Machine Learning

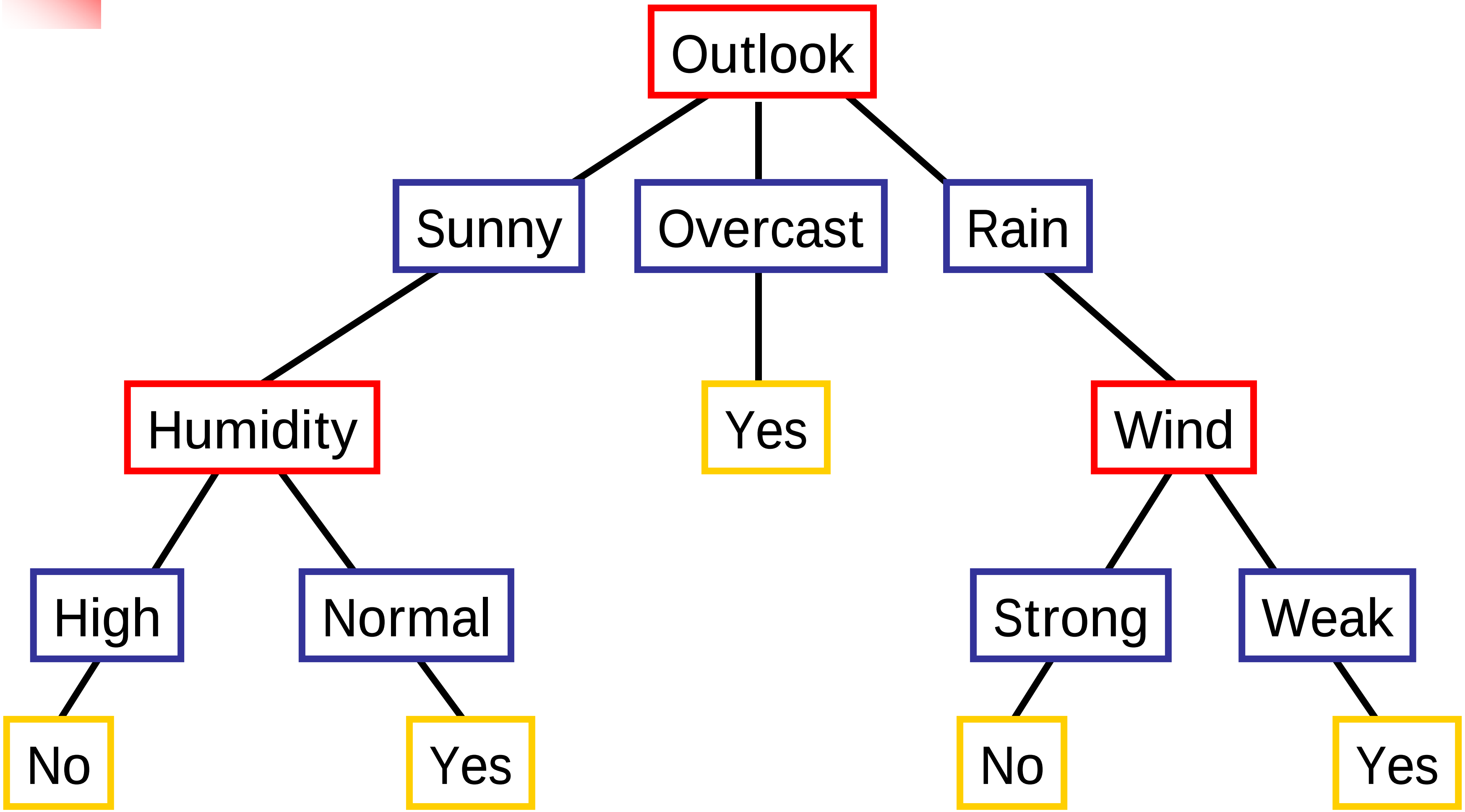
Lecture 4

Decision Tree Learning

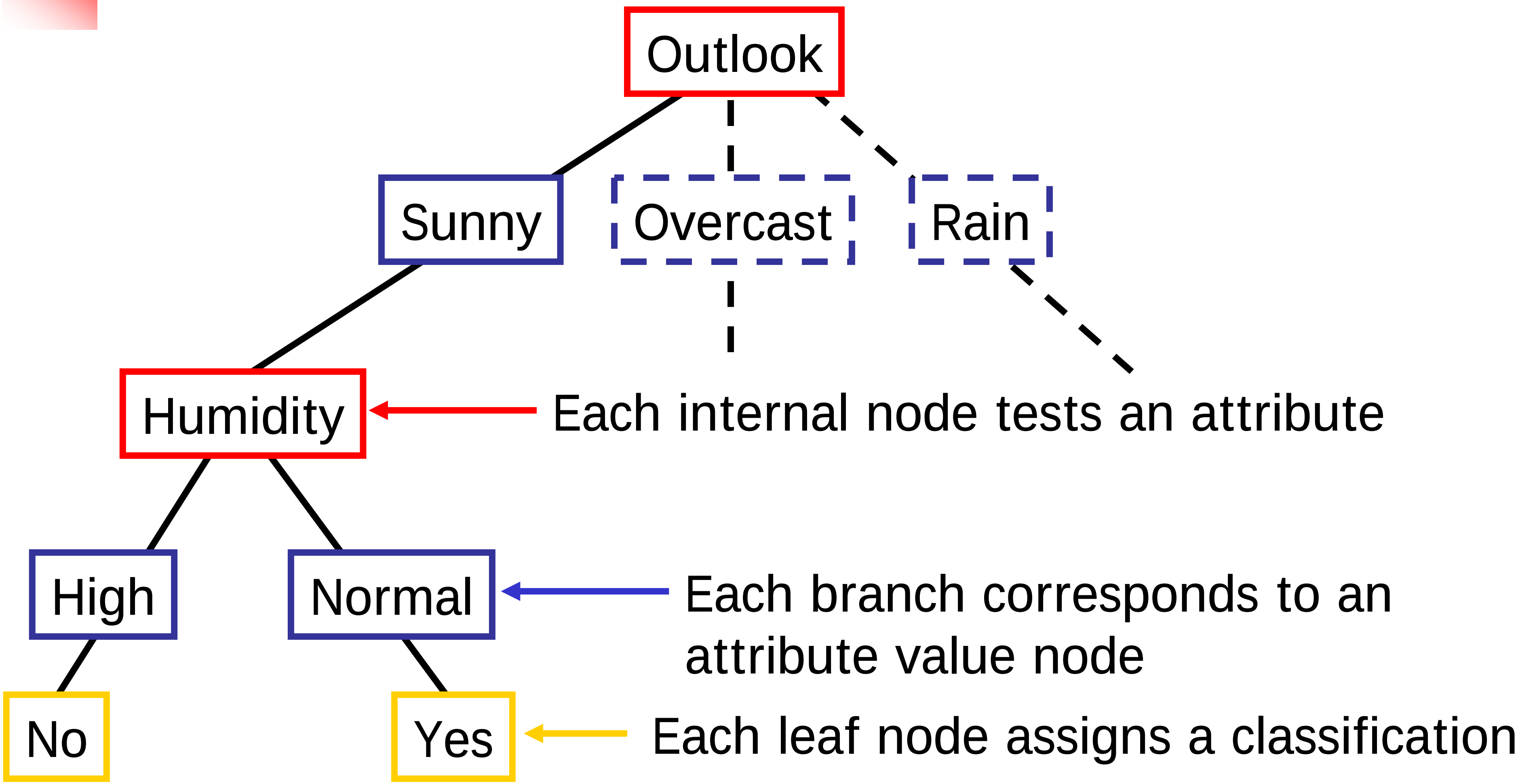
Outline

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

