Decision Tree Learning

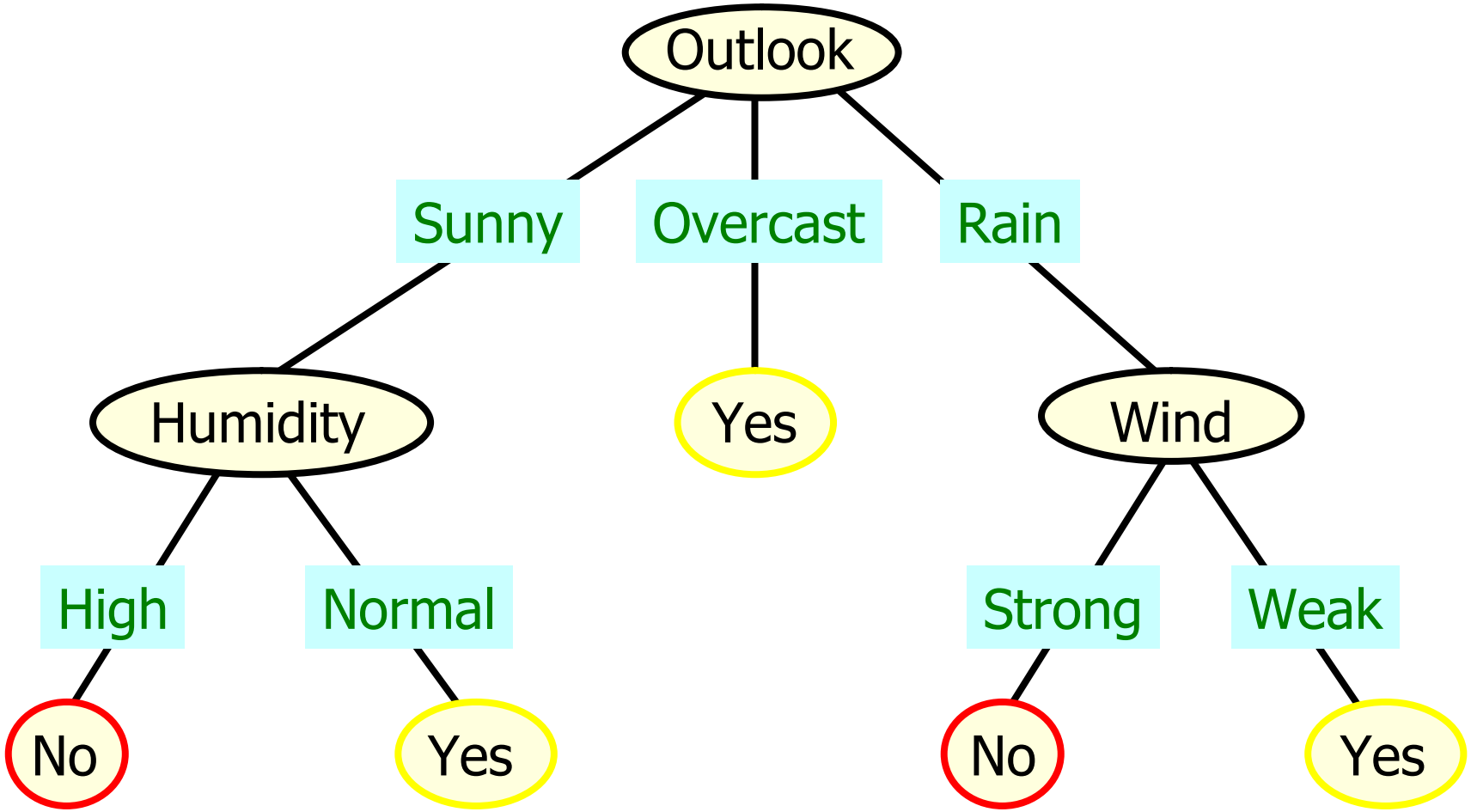
鮑興國 Ph.D.

National Taiwan University of
Science and Technology

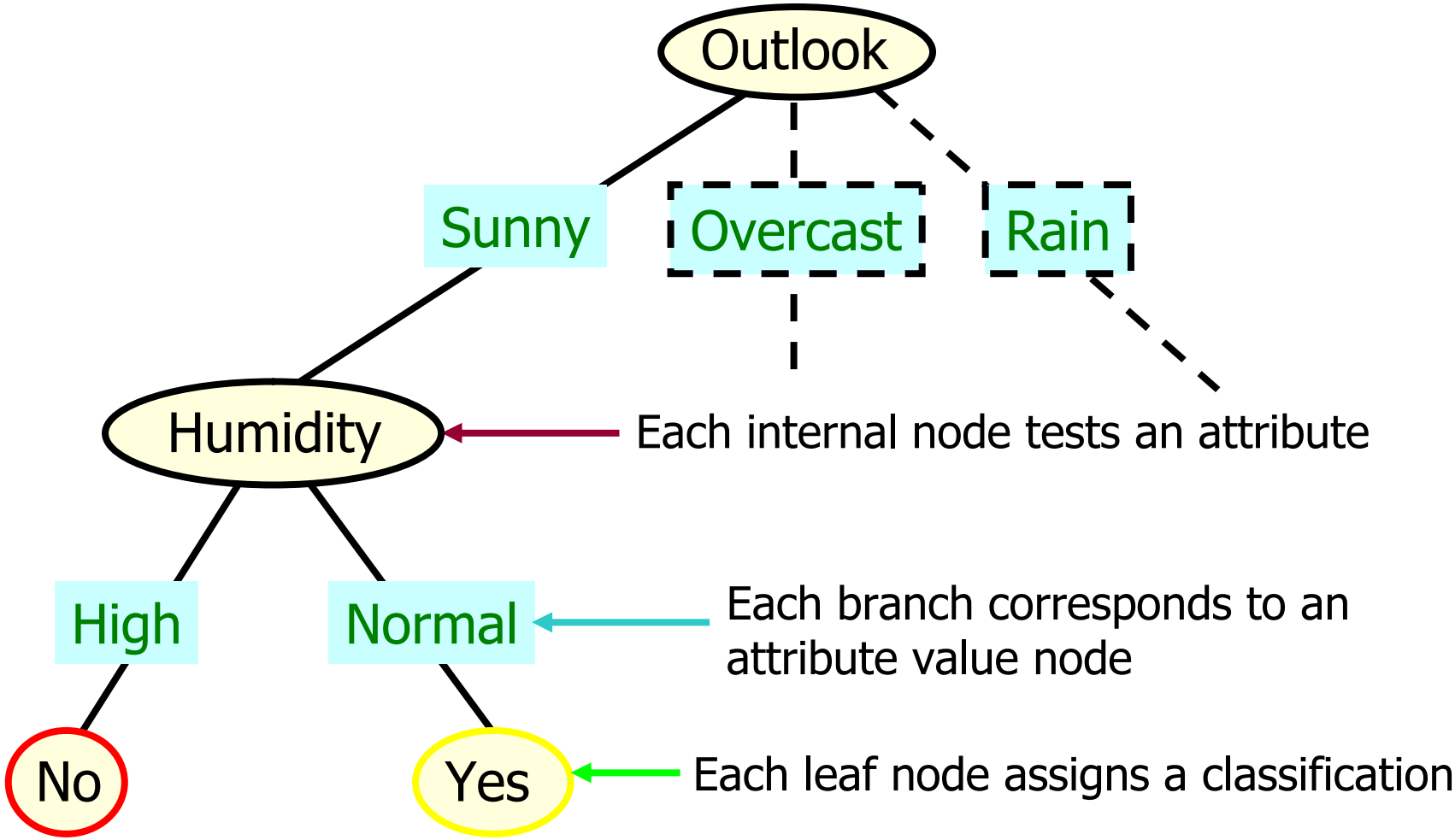
Outline

- Decision tree representation
- ID3 learning algorithm
- Entropy, information gain
- Overfitting

