Decision Trees

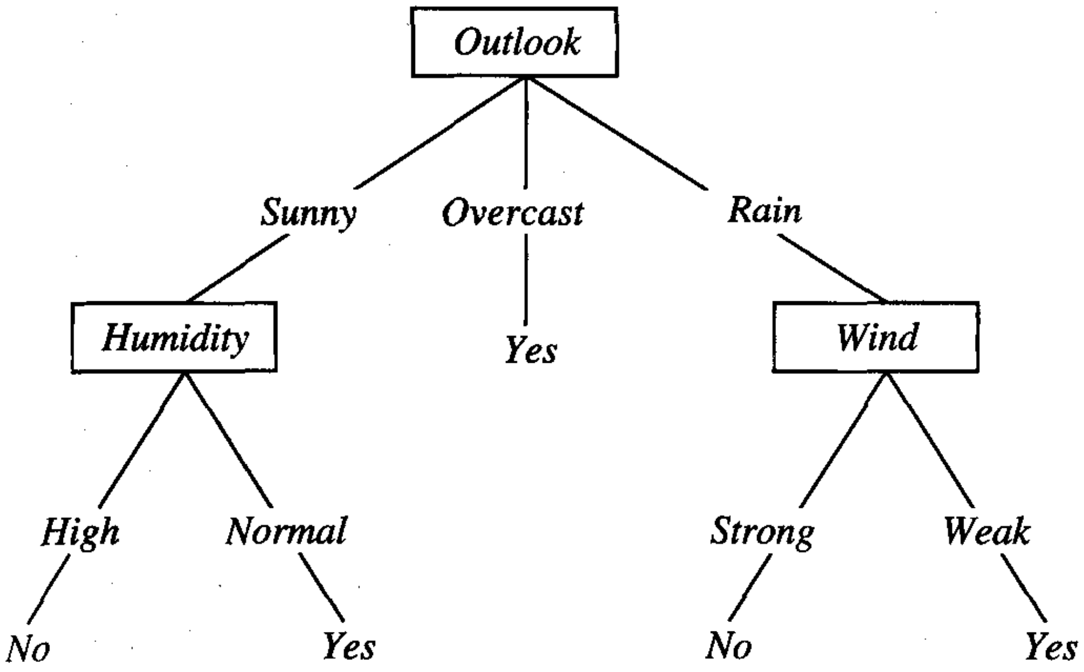
Zhiyao Duan
Associate Professor of ECE and CS
University of Rochester

Some figures are copied from the following books

- **LWLS** - Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, *Machine Learning: A First Course for Engineers and Scientists*, Cambridge University Press, 2022.
- **Mitchell** - Tom M. Mitchell, *Machine Learning*, McGraw-Hill Education, 1997.

What is a decision tree?

- A tree used to sort a test example through internal nodes to a leaf node for decision making

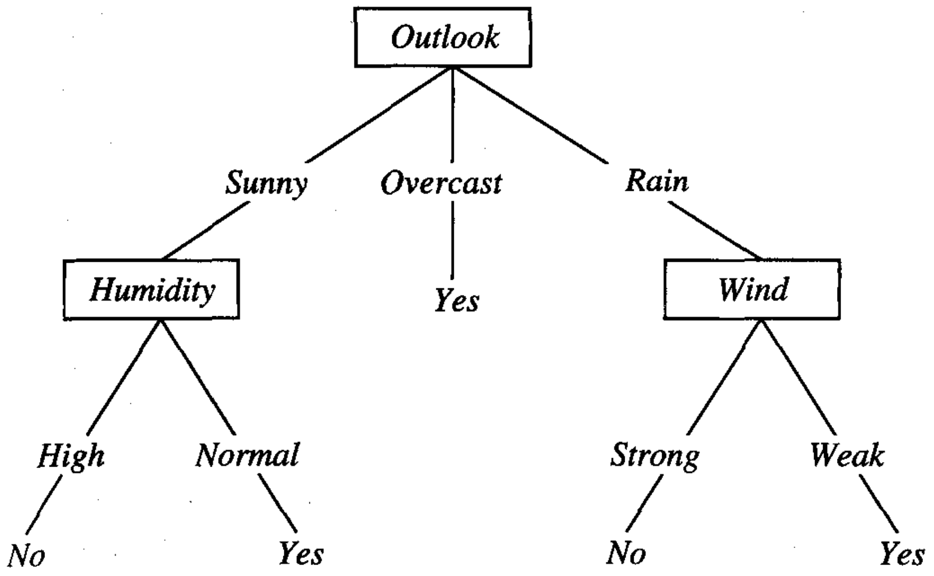


(Fig. 3.1 in Mitchel)

Shall we play tennis if <Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong>?