Decision Tree for PlayTennis

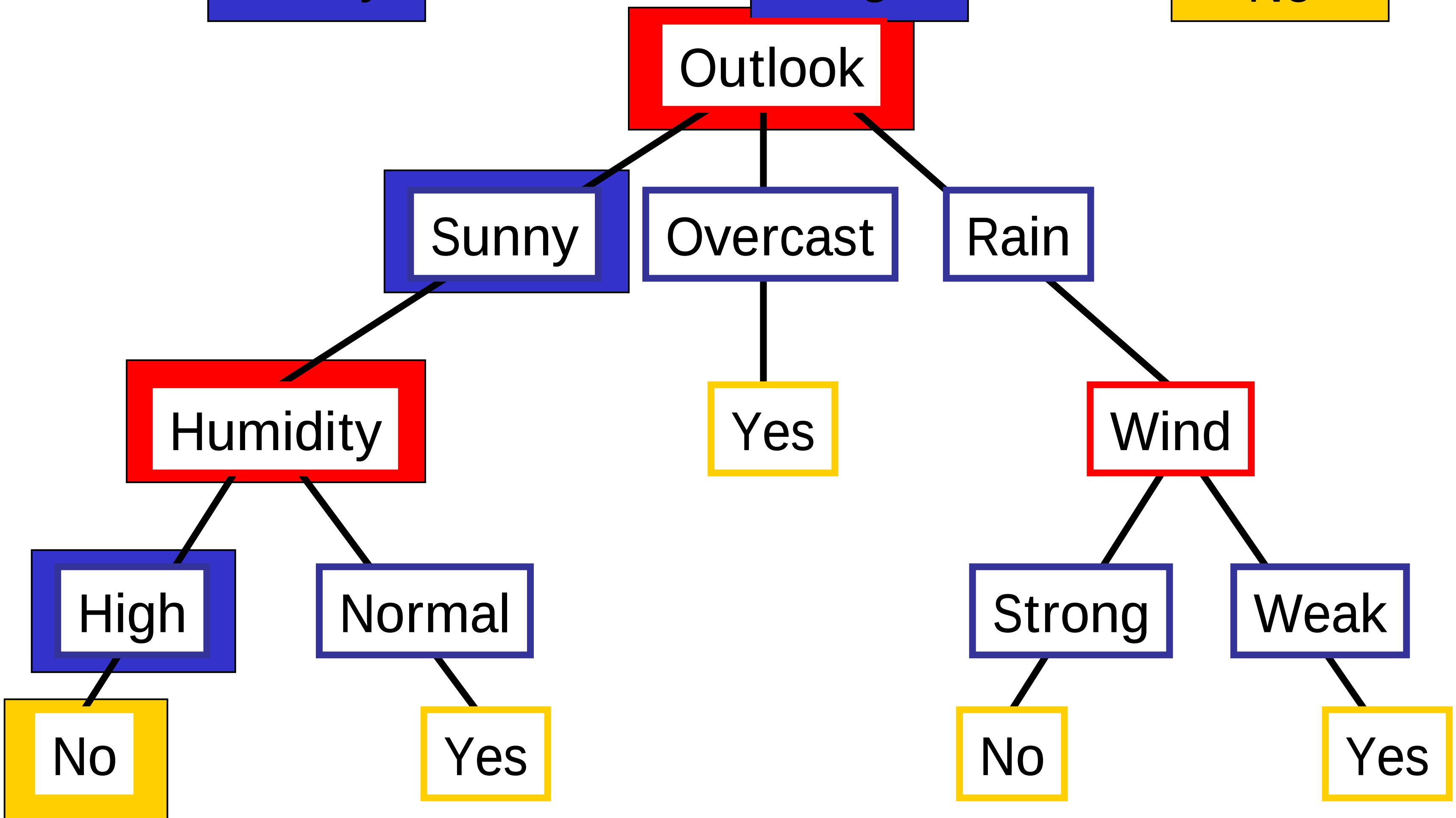


Decision Tree for PlayTennis



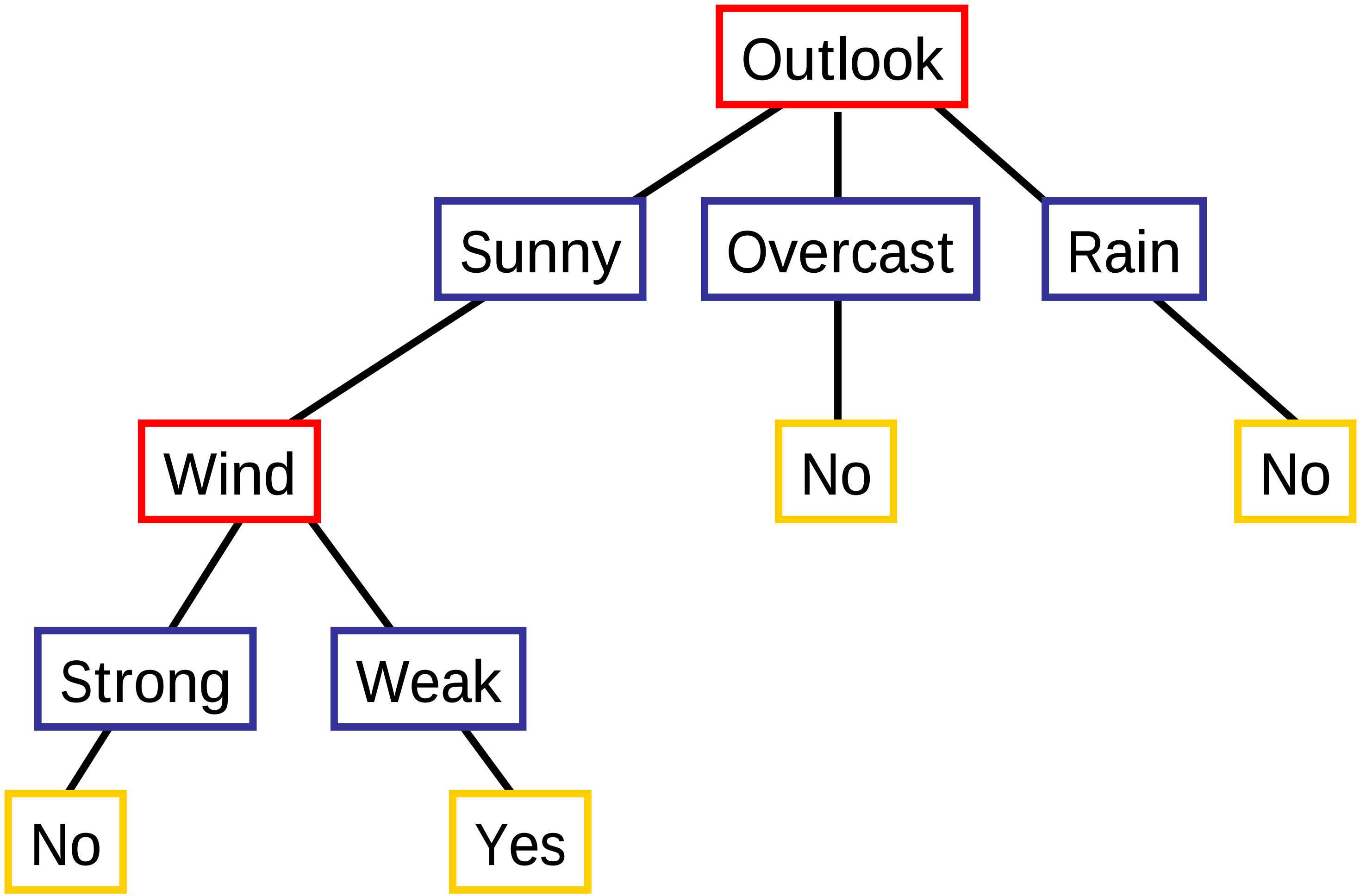