Decision Tree for PlayTennis

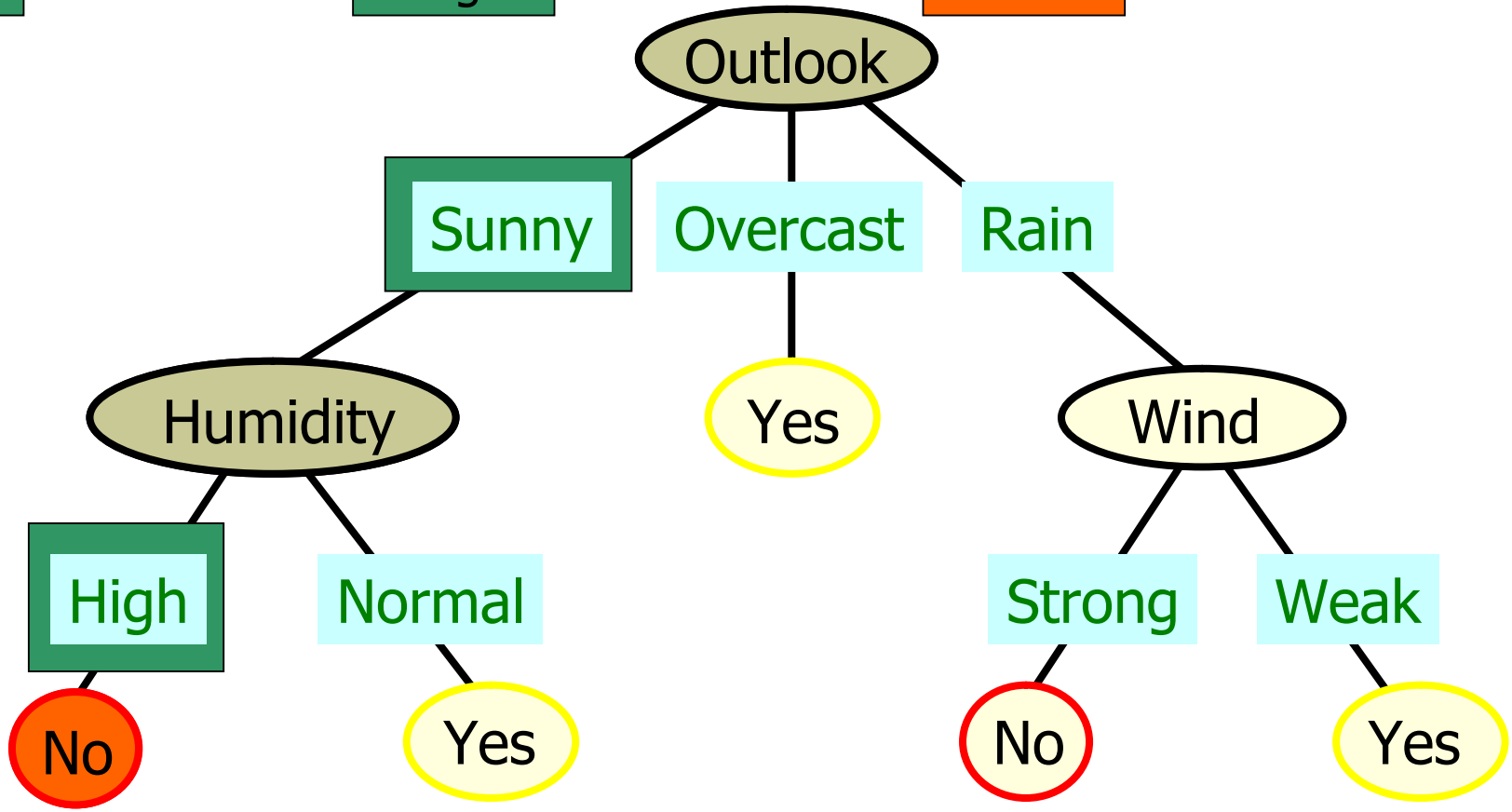


Decision Tree for PlayTennis



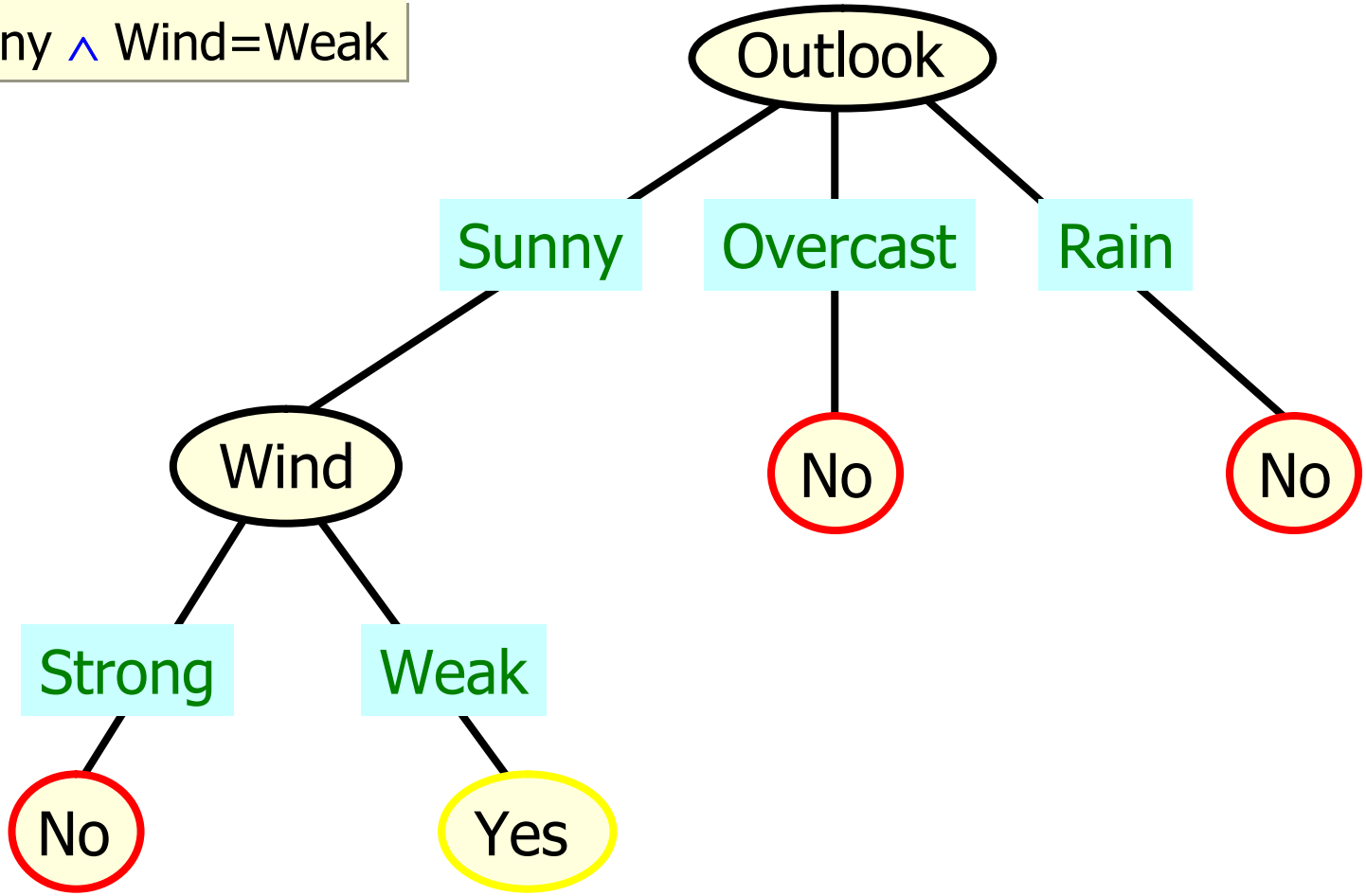
Decision Tree for PlayTennis

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No



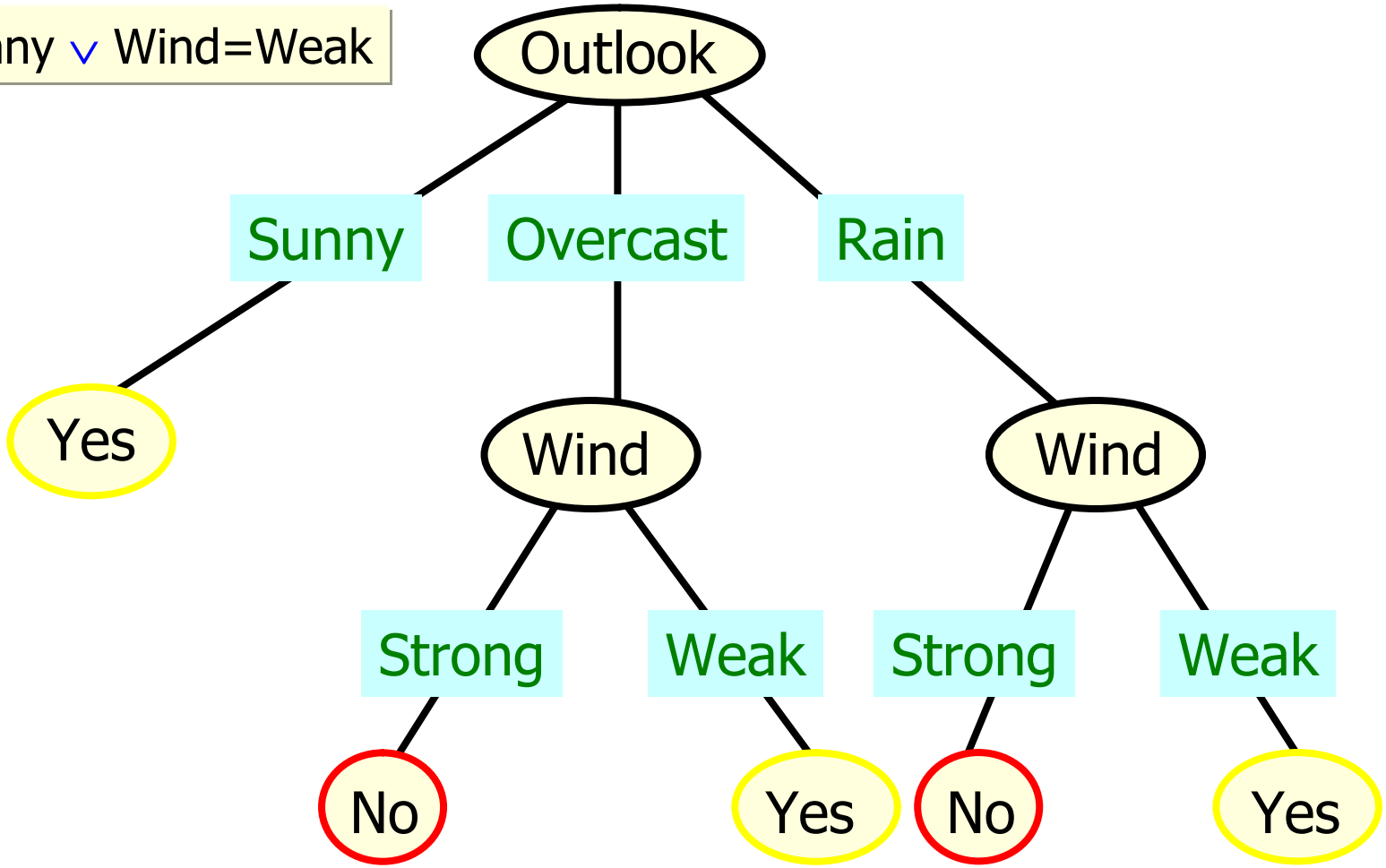
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