Terminology of Decision Trees

- Attributes (features) can be categorical or numeric
  - Let's start from categorical attributes
- Internal node: chooses one attribute and split
  - Categorical: split fully
  - Numerical: split into two
- Leaf node: makes final decision
- Root, descendants and subtrees



(Fig. 3.1 in Mitchel)

- Path from root to a leaf node is a conjunction rule
- Learned concept: disjunction of conjunctions

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast)$$
$$\vee \quad (Outlook = Rain \wedge Wind = Weak)$$

Learning Decision Tree

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Goal: find a decision tree that sorts all training examples to leaf nodes
- Naïve idea: traverse all possible trees

How many trees are there?

- Let M be the number of attributes, all categorical
- Let m_i be the number of values for the i-th attribute
- Let C be the number of classes
- Root: M choices (splitting) + C choice (not splitting)
  - If root takes the i-th attribute, then it has m_i branches
  - Each branch's root: $M-1$ choices (splitting) + C choice (not splitting)
    - Recursion till all attributes are traversed
- Let $N(\mathcal{A})$ be the number of possible trees constructed using attribute set $\mathcal{A}=\{1,\ldots,M\}$

$$N(\mathcal{A})=\sum_{i=1}^{M}\left(N(\mathcal{A}\backslash\{i\})\right)^{m_i}+C$$

$$N(\mathcal{A}\backslash\{i\})=\sum_{j=1}^{M-1}\left(N(\mathcal{A}\backslash\{i,j\})\right)^{m_j}+C$$