Decision Tree for PlayTennis

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	? No



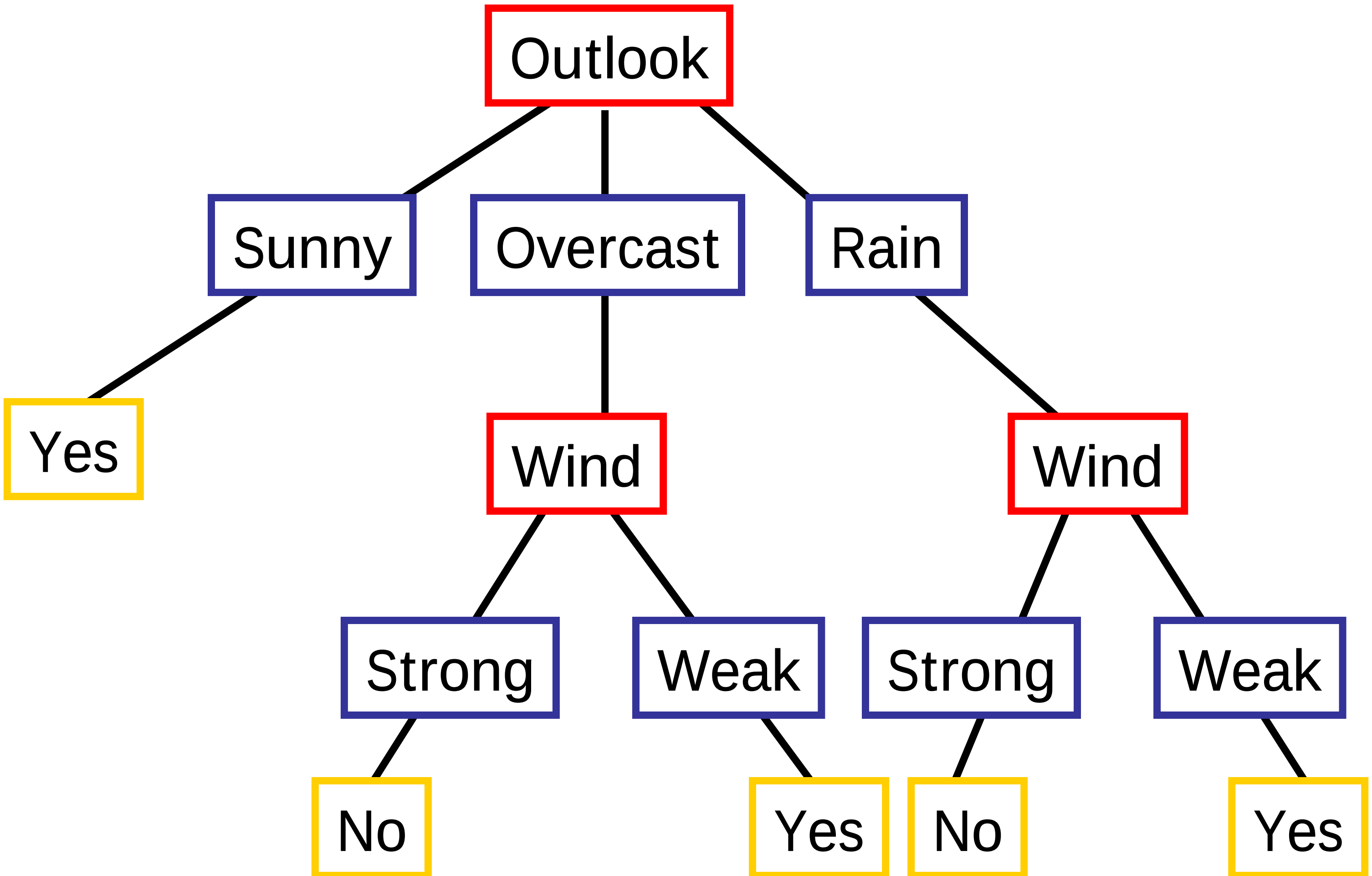
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