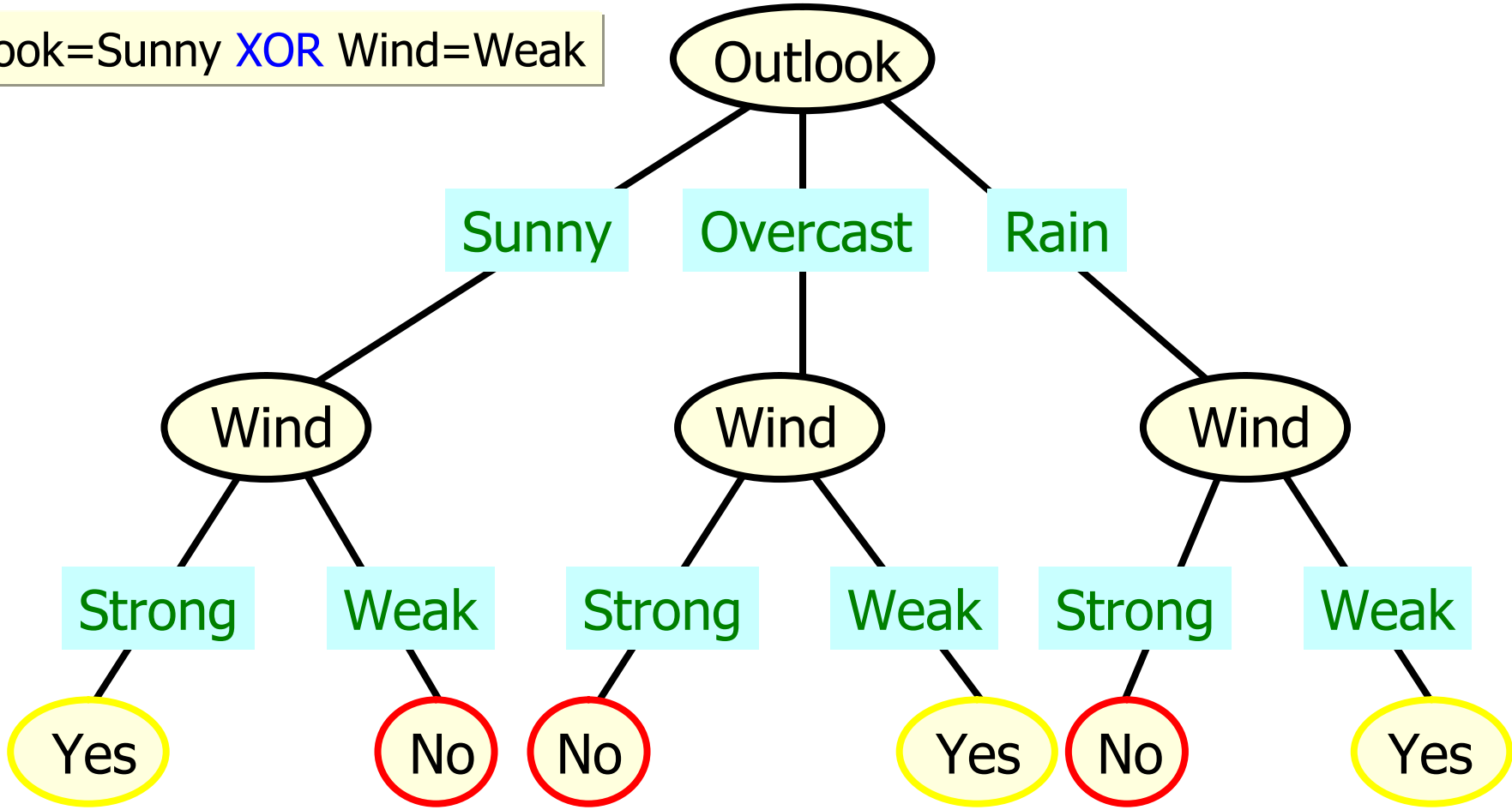
Decision Tree for Disjunction

Outlook=Sunny ∨ Wind=Weak



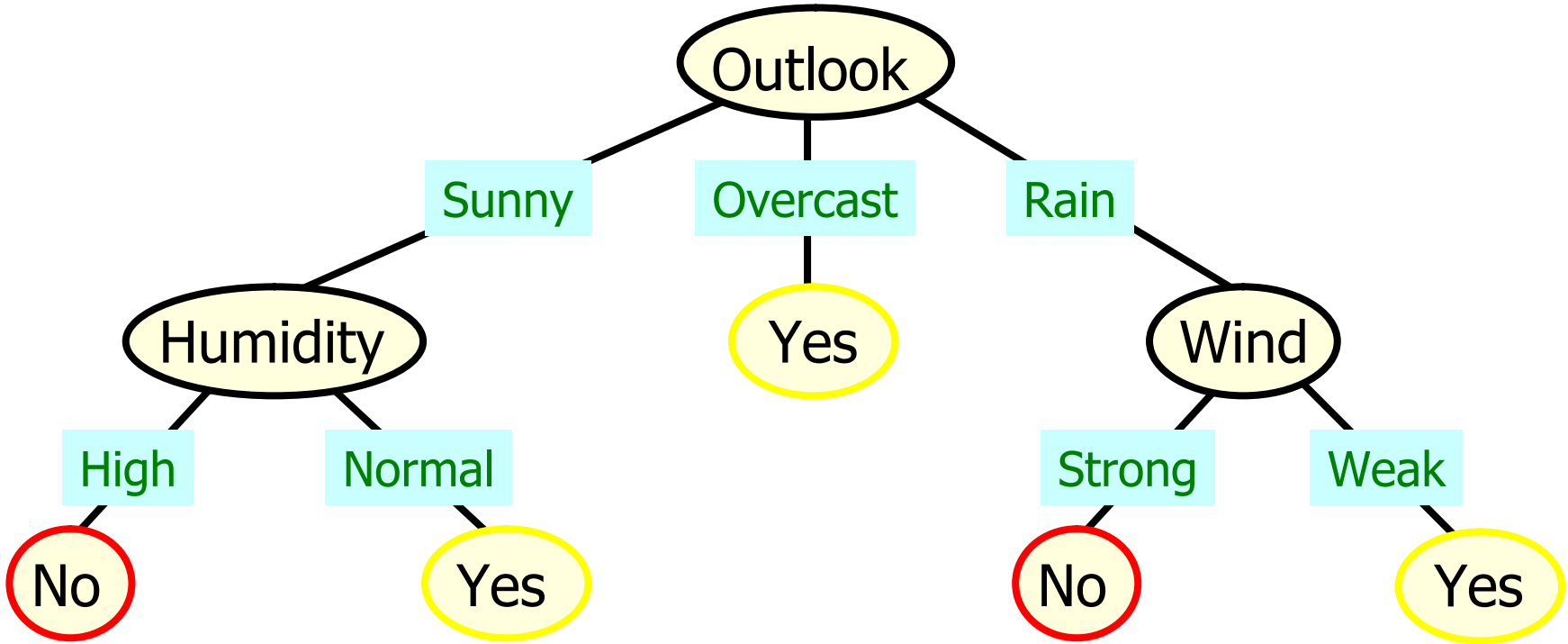
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

- decision trees represent disjunctions of conjunctions



(Outlook=Sunny ∧ Humidity=Normal)
∨ (Outlook=Overcast)
∨ (Outlook=Rain ∧ Wind=Weak)

When to consider Decision Trees

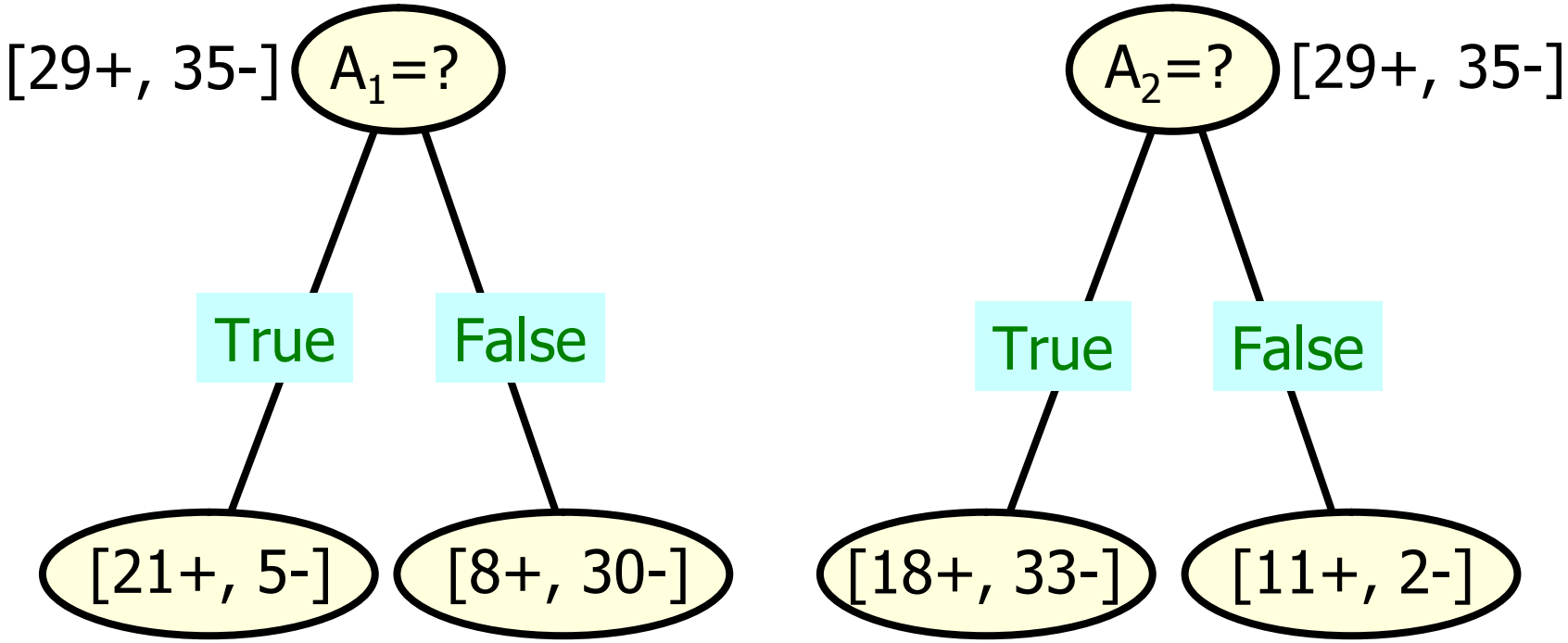
- Instances describable by attribute-value pairs
- Target function is *discrete valued* categorical
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Missing attribute values
- Examples:
  - Medical diagnosis
  - Credit risk analysis
  - Object classification for robot manipulator (Tan, 1993)

Top-Down Induction of Decision Trees ID3

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value v_i of A
    4. create new descendant (tree branch) corresponding to the test $A = v_i$
    5. Sort training examples to leaf node according to the attribute value of the branch
    6. If all those training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.
    7. If no training examples satisfying $A = v_i$ stop, with label assigned to the majority target attribute
    8. Else iterate over new leaf nodes.

