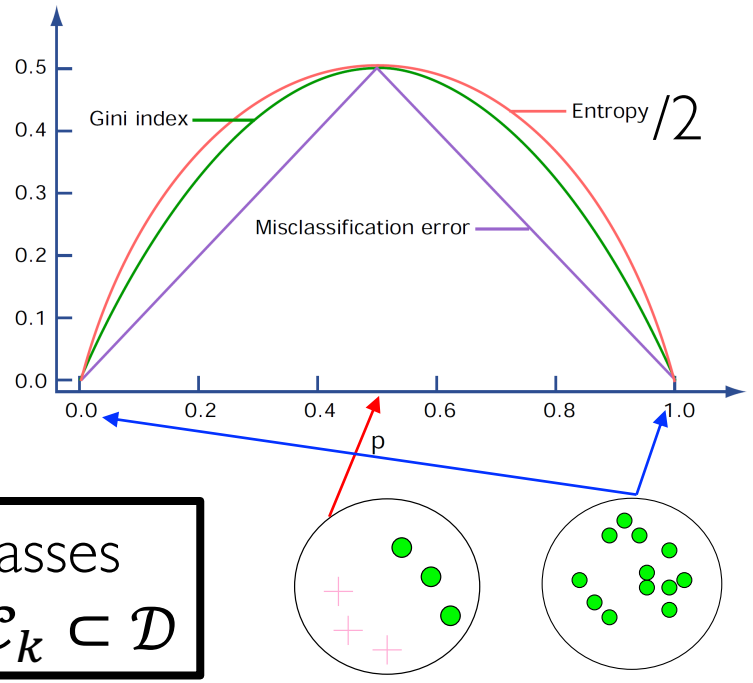
  - Entropy (used in ID3 and C4.5):

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

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

  - Gini index (used in CART):

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



Max:

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

Min:

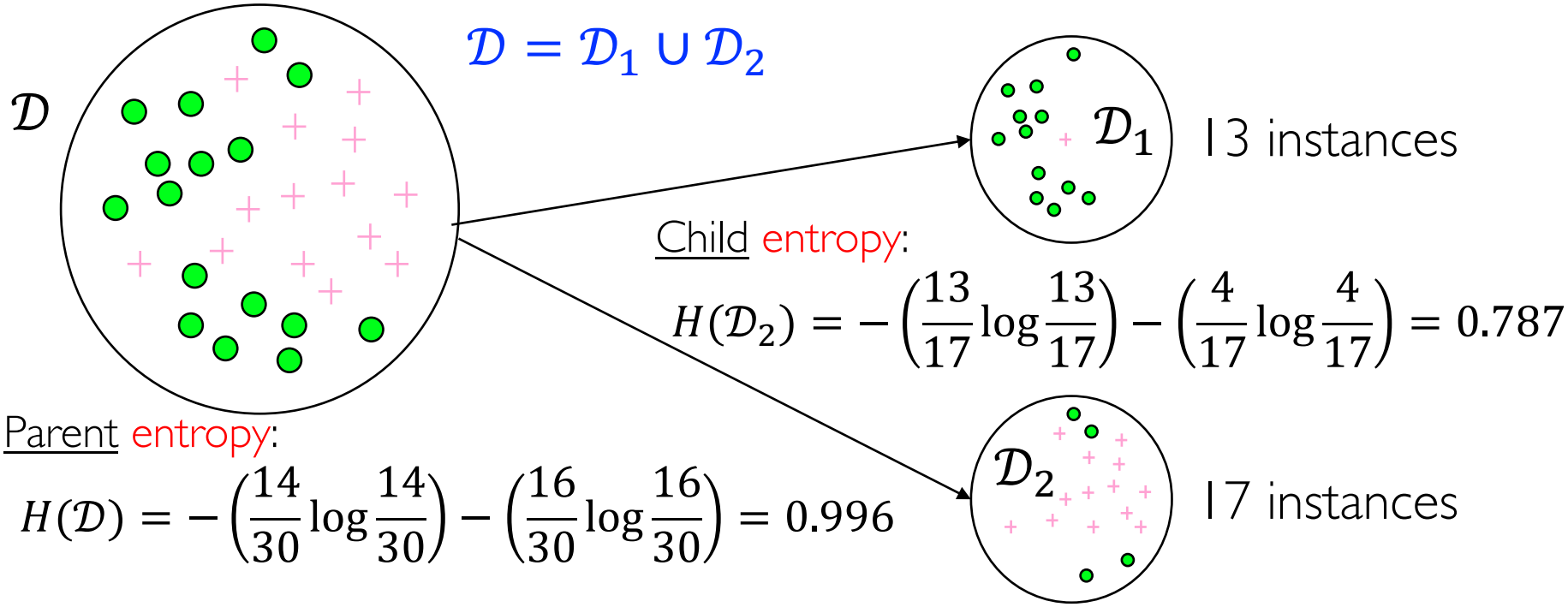
$$\frac{|\mathcal{C}_k|}{|\mathcal{D}|} = 0$$