$$\ldots\ldots$$

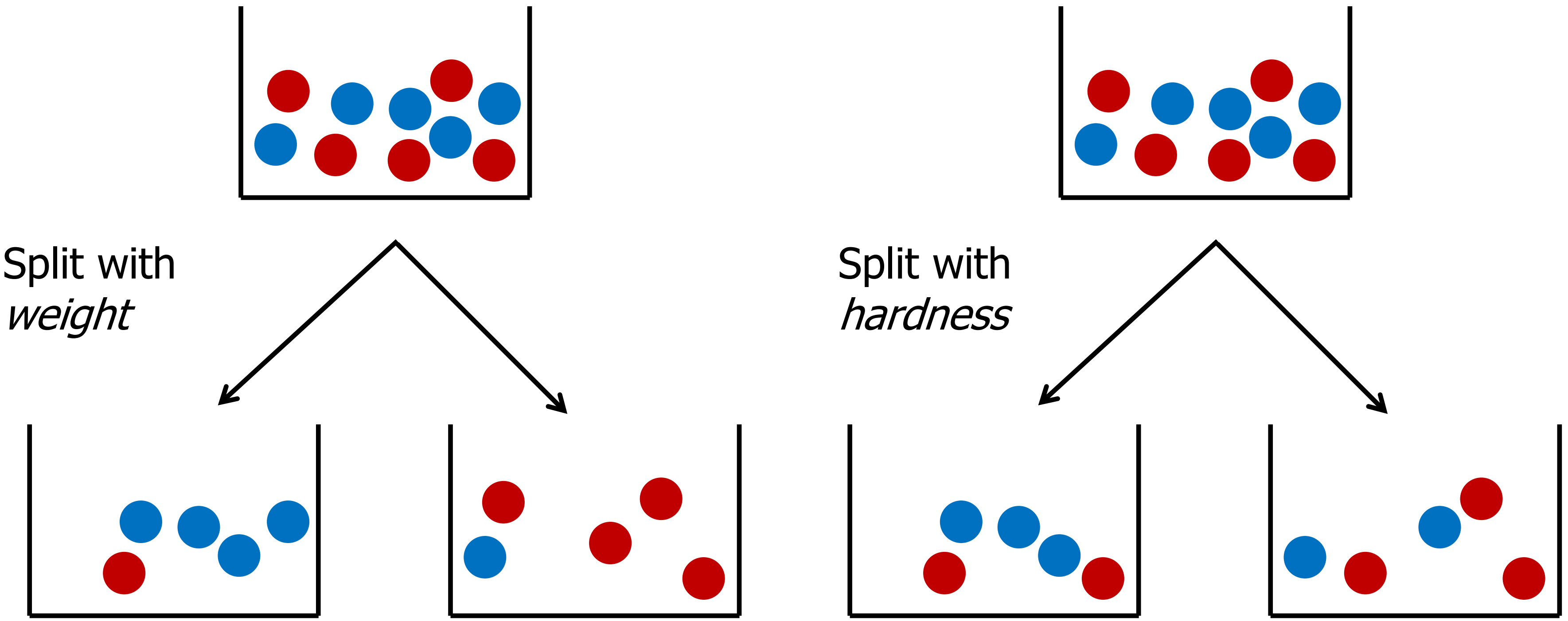
$$N(\{k\})=C^{m_k}+C$$

A Greedy Idea

- Grow a tree from root to leaves; do not backtrack
- Let's first choose a good attribute for the root
- Choose to split the root or not
  - If split: each branch grows into a subtree by recursion
  - Else: this is a leaf node, make a decision

- What is a good attribute to choose?
  - The one that better classifies training examples
- How to decide splitting or not?
  - Purity of class labels of training examples falling in this node
- How to make decision at a leaf node?
  - Majority vote of the training examples falling in this node

Choosing the Best Attribute



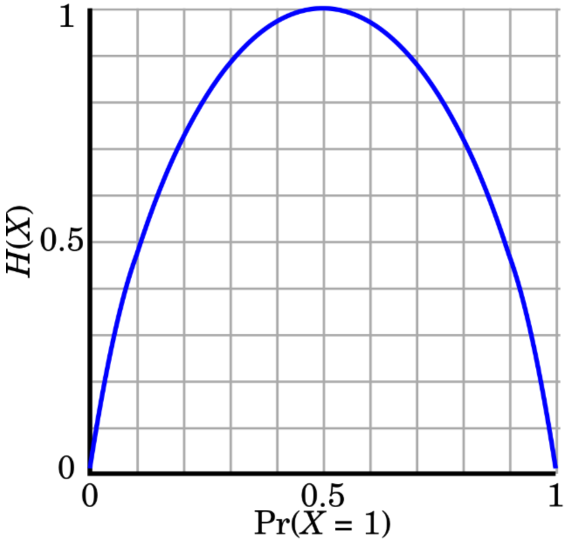
- Which attribute is better?
  - Splitting with *weight* better classifies the balls

Entropy

- Entropy $H(X)$ is a measure of (im)purity of a random variable
- For categorical or discrete variables with C values