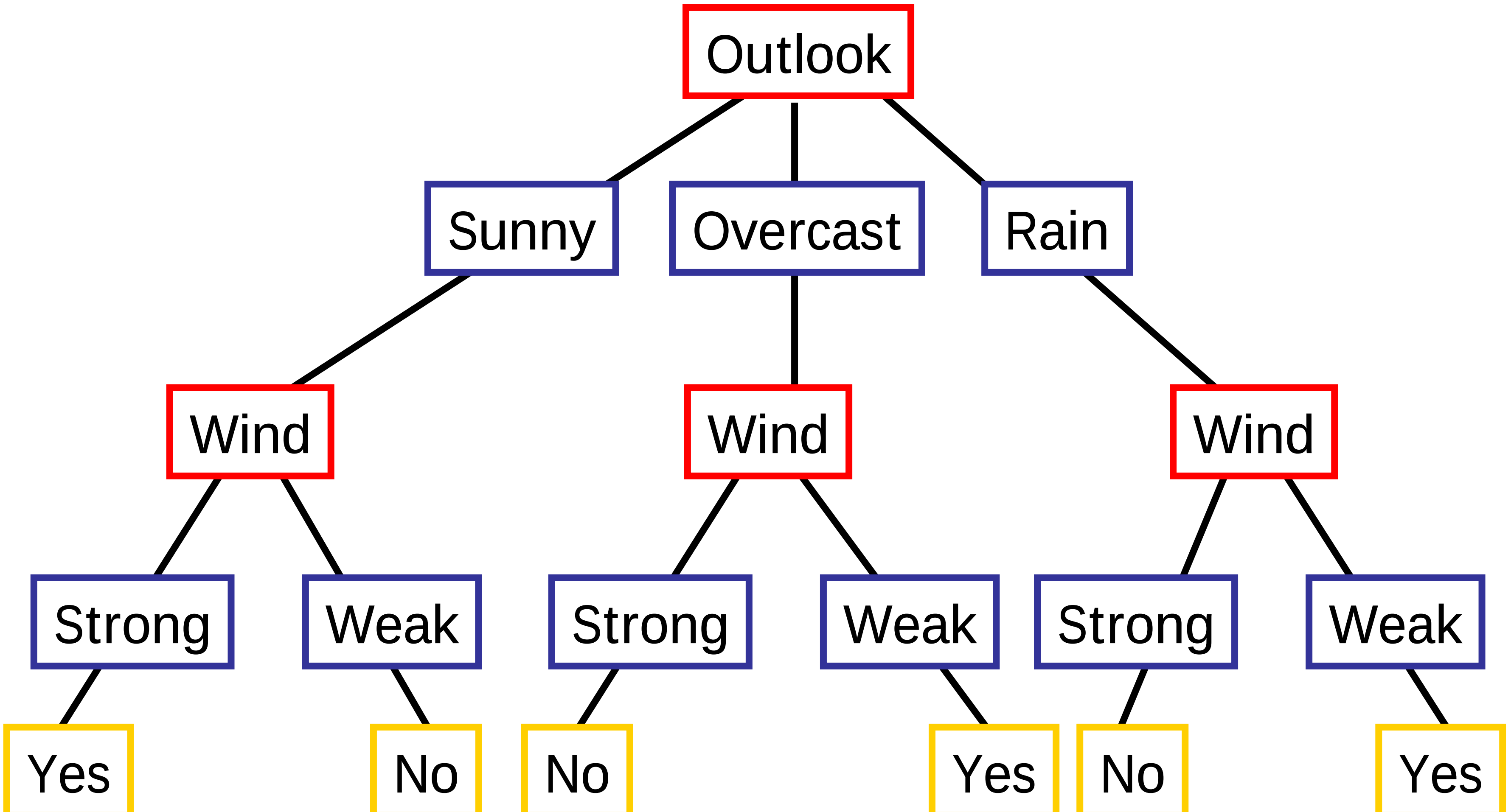
Decision Tree for Disjunction

Outlook=Sunny $\vee$ Wind=Weak



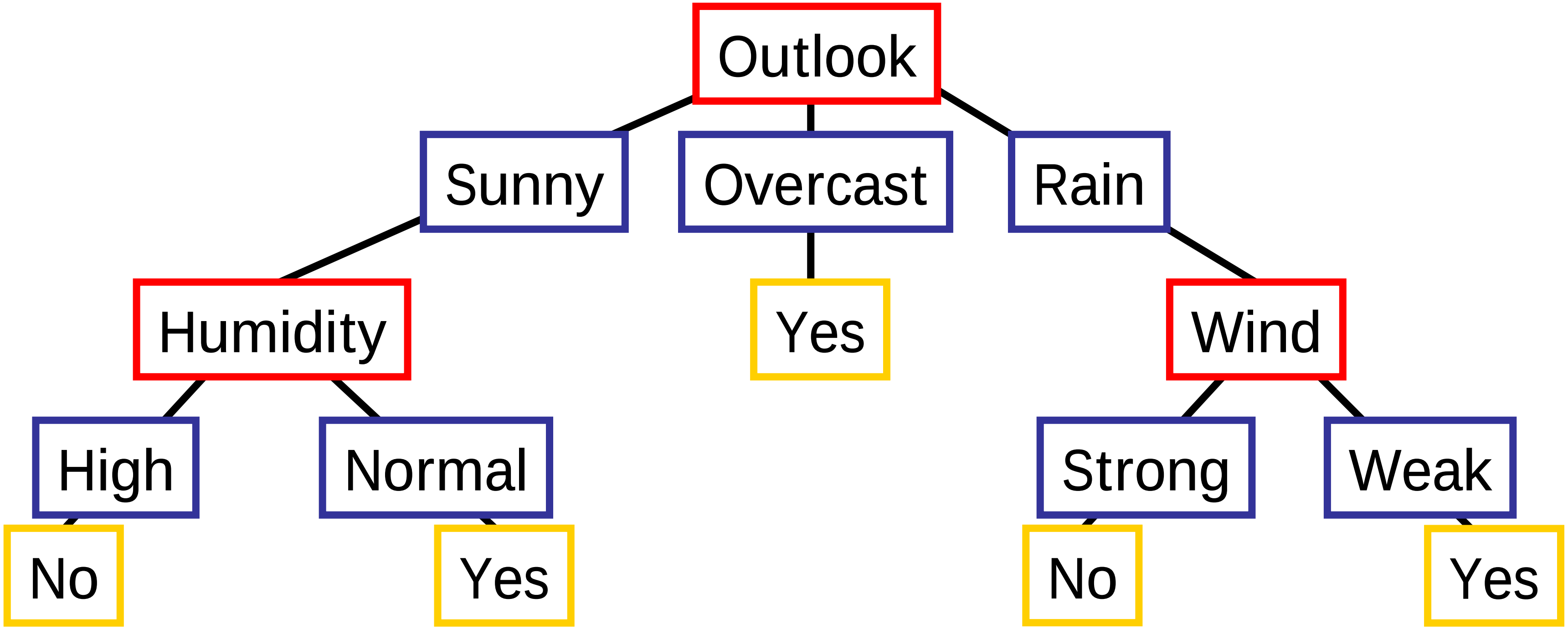
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

- decision trees represent disjunctions of conjunctions



$$
\begin{aligned}
&(\text{Outlook=Sunny} \wedge \text{Humidity=Normal}) \\
\vee\ &(\text{Outlook=Overcast}) \\
\vee\ &(\text{Outlook=Rain} \wedge \text{Wind=Weak})
\end{aligned}
$$

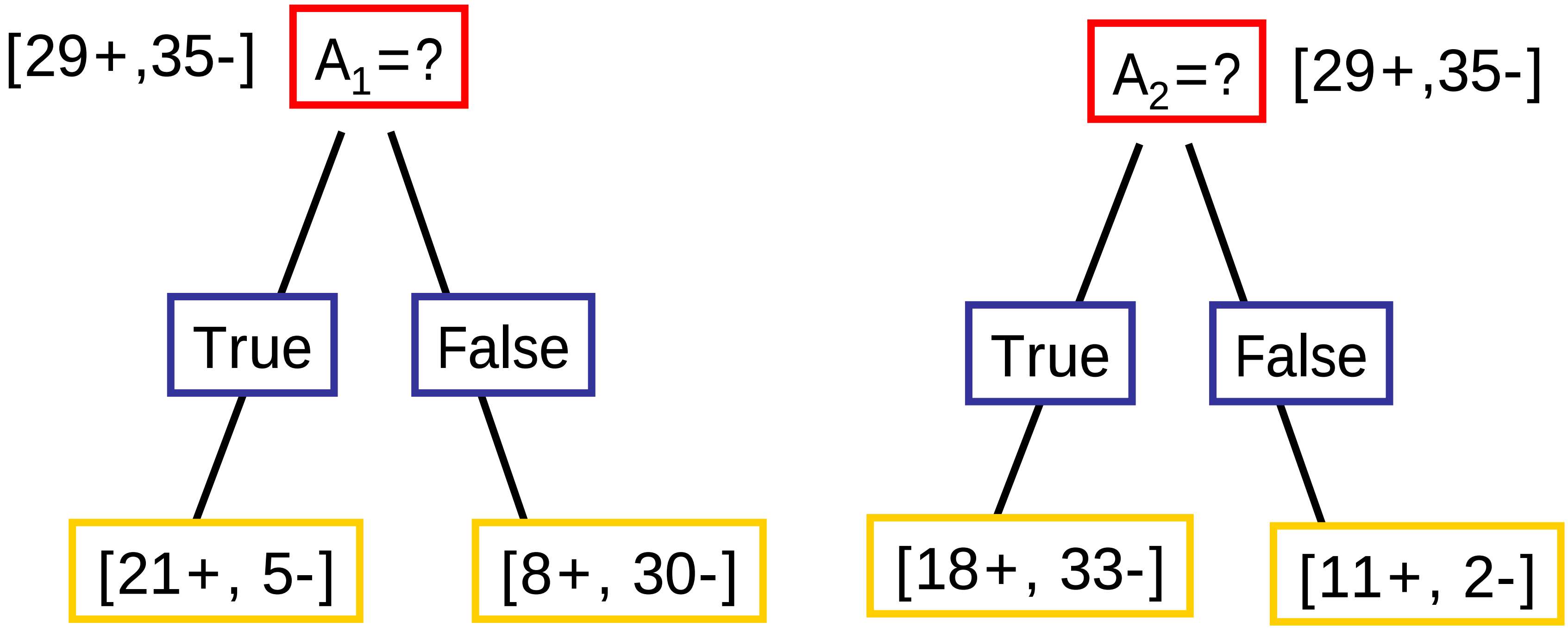
When to consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- 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)