Will the attributes allowed to be reused?

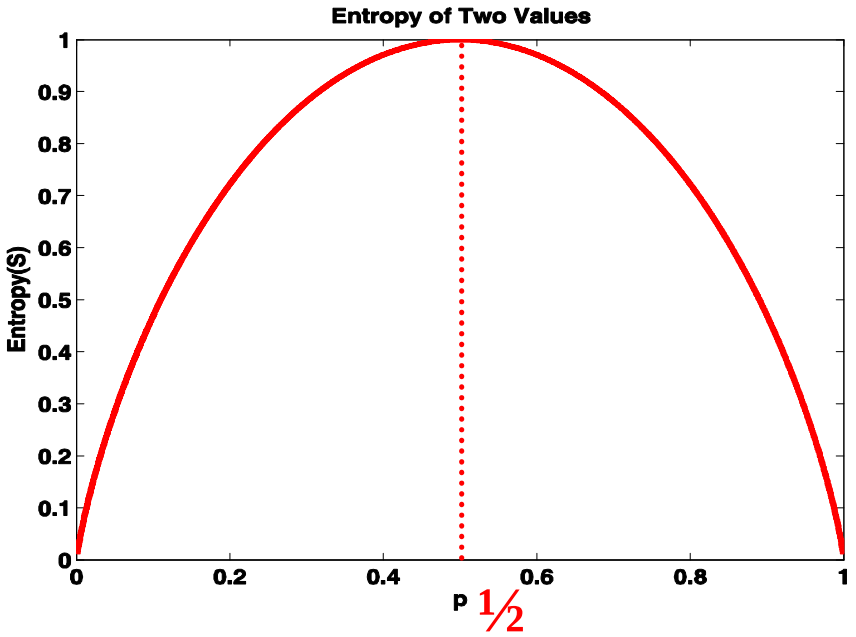
Which Attribute is "best"?



Entropy

- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$



Entropy

- Entropy(S)= expected number of bits needed to encode class (+ or –) of randomly drawn members of S (under the optimal, shortest length-code)

Why?

- Information theory optimal length code assign $-\log_2 p$ bits to messages having probability p.
- So the expected number of bits to encode (+ or –) of random member of S:

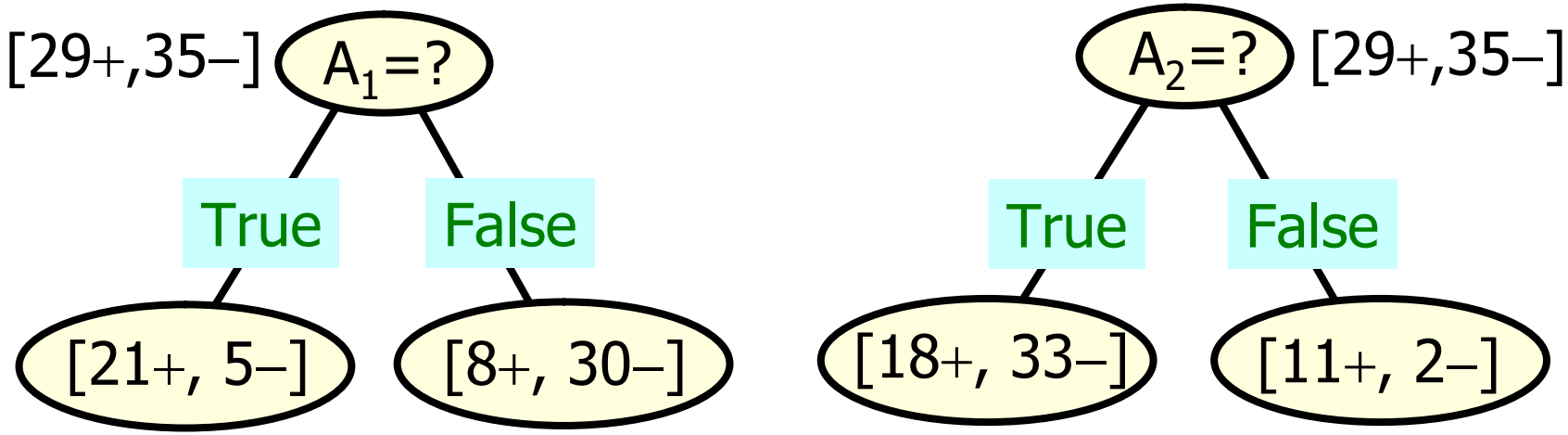
$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

- Gain(S, A): expected reduction in entropy due to sorting S on attribute A
- Gain(S, A) = Entropy(S) – $\sum_{v \in \text{values}(A)} |S_v|/|S|$ Entropy(S_v)
- Entropy([29+,35–])

$$= -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$

$$= 0.99$$



Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S, A_1) = Entropy(S)

–26/64×Entropy([21+,5–])

–38/64×Entropy([8+,30–])

= 0.27

Entropy([18+,33-]) = 0.94

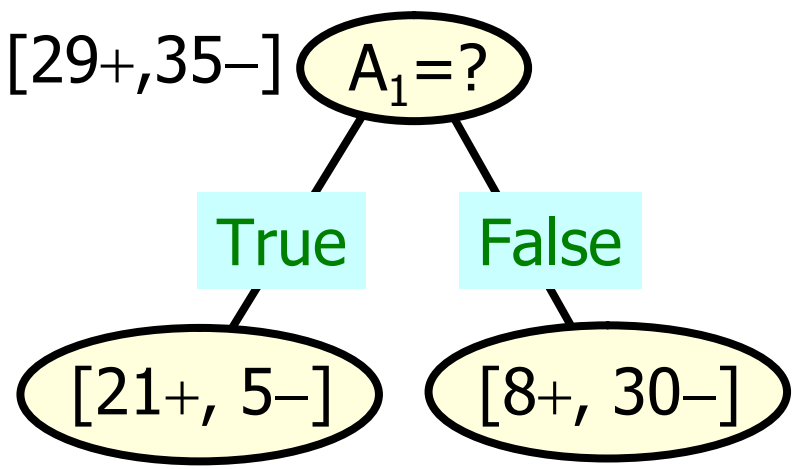
Entropy([11+,2-]) = 0.62

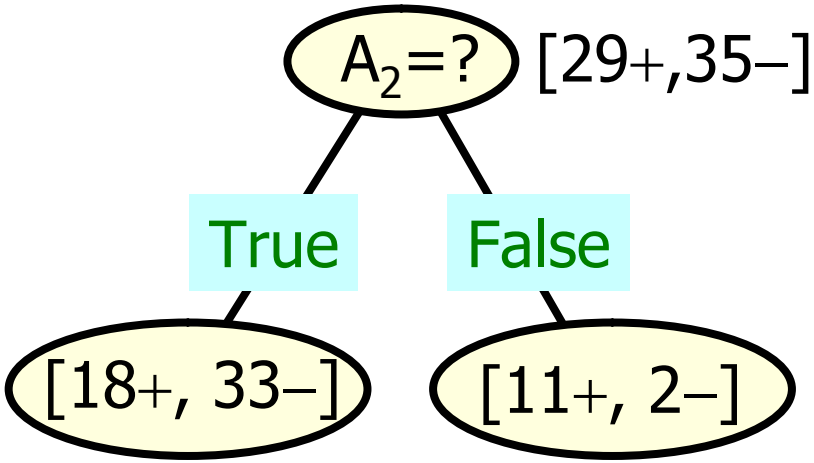
Gain(S, A_2) = Entropy(S)

–51/64×Entropy([18+,33–])

–13/64×Entropy([11+,2–])