$$H = -\sum_{i=1}^{C} p_i \log_2 p_i$$

e.g., if $C = 2$, $H = -p\log_2 p - (1-p)\log_2(1-p)$



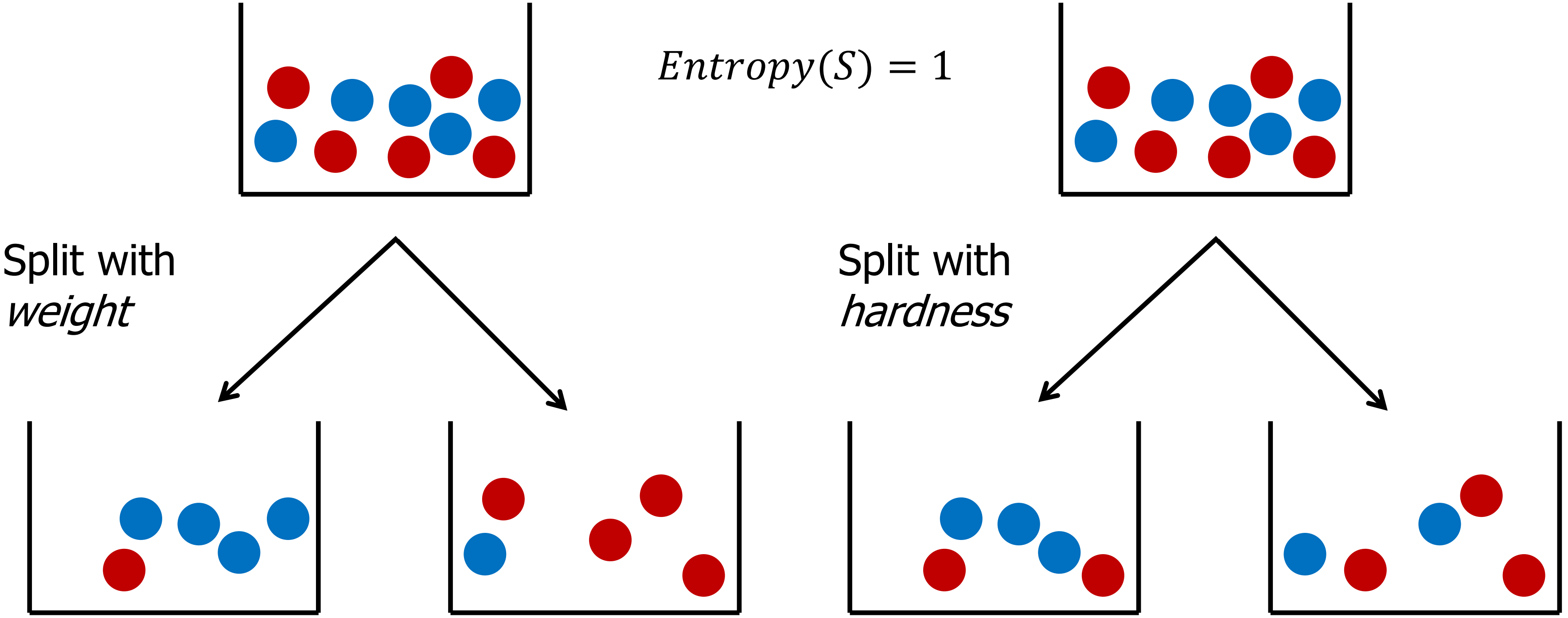
- It quantifies the number of bits needed to encode the variable

Information Gain

- Given a collection of training examples S, denote the entropy of their class labels as $Entropy(S)$
- If we split them according to attribute A into subsets $\{S_v\}$, where $v \in Values(A)$
- Each subset has its class label entropy as $Entropy(S_v)$
- Information gain: the reduction of entropy
  - The information gained for classifying the training examples through this split

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Information Gain Illustration



$Entropy(S) = 1$

$Entropy(S_1) = 0.72$ $Entropy(S_2) = 0.72$ $Entropy(S_1) = 0.97$ $Entropy(S_2) = 0.97$

$Gain(S, weight) = 0.28$

$Gain(S, hardness) = 0.03$

Other Ways to Select Attributes

- Information gain = entropy reduction

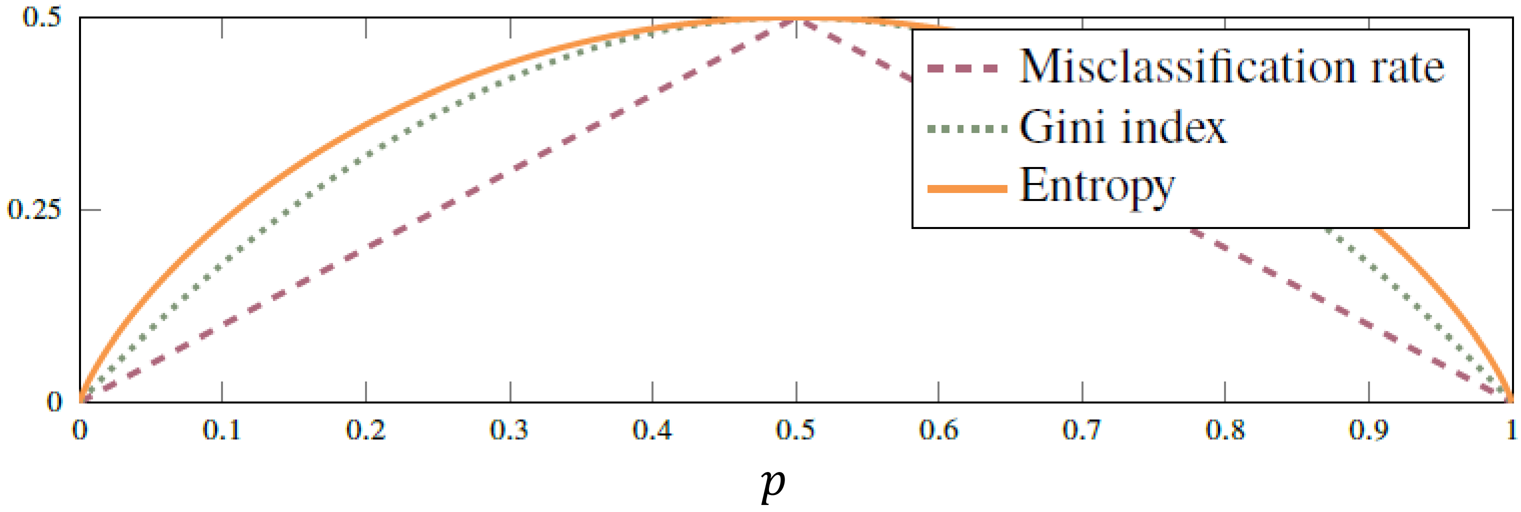
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- There are other ways that only evaluate some post-splitting statistic, e.g., weighted average of Q of all child nodes, and choose the attribute that minimizes it.
- Let $C = \#\text{classes}$, p_i be the percentage of examples in a node that belong to the i-th class, Q can be defined as
  - Misclassification rate: $Q = 1 - \max_{i \in \{1,\ldots,C\}} p_i$
  - Gini index: $Q = 1 - \sum_{i=1}^{C} p_i^2$
  - Entropy: $Q = -\sum_{i=1}^{C} p_i \log_2 p_i$