$(0 \log 0 = 0)$

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:

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

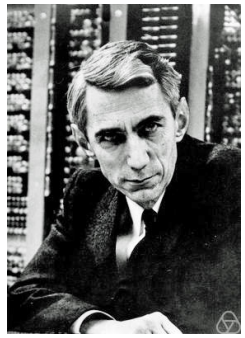
- Equivalent to maximizing the Information Gain (IG):

Child nodes

Parent node

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

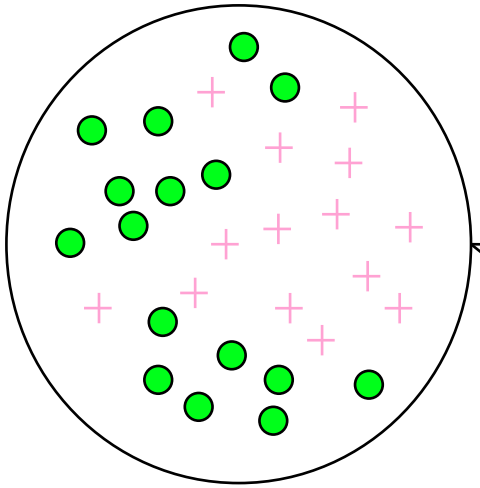
- In information theory, entropy is the expected number of bits needed to encode a randomly drawn value of X.
- Larger entropy, less information. That's why we call it IG. https://en.wikipedia.org/wiki/Entropy_(information_theory)



Claude Shannon

Calculating Information Gain (IG)

Entire population (30 instances)

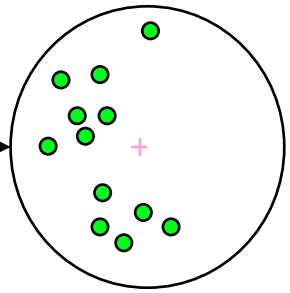


Parent entropy:

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

Child entropy:

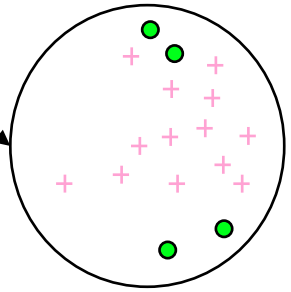
$$-\left(\frac{1}{13}\log\frac{1}{13}\right)-\left(\frac{12}{13}\log\frac{12}{13}\right)=0.391$$



13 instances

Child entropy:

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

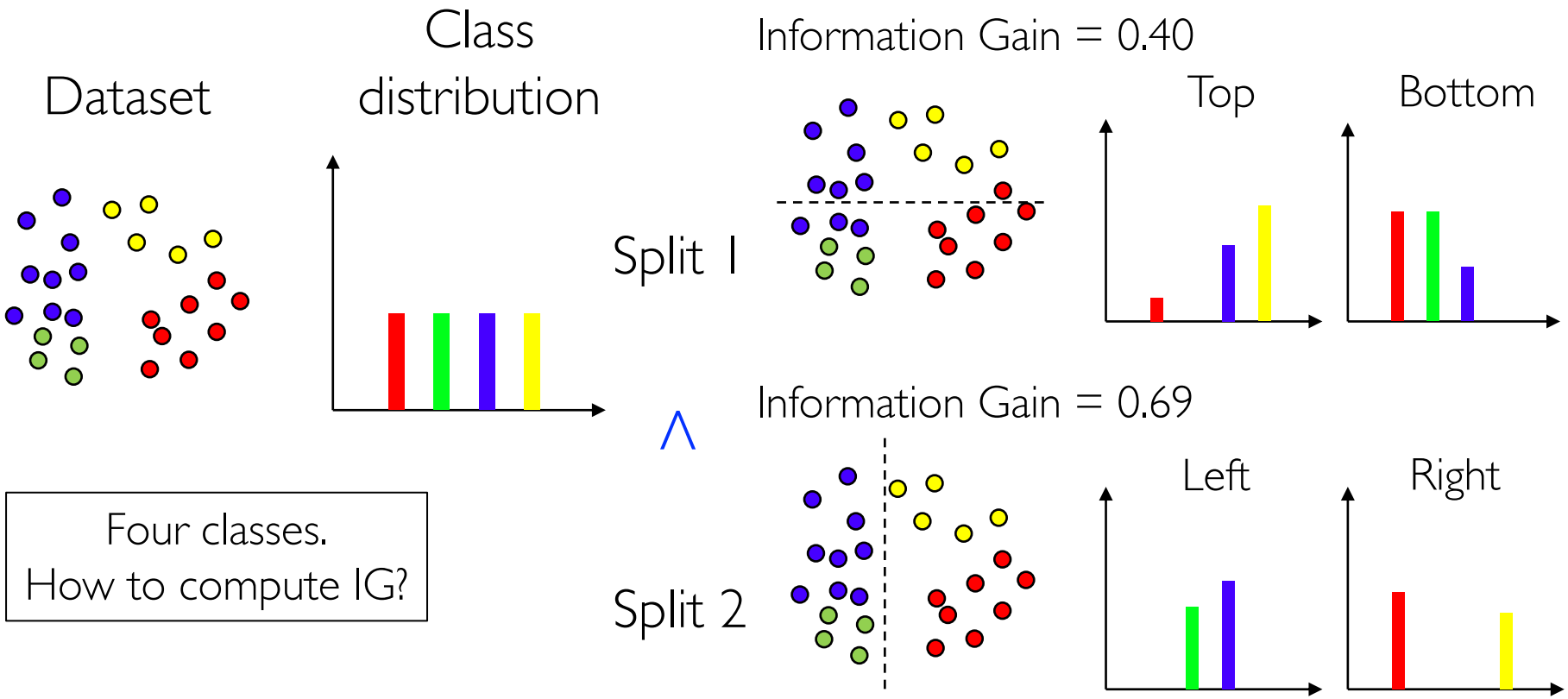


17 instances

Information Gain (IG):