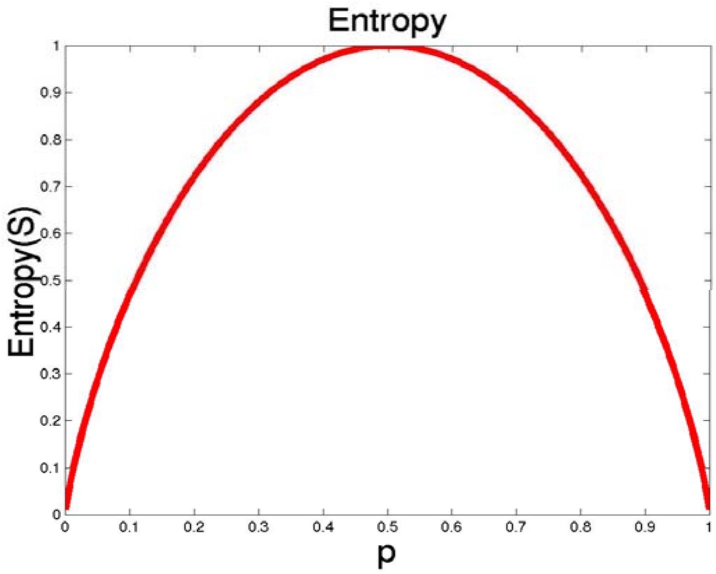
Top-Down Induction of Decision Trees ID3

1. A $\leftarrow$ the “best” decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

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:

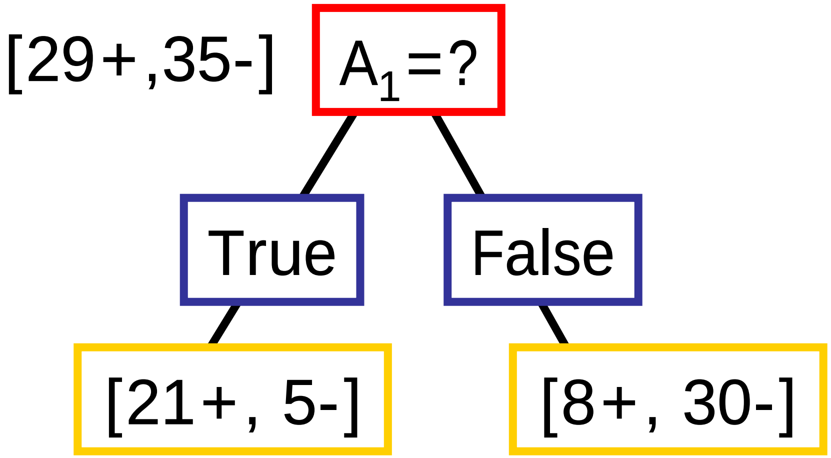
$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

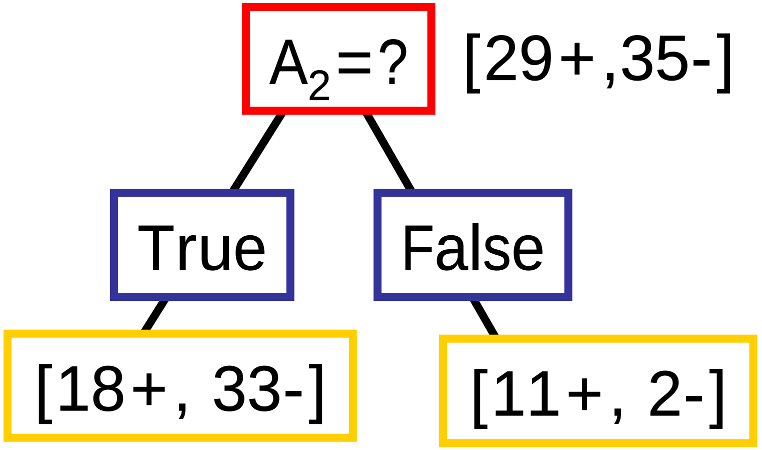
Information Gain

- Gain(S,A): expected reduction in entropy due to sorting S on attribute A

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} |S_v|/|S| \; \text{Entropy}(S_v)$$

$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$





Information Gain

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

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

$\text{Gain}(S,A_1)=\text{Entropy}(S)$

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

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

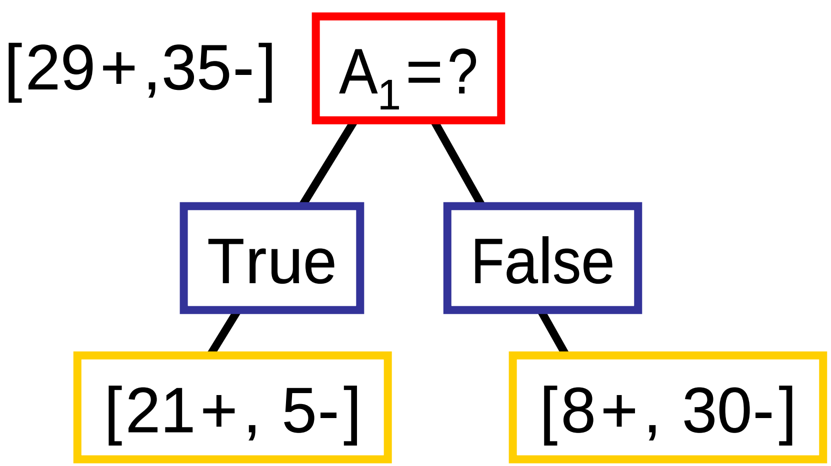
Entropy([8+,30-]) = 0.62

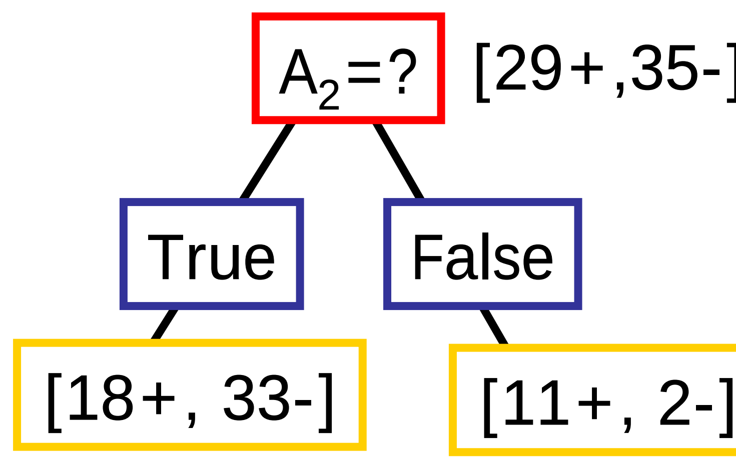
$\text{Gain}(S,A_2)=\text{Entropy}(S)$

-51/64*Entropy([18+,33-])

-13/64*Entropy([11+,2-])