Comparing the Three Criteria

- If C=2 (binary classification)
  - Misclassification rate: $Q = 1 - \max(p, 1 - p)$
  - Gini index: $Q = 2p(1 - p)$
  - Entropy: $Q = -p\log_2 p - (1 - p)\log_2(1 - p)$

(Figure 2.10 in LWLS): entropy is scaled by 0.5



- Misclassification rate does not favor pure nodes as entropy and Gini index do

The ID3 Algorithm

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

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree   Stop growing
- 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 = most common value of *Target_attribute* in *Examples*
- Otherwise Begin   Splitting criterion
  - $A \leftarrow$ 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 value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − $\{A\}$))
- End
- Return *Root*

(Table 3.1 in Mitchell)

Hypothesis Space

- Hypothesis space is the function space that a machine learning model explores
  - Learning can be viewed as a function search problem
- ID3's hypothesis space is the set of all possible trees, which is the complete space of categorical functions of the attributes
  - Because any such function can be expressed as a tree
- Top-down greedy search without backtracking: converging to locally optimal solutions
- Maintains a single current hypothesis through the search
- Uses all available training examples at each step of the search; less sensitive to errors in individual training examples

Inductive Bias

- Inductive bias is the set of assumptions based on which the machine learning model learns from training data and makes predictions on unseen data deductively
  - Linear regression: a linear mapping from $\boldsymbol{x}$ to y
  - Nearest neighbor: labels of a test example is most similar to that of its nearest neighbor
- A machine learning model cannot learn anything without an inductive bias (i.e., assumptions)
- What is the inductive bias of ID3?
  - Remember that ID3's hypothesis space contains all possible trees, but it explores them from simple to complex
  - 1) Prefers shorter trees over longer ones
  - 2) Prefers trees that place high information gain attributes close to the root over those that do not

Restriction vs. Preference

- Inductive bias may be presented in the hypothesis space, the search strategy, or both
- Restriction biases: restrict the hypothesis space
  - E.g., linear regression restricts the space to linear functions
  - It may exclude the target function from the search
- Preference biases: set search preferences in the complete hypothesis space
  - E.g., ID3 prefers shorter trees among all possible trees
  - The hypothesis space always contains the target function
- Both: restrict the hypothesis space *and* set search preferences
  - E.g., linear regression with L1 regularization on weights

Occam's Razor

Prefer the simplest hypothesis that fits the data.