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

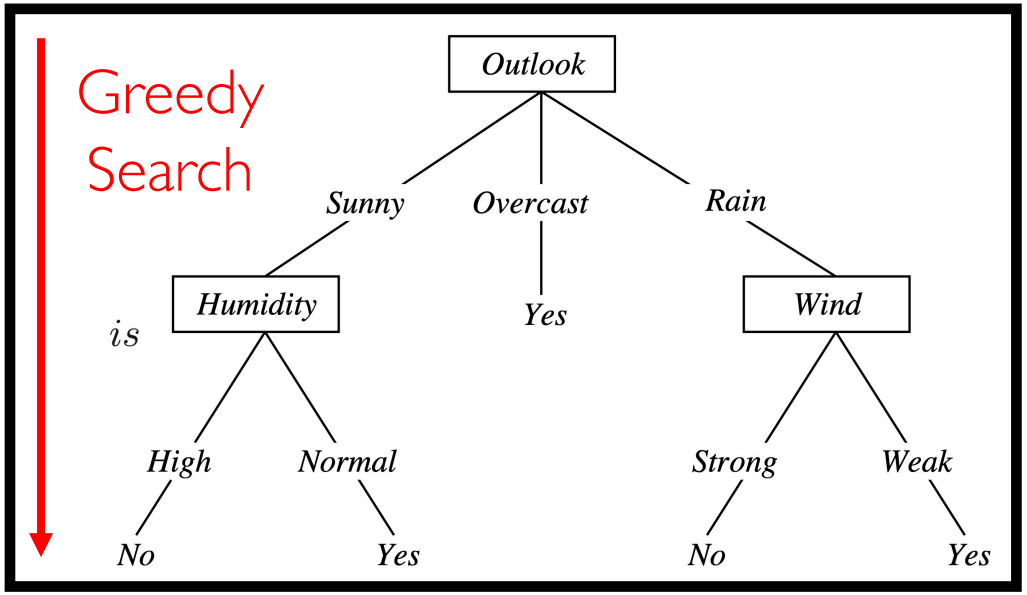
Split Search



- 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*$\backslash\{A\}$);

- return *Root*;