=0.12

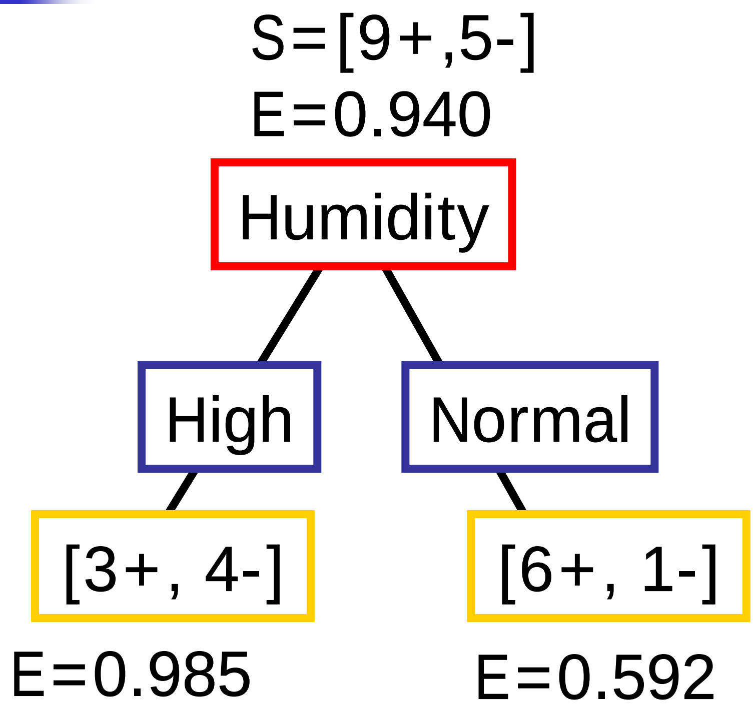


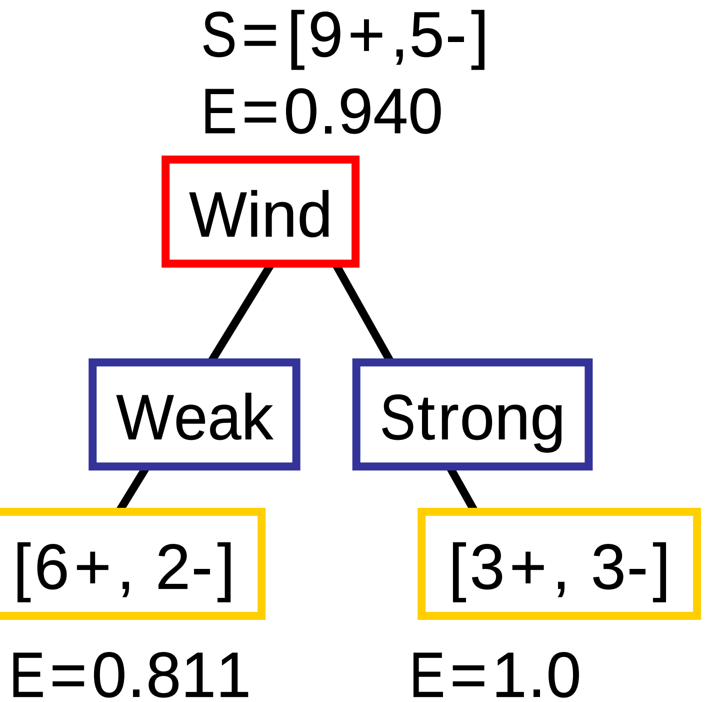


Training Examples

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

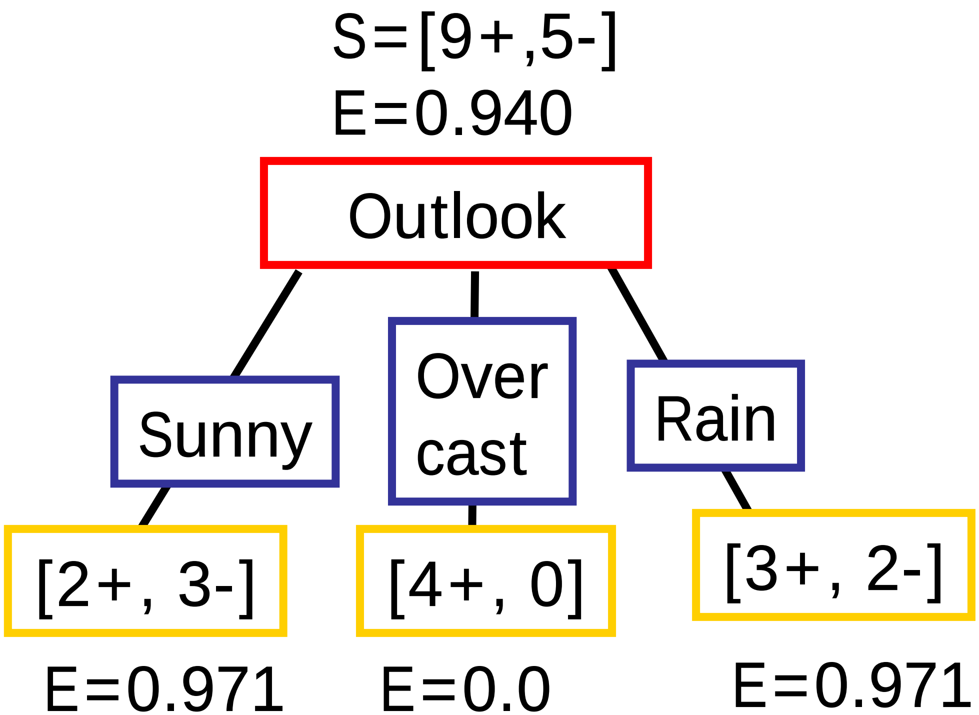




Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

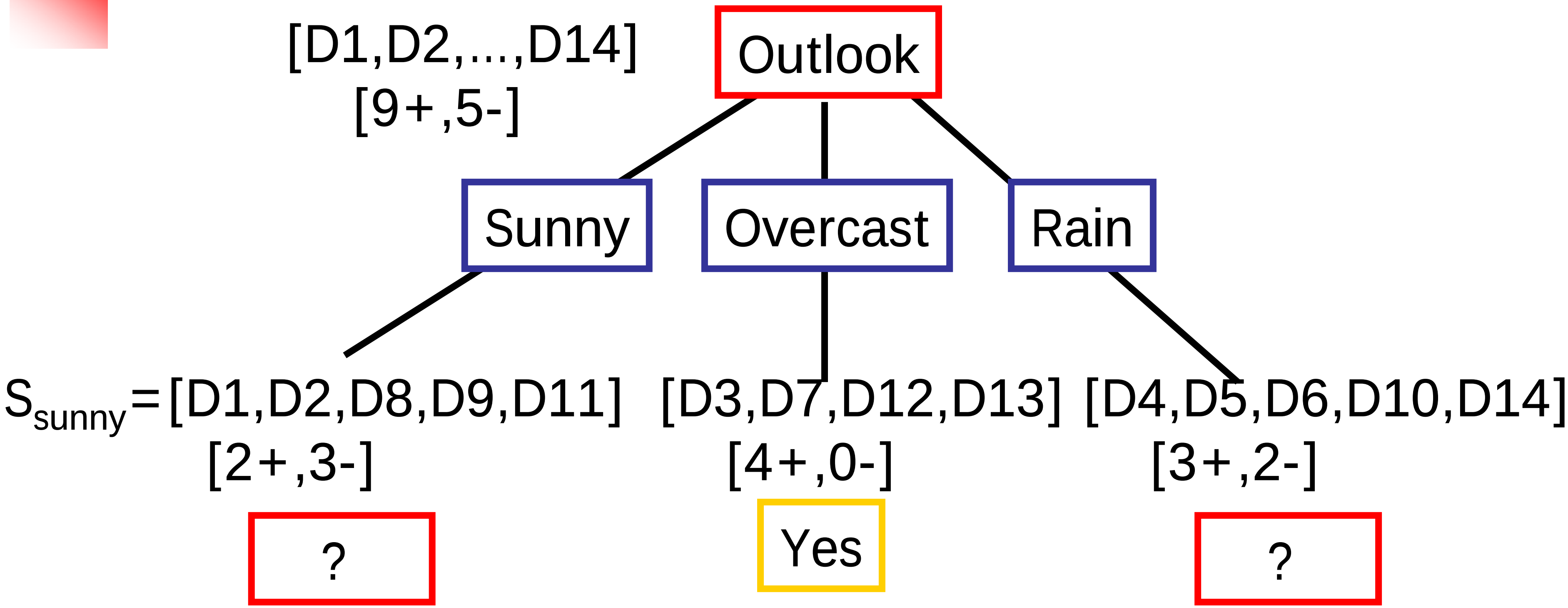
Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