---- William of Occam, ~1320

- This is a philosophy that many scientists believe

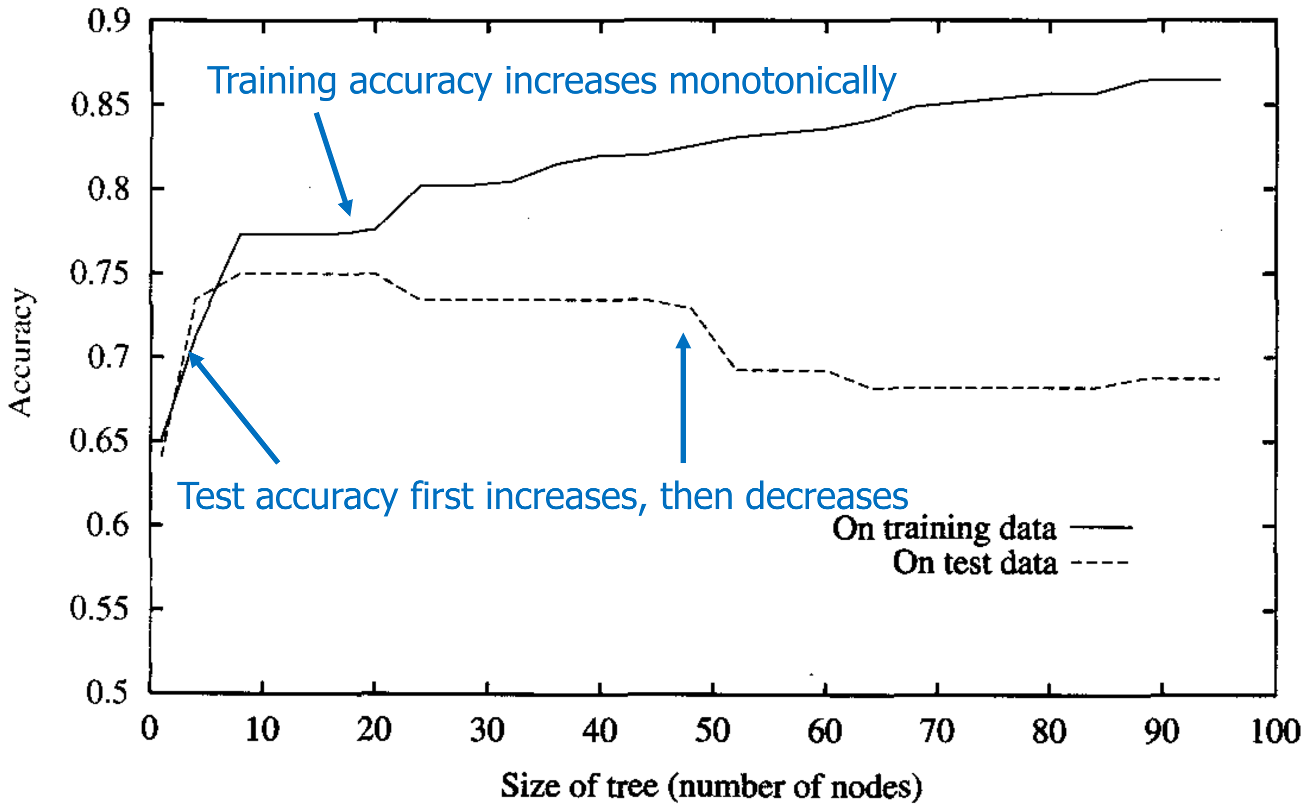
Everything should be made as simple as possible, but not simpler.

---- Albert Einstein

- Whether it is true is debatable
- Its interpretation can also be vague

Overfitting Issue

- ID3 grows the tree to perfectly classify training examples
  - Like NN, it has zero training error, hence likely overfit
  - Overfitting may be due to data noise, or coincidental regularities



(Figure 3.6 in Mitchell)

Tree Pruning

- Two ways to avoid overfitting
  - Stop growing tree earlier
  - Grow to overfit, then post-prune the tree
- Pruning: use a validation set
  - Reduce error pruning
    - Iteratively prune the node (i.e., remove subtrees + change to leaf node) that results in the most increase of validation accuracy
  - Rule post-pruning
    - Covert tree into an equivalent set of rules (i.e., paths from root to leaf nodes)

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = High) \\ \text{THEN} & PlayTennis = No \end{array}$$

    - Prune each rule by removing any preconditions that results in a higher validation accuracy
    - Sort the pruned rules by their validation accuracy, and use them in this order during classification