= 0.12

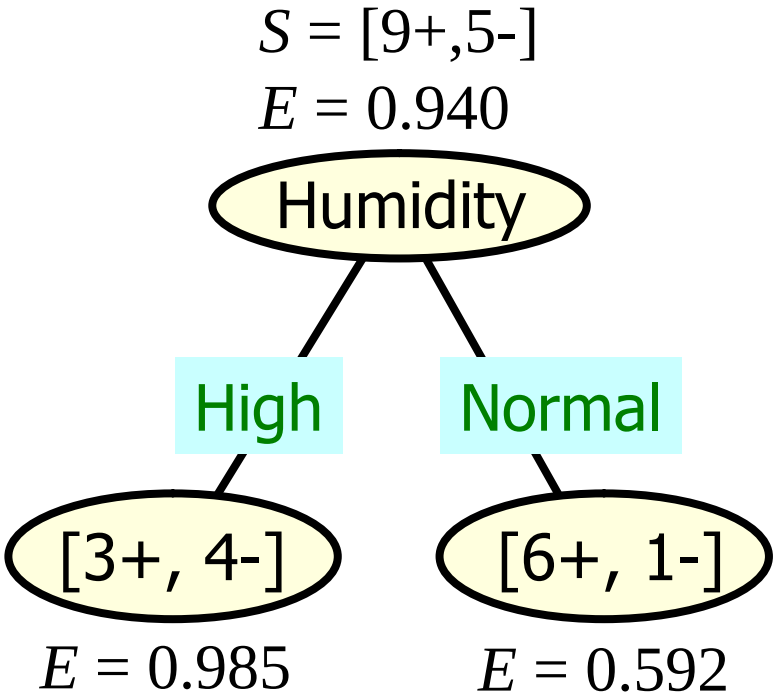


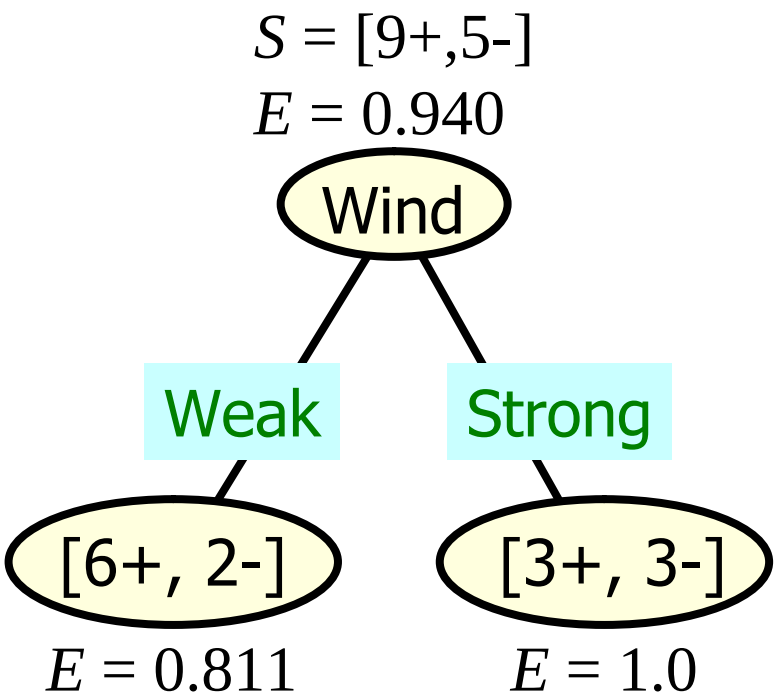


Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

Selecting the Next Attribute



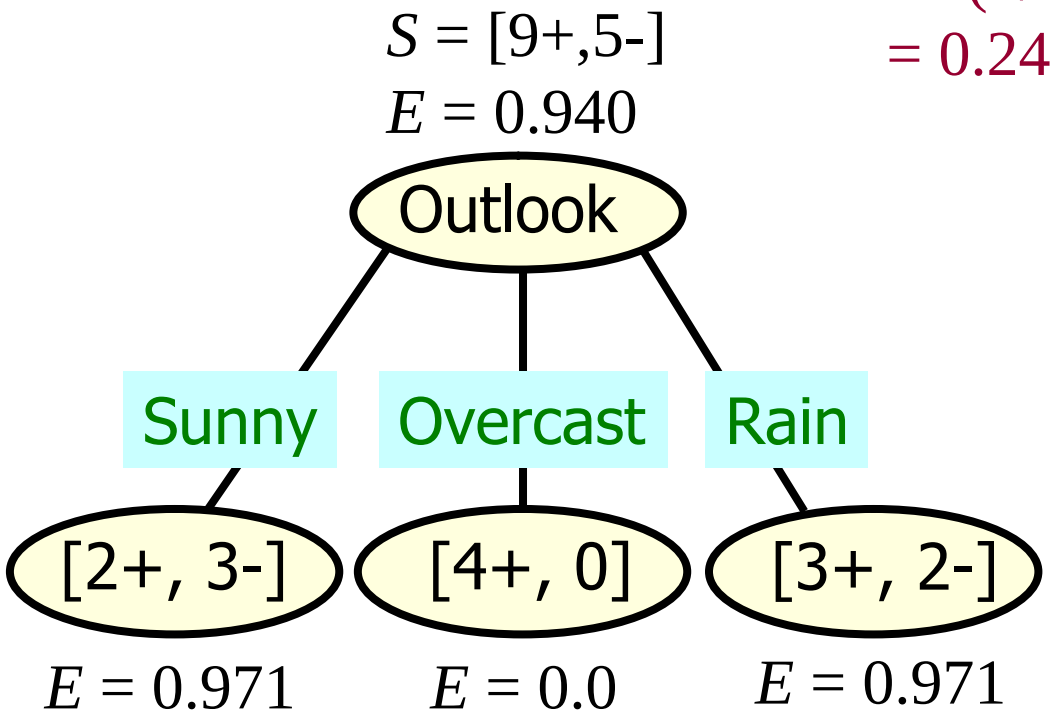


Gain(S, Humidity)
= 0.940 – (7/14)×0.985
– (7/14)×0.592
= 0.151

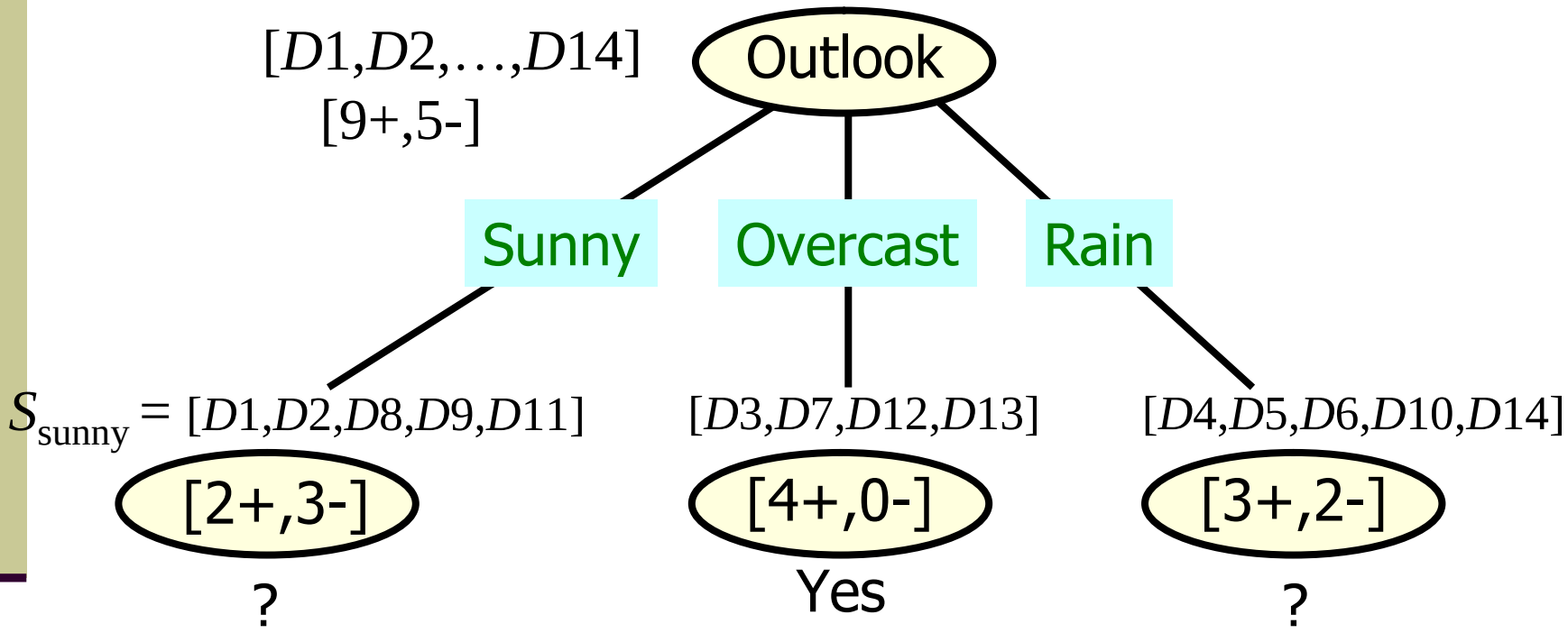
Gain(S, Wind)
= 0.940 – (8/14)×0.811
– (6/14)×1.0
= 0.048