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

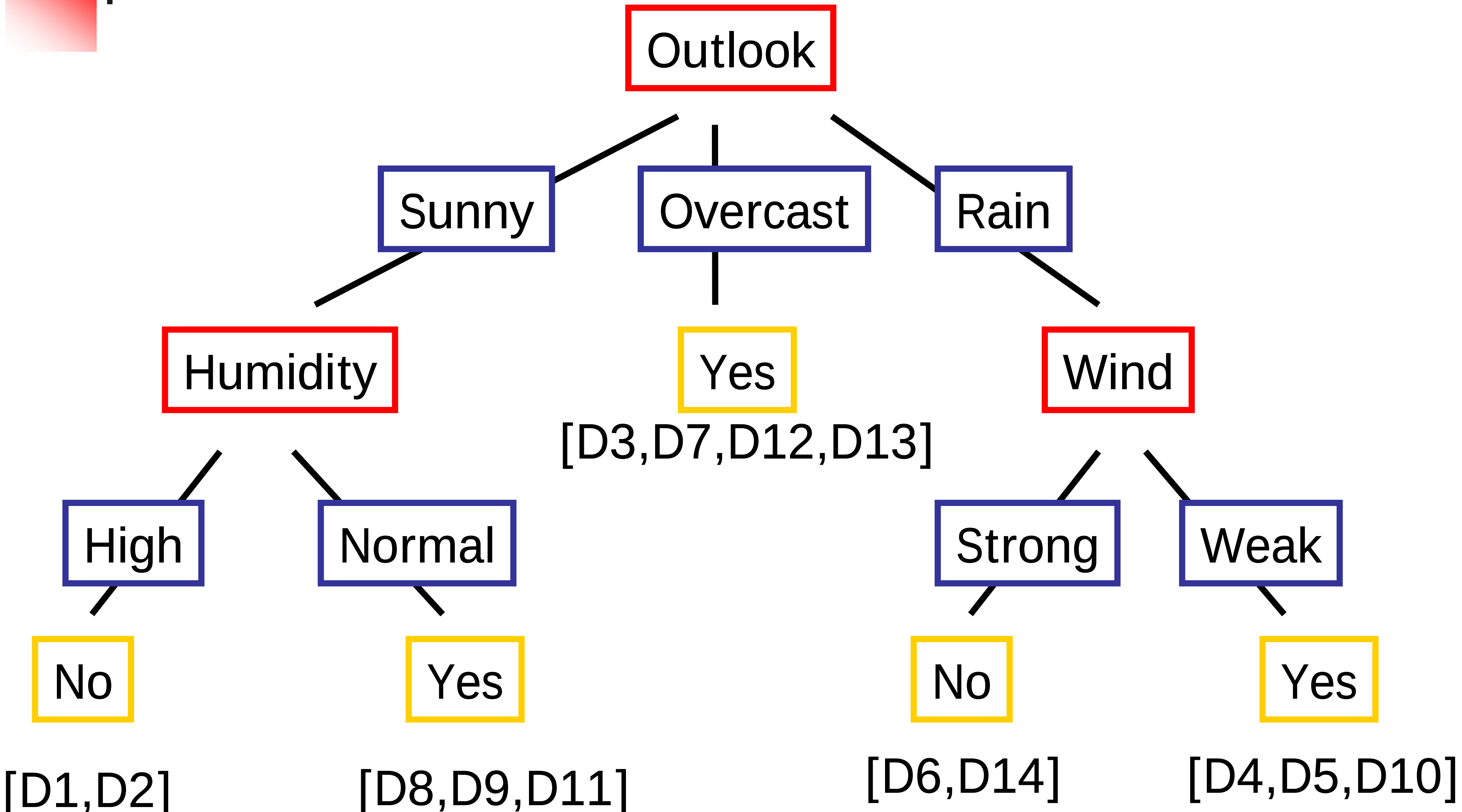


$\text{Gain}(S_{sunny}\text{ , Humidity})=0.970-(3/5)0.0 - 2/5(0.0) = 0.970$

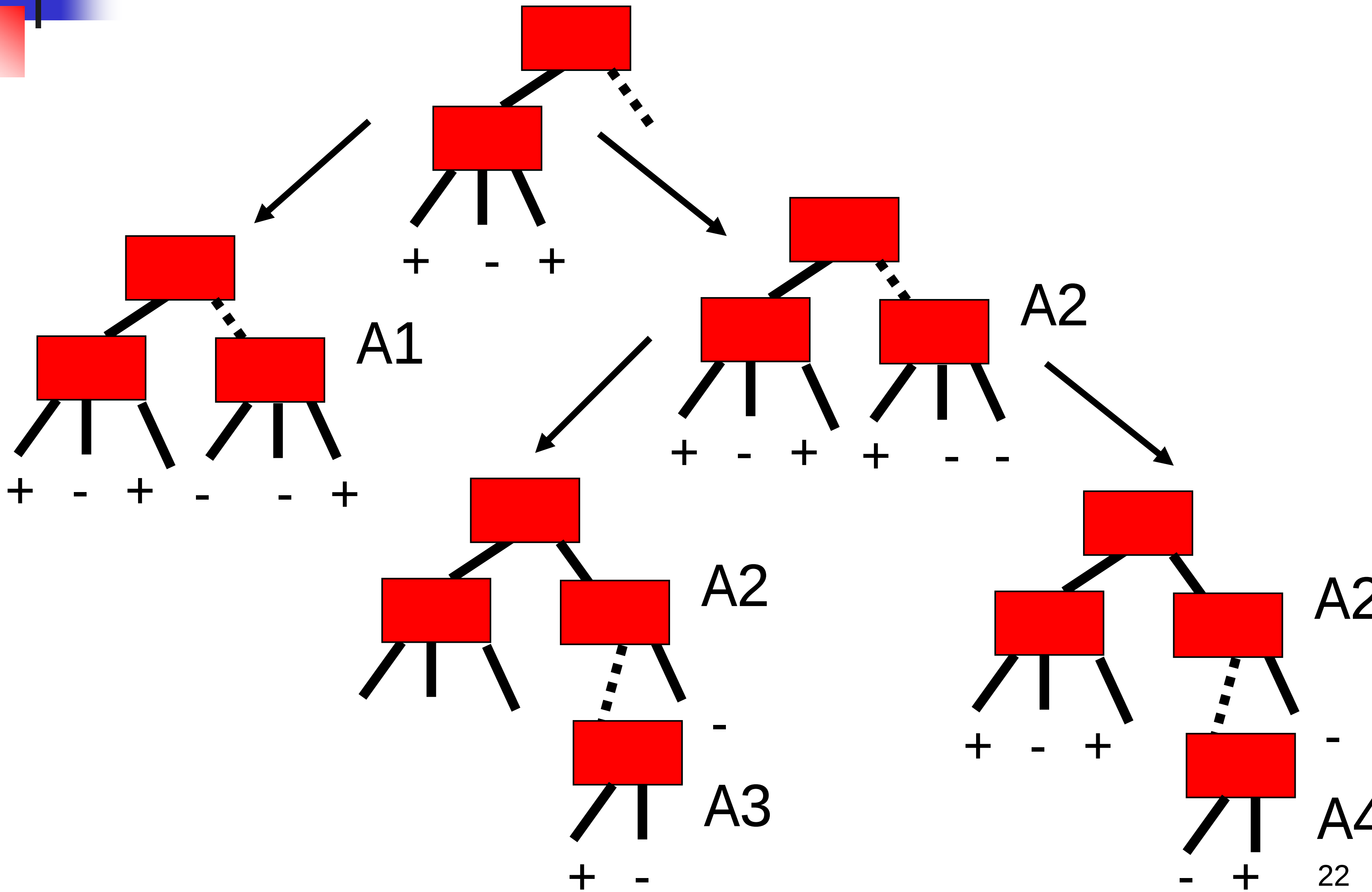
$\text{Gain}(S_{sunny}\text{ , Temp.})=0.970-(2/5)0.0 - 2/5(1.0)-(1/5)0.0 = 0.570$