Handling Missing Values

- Some decision trees (e.g., C4.5) can handle missing attributes in the data

- Assume training example $(\boldsymbol{x}, y)$ has a missing attribute x_i

- Approach 1: assign the most common value among all training examples in the node
- Approach 2: assign the most common value among all training examples in the node that share the same class label

- Approach 3: split the example into fractions that take different values on the missing attribute, following the probability of those values in the training examples in the node; Then pass these fractions to child nodes.
  - E.g., 0.6 of the example has $x_i = heavy$, 0.4 of the example has $x_i = light$

Continuous-Valued Attributes

- Threshold the attribute to split it into two halves
- Feature space is divided into rectangles, each corresponding to a leaf node

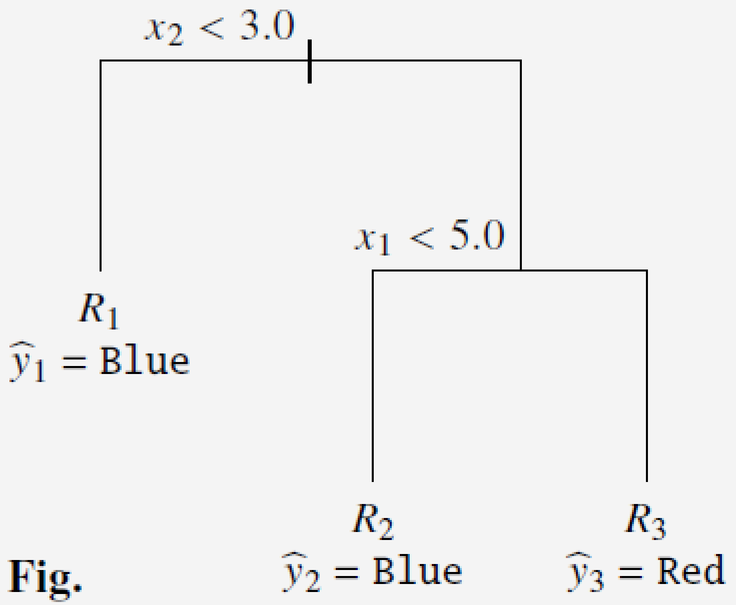


Fig. 2.6

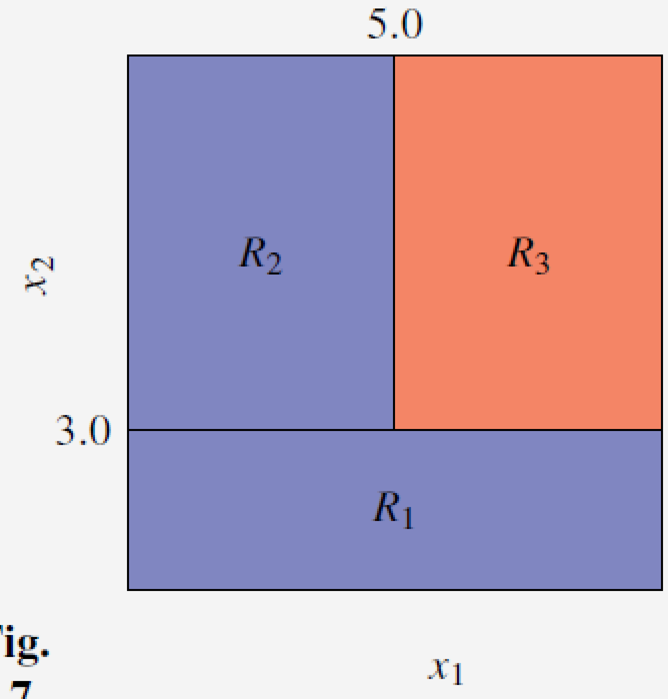


Fig. 2.7

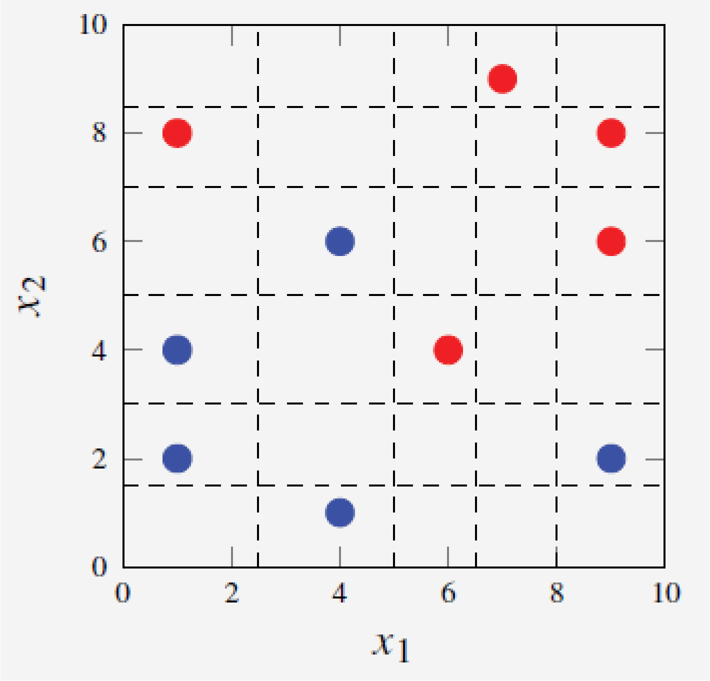
(Figures 2.6 and 2.7 in LWLS)

Continuous-Valued Attributes

- For an attribute, how to choose the splitting threshold?
  - Choose threshold that minimizes misclassification rate, Gini index, or entropy, or maximizes information gain!

- There are infinitely many possible values for each threshold. Try all?