Selecting the Next Attribute

Gain(*S*, Outlook)
= 0.940 – (5/14)×0.971
– (4/14)×0.0 – (5/14)×0.0971
= 0.247



ID3 Algorithm

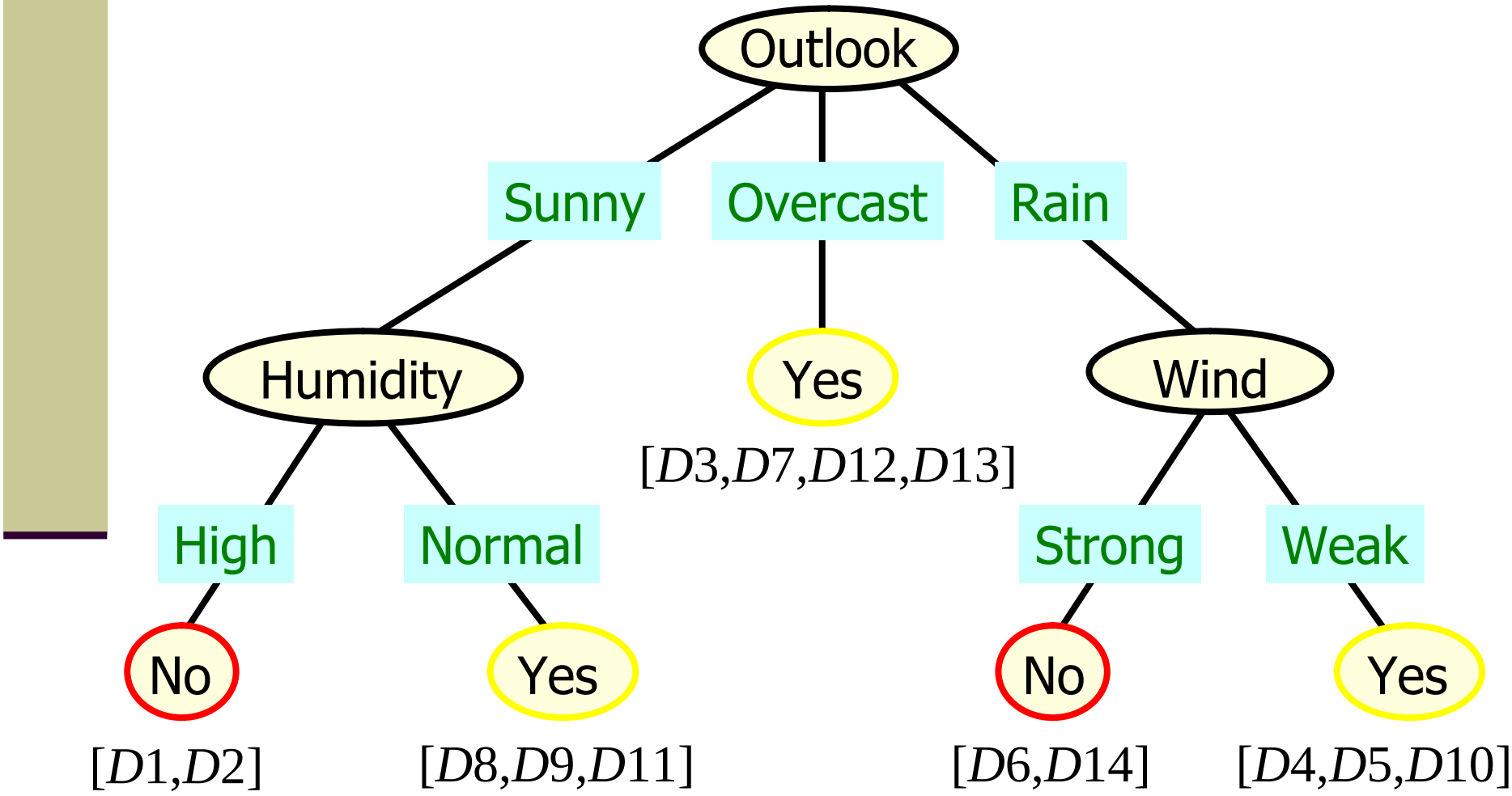


Gain(S_{sunny} , Humidity) = 0.970 – (3/5)0.0 – 2/5(0.0) = 0.970

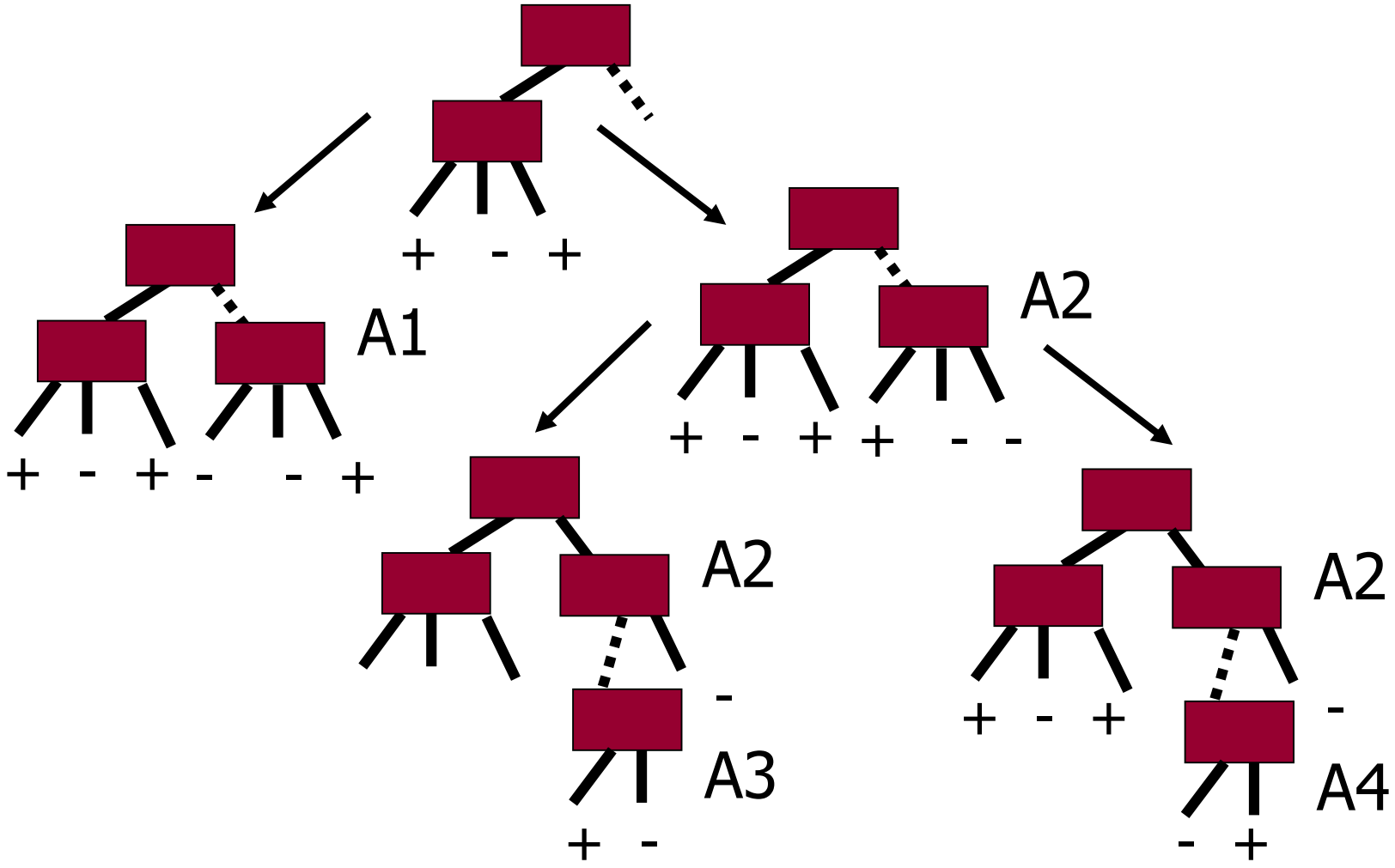
Gain(S_{sunny} , Temp.) = 0.970 – (2/5)0 – 2/5(1.0) – (1/5)0 = 0.570

Gain(S_{sunny} , Wind) = 0.970 = – (2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



Hypothesis Space Search ID3



Hypothesis Space Search ID3

- Hypothesis space is complete!
  - Target function surely in there…
- Outputs a single hypothesis
- No backtracking on selected attributes (greedy search)
  - Local minimal (suboptimal splits)
- Statistically-based search choices
  - Robust to noisy data
- Inductive bias (search bias)
  - Prefer shorter trees over longer ones
  - Place highest information gain attributes closest to the root

Inductive Bias in ID3

- H is the power set of instances X
  - Unbiased ?
- Preference for short trees, and for those with high information gain attributes near the root
  - BFS-ID3 vs ID3
- Bias is a *preference* for some hypotheses, rather than a *restriction* of the hypothesis space H
- Occam's razor: prefer the shortest (simplest) hypothesis that fits the data
  - "plurality should not be posited without necessity"

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

Argument opposed:

- There are many ways to define small sets of hypotheses (notion of coding $length(X) = -\log_2 P(X)$, Minimum Description Length…)
- E.g. All trees with a prime number of nodes that use attributes beginning with "Z"
- What is so special about small sets based on *size* of hypothesis

Overfitting

Consider error of hypothesis h over

- Training data: $\text{error}_{\text{train}}(h)$
- Entire distribution D of data: $\text{error}_{D}(h)$