$\text{Gain}(S_{sunny}\text{ , Wind})=0.970= -(2/5)1.0 - 3/5(0.918) = 0.019$

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 high information gain attributes close 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
- 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

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
- 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: $error_{train}(h)$
- Entire distribution D of data: $error_D(h)$

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

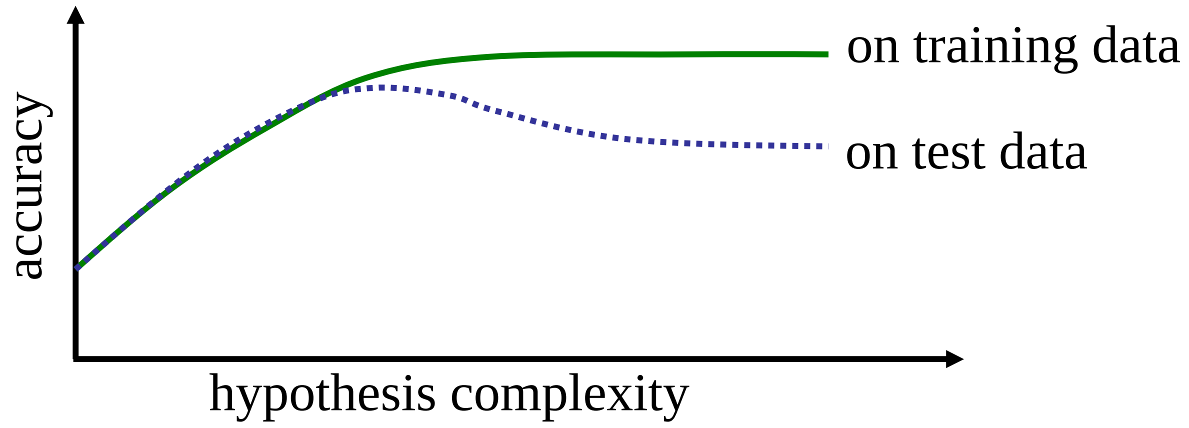
$$error_{train}(h) < error_{train}(h')$$

and

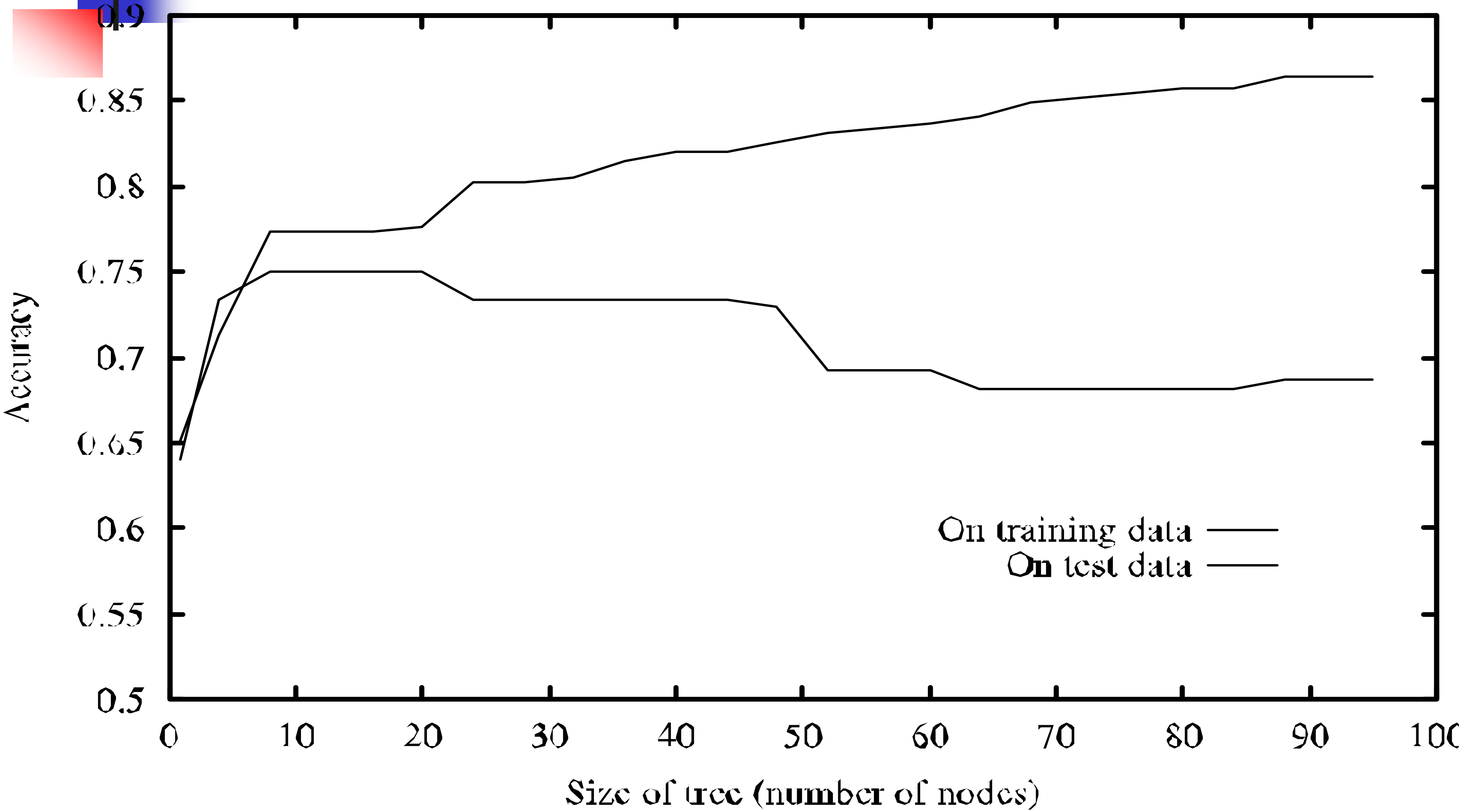
$$error_D(h) > error_D(h')$$

Overfitting (2)

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - There may be noise in the training data that the tree is erroneously fitting.
  - The algorithm may be making poor decisions towards the leaves of the tree that are based on very little data and may not reflect reliable trends.
- A hypothesis, *h*, is said to overfit the training data is there exists another hypothesis which, *h´* , such that *h* has less error than *h´* on the training data but greater error on independent test data.



Overfitting in Decision Tree Learning