Stop Criteria

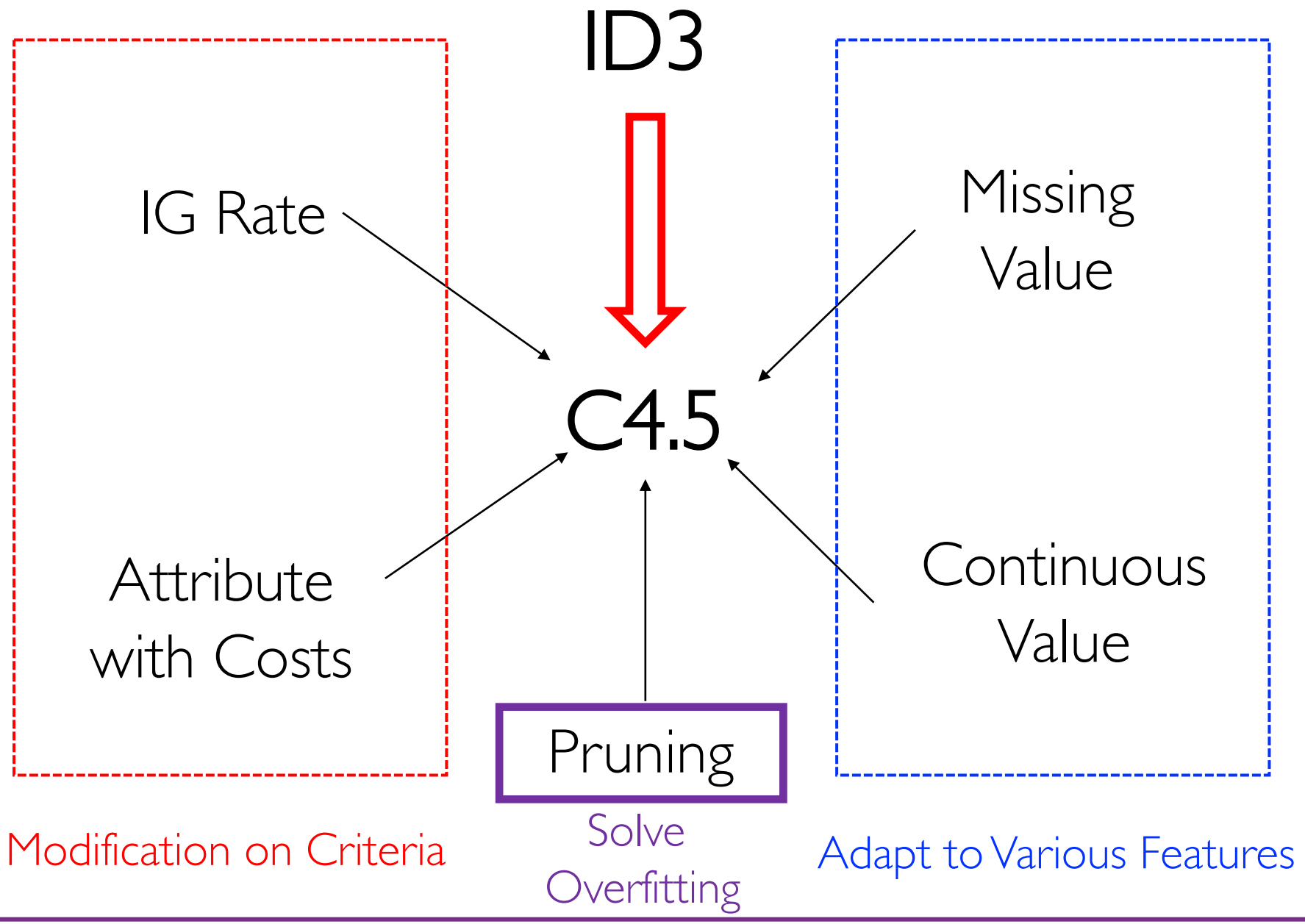
$O(dn)$ in each layer

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

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

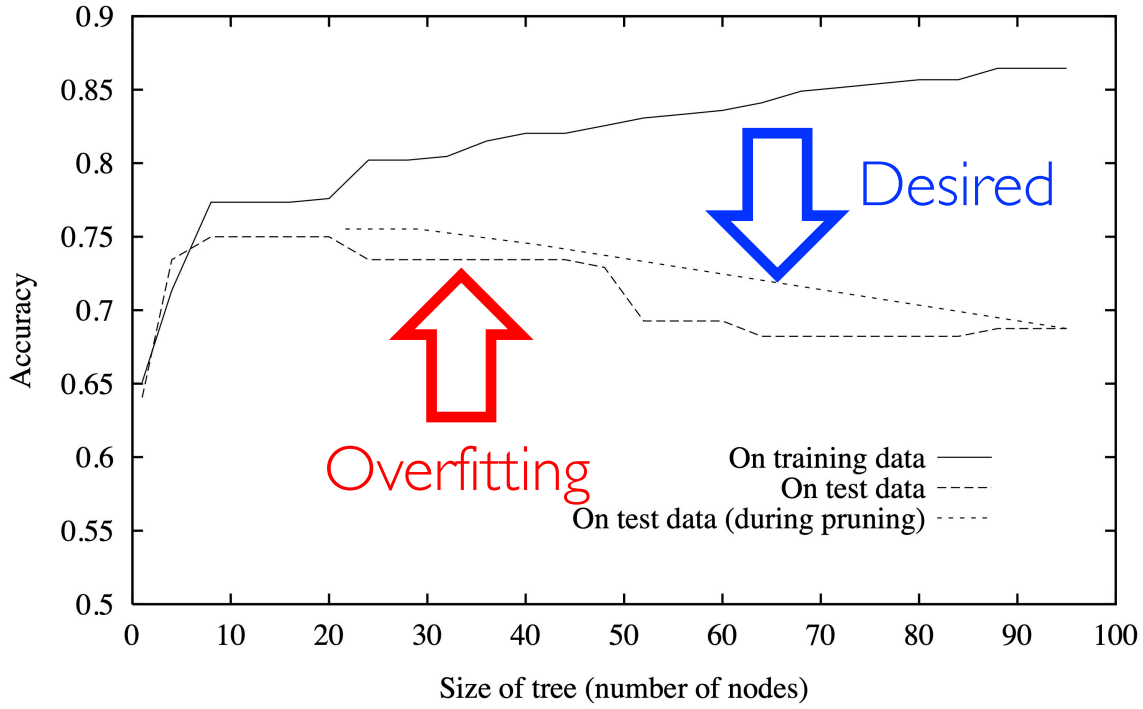
Outline

- **Decision Tree**
  - ID3 Algorithm
  - **C4.5 Algorithm**
- Random Forest


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