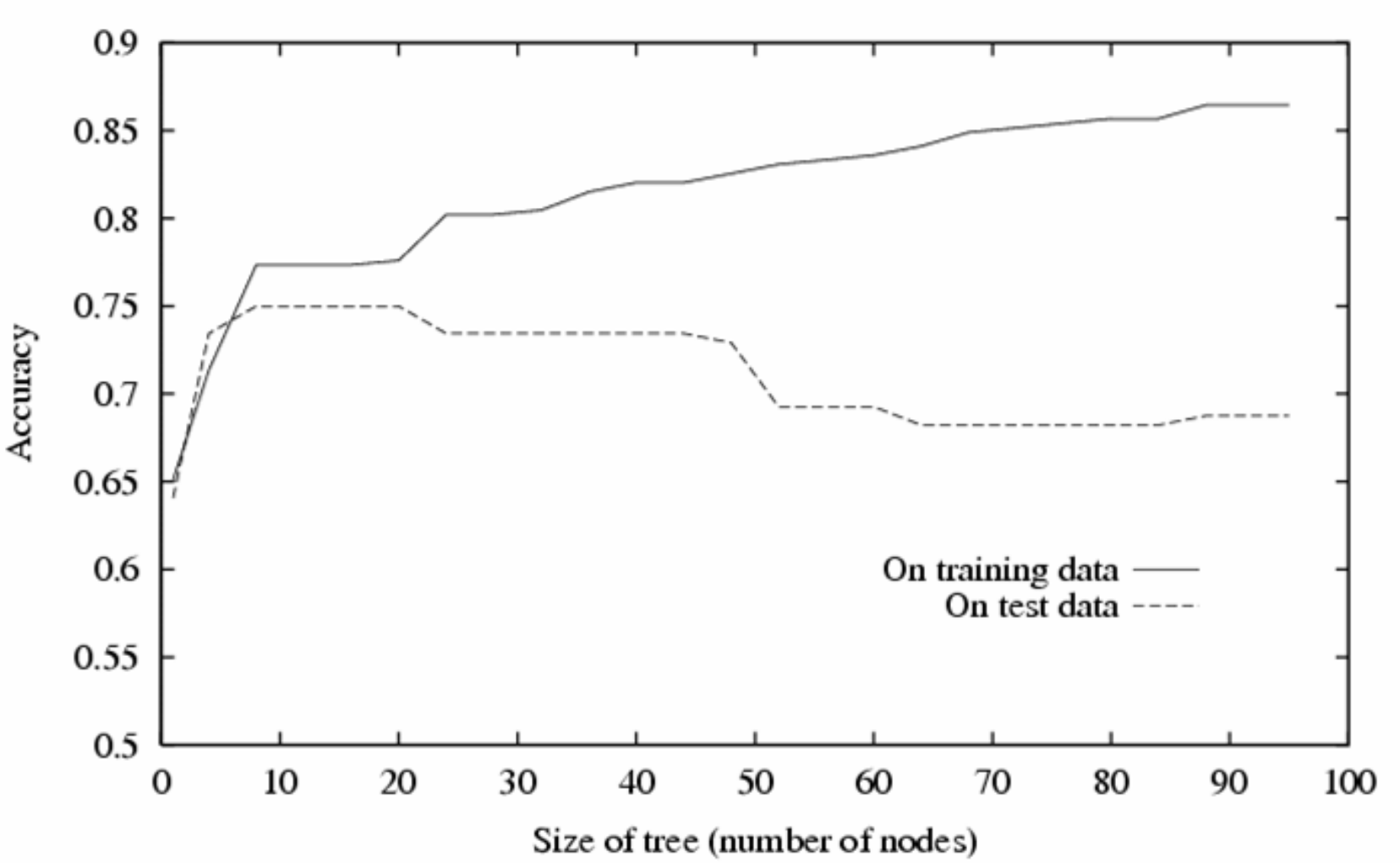
⇨ Hypothesis $h \in H$ *overfits* training data if there is an alternative hypothesis $h' \in H$ such that

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

and

$$\text{error}_{D}(h) > \text{error}_{D}(h')$$

Overfitting in Decision Tree Learning



Avoid Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree then post-prune
- Minimum description length (MDL):
  Minimize:

  size(*tree*) +
  size(misclassifications(*tree*))

Reduced-Error Pruning

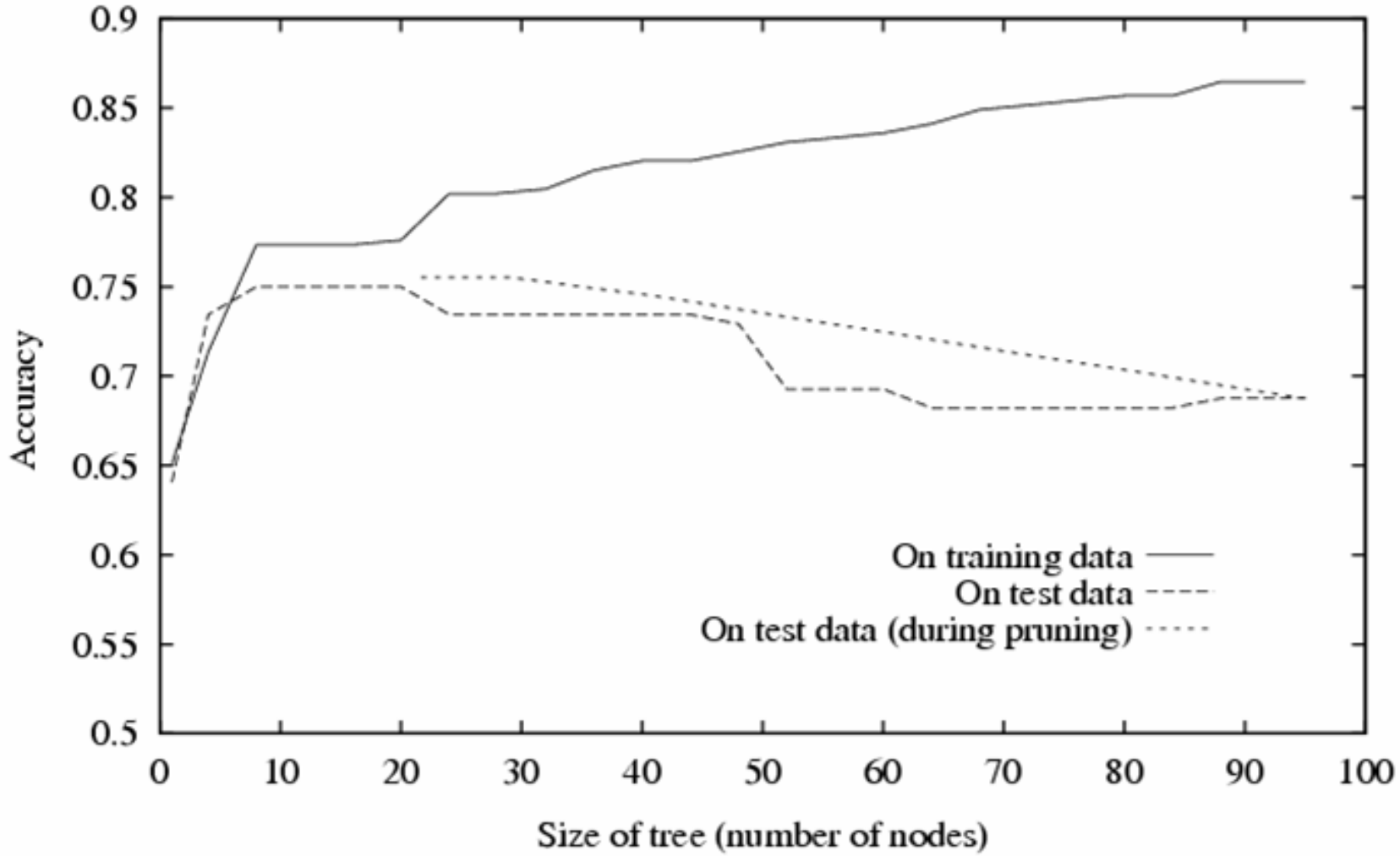
Split data into *training* and validation set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves the *validation* set accuracy

Produces smallest version of most accurate subtree

Effect of Reduced Error Pruning

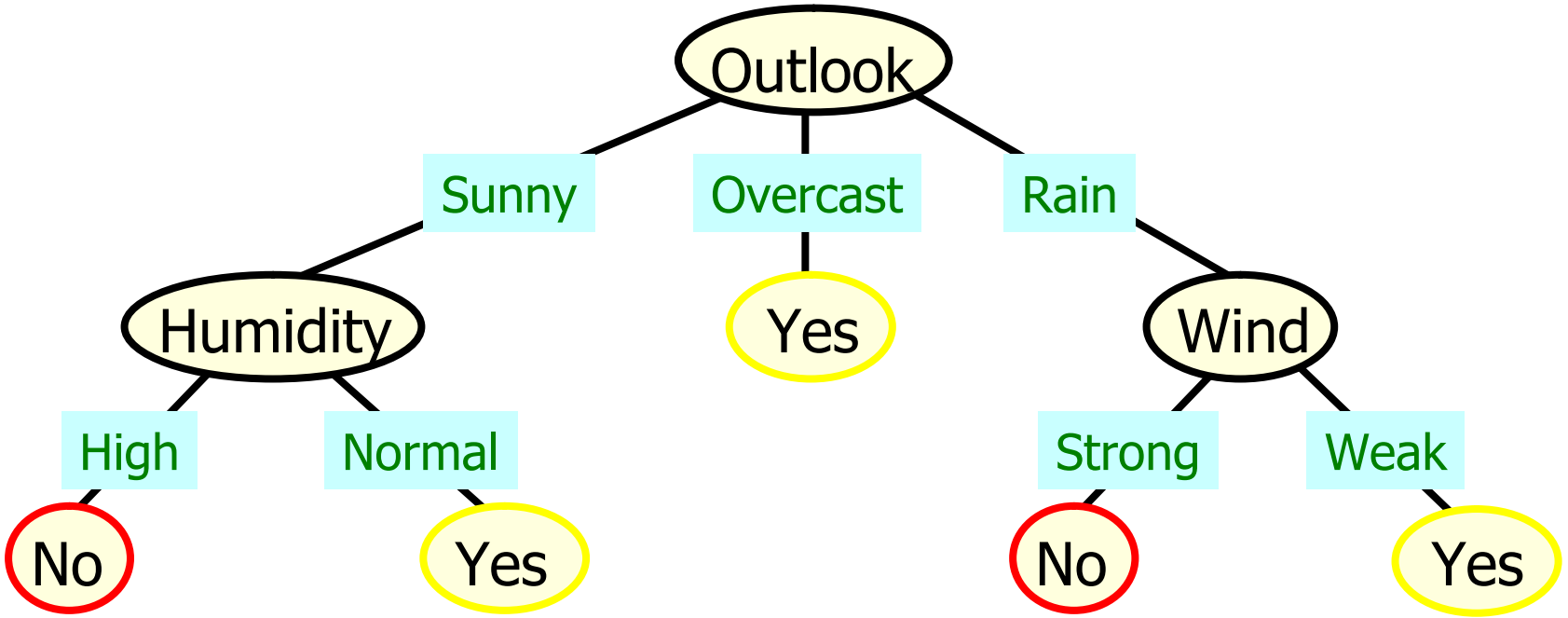


Rule-Post Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of each other
3. Sort final rules into a desired sequence to use

Method used in *C4.5*

Converting a Tree to Rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Continuous Valued Attributes

- Create a discrete attribute to test continuous
  - ⇨ Temperature = 24.5ºC
  - ⇨ (Temperature > 20.0ºC) = {true, false}

Where to set the threshold?