  - No, only needs to try mid points between adjacent data points
  - Why?

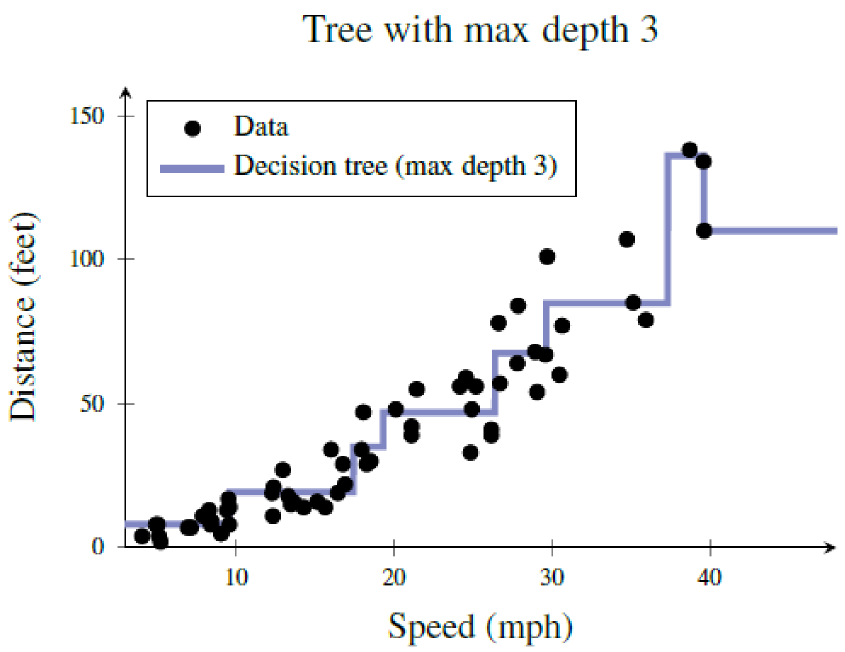
- Compare attributes using its best threshold



(Fig. 2.8 in LWLS)

Regression Tree

- target y is a numerical variable
- At a leaf node, prediction is made by taking the average of the target values of training examples in that leaf
- What kind of function does a regression tree represent?
  - Piecewise constant



How to split a node?

- Try different attributes to split, and choose the one that minimizes the sum of squared errors between the prediction and the ground-truth of training examples

$$\sum_{b \in \text{all branches}} \sum_{i \in b} \left(y^{(i)} - \overline{y_b}\right)^2$$

- For categorical attributes, #branches = #values
- For numerical attributes, #branches = 2 by thresholding
  - Threshold is searched to minimize the sum of squared errors

- All other discussions follow those for classification trees

Summary

- Decision trees sort test examples from root to leaf nodes
- They grow in a greedy fashion to fit training data
- ID3 searches a complete hypothesis space
- Inductive bias includes a preference for smaller trees and trees that put important attributes closer to the root
- Easy to overfit; Post-pruning is an effective solution