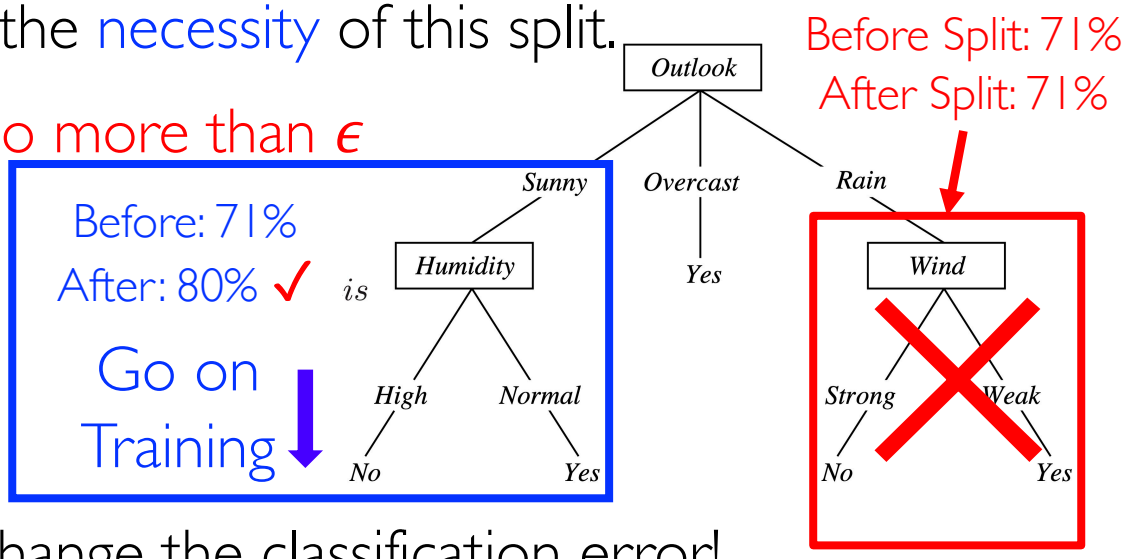
Overfitting in Decision Tree

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



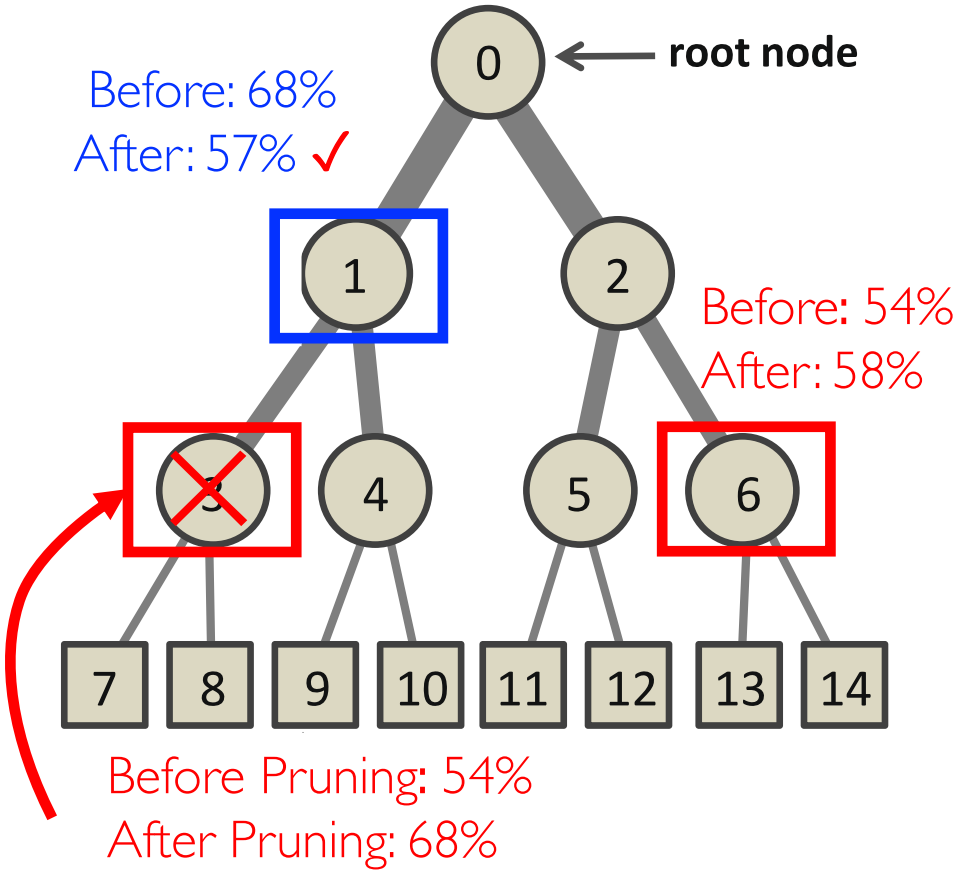
Pre-Pruning

- 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.



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
- **Random Forest**

Ensemble Learning

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

Random Forest