Temperature	15°C	18°C	19°C	22°C	24°C	27°C
PlayTennis	No	No	Yes	Yes	Yes	No

(see paper by [Fayyad, Irani 1993]

Attributes with many Values

- Problem: If an attribute has many values, maximizing InformationGain will select it.
- E.g.: Imagine using Date = 12.7.1996 as attribute perfectly splits the data into subsets of size 1
- Use GainRatio instead of information gain as criteria:
- $\text{GainRatio}(S, A) = \text{Gain}(S, A) / \text{SplitInformation}(S, A)$
- $\text{SplitInformation}(S, A) = -\sum_{i=1..c} |S_i|/|S| \log_2 |S_i|/|S|$,
  where S_i is the subset for which attribute A has the value v_i

Attributes with Cost

Consider:

- Medical diagnosis : blood test costs 1000 SEK
- Robotics: width_from_one_feet has cost 23 secs.

How to learn a consistent tree with low expected cost?

Replace *Gain* by :

$\text{Gain}^2(S, A)/\text{Cost}(A)$ [Tan, Schimmer 1990]

$2^{\text{Gain}(S, A)}-1/(\text{Cost}(A)+1)^w$, $w \in [0,1]$ [Nunez 1988]

Unknown Attribute Values

What is some examples missing values of A?

Use training example anyway sort through tree

- If node n tests A, assign most common value of A among other examples sorted to node n.
- Assign most common value of A among other examples with same target value
- Assign probability p_i to each possible value v_i of A
  - Assign fraction p_i of example to each descendant in tree

Classify new examples in the same fashion

Cross-Validation

- Estimate the accuracy of a hypothesis induced by a supervised learning algorithm
- Predict the accuracy of a hypothesis over future unseen instances
- Select the optimal hypothesis from a given set of alternative hypotheses
  - Pruning decision trees
  - Model selection
  - Feature selection
- Combining multiple classifiers (boosting)

Holdout Method

- Partition data set $D = \{(v_1, y_1),\ldots,(v_n, y_n)\}$ into *training* D_t and *validation* set $D_h = D \backslash D_t$

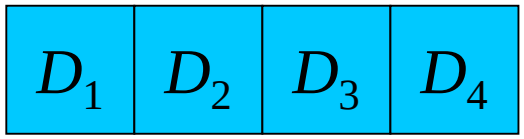
- $\text{acc}_h = 1/h \sum_{(vi, yi) \in Dh} \delta(I(D_t, v_i), y_i)$
- $I(D_t, v_i)$: output of hypothesis induced by learner I trained on data D_t for instance v_i
- $\delta(i, j) = 1$ if $i = j$ and 0 otherwise

Problems:

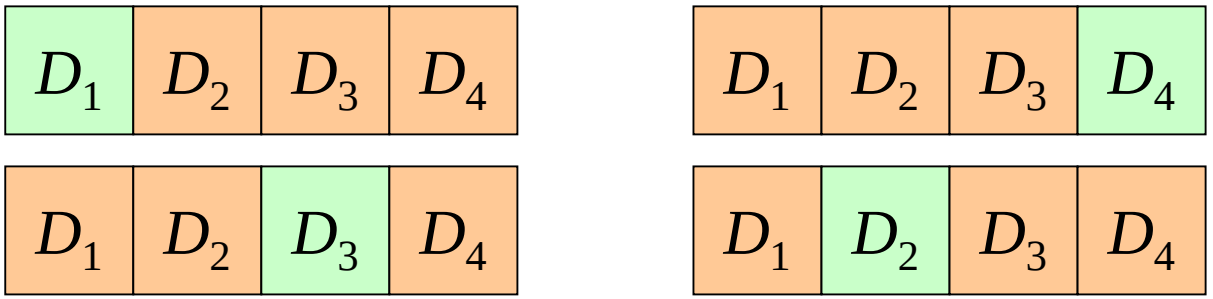
- makes insufficient use of data
- training and validation set may be correlated

Cross-Validation

- k-fold cross-validation splits the data set D into k mutually exclusive subsets $D_1, D_2, \ldots, D_k$



- Train and test the learning algorithm k times, each time it is trained on $D \backslash D_i$ and tested on D_i

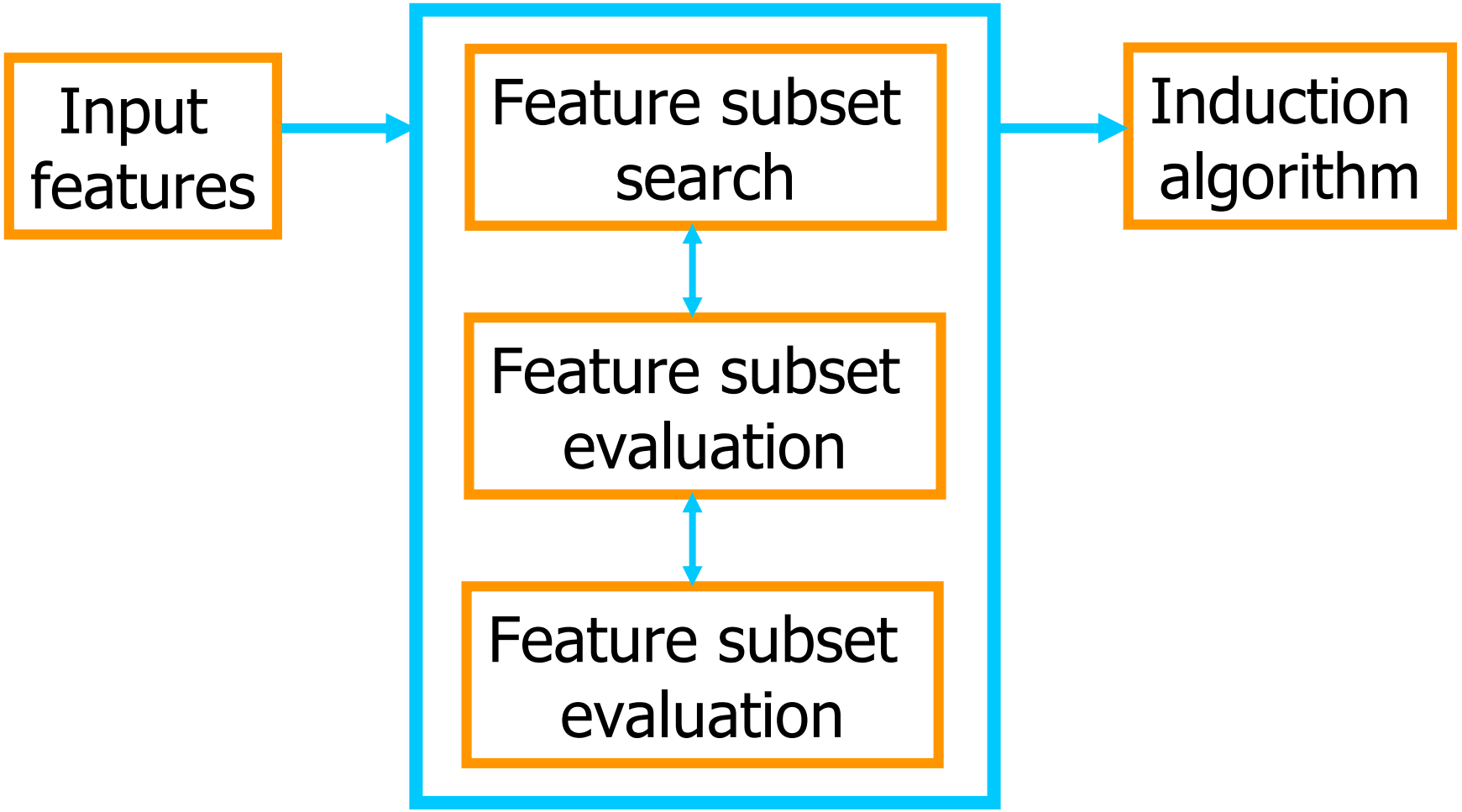


$$\text{acc}_{\text{cv}} = 1/n \sum_{(vi,\, yi) \in \text{D}} \delta(I(D \backslash D_i, v_i), y_i)$$

Cross-Validation

- Uses all the data for training and testing
- Complete k-fold cross-validation splits the dataset of size m in all (m over m/k) possible ways (choosing m/k instances out of m)
- Leave n-out cross-validation sets n instances aside for testing and uses the remaining ones for training (leave one-out is equivalent to n-fold cross-validation)
- In stratified cross-validation, the folds are stratified so that they contain approximately the same proportion of labels as the original data set

Wrapper Model



Wrapper Model

- Evaluate the accuracy of the inducer for a given subset of features by means of n–fold cross-validation
- The training data is split into n folds, and the induction algorithm is run n times. The accuracy results are averaged to produce the estimated accuracy.
- Forward elimination:

  Starts with the empty set of features and greedily adds the feature that improves the estimated accuracy at most
- Backward elimination:

  Starts with the set of all features and greedily removes features and greedily removes the worst feature

Readings

- Read Ch. 3 in Mitchell or Ch. 9 in Alpaydin on decision tree learning
- Read at least one out the following three articles
  - “A study of cross-validation and bootstrap for accuracy estimation and model selection” [Kohavi 1995]
  - “Irrelevant features and the subset selection problem” [John, Kohavi, Pfleger]
  - “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning” [Fayyad, Irani 1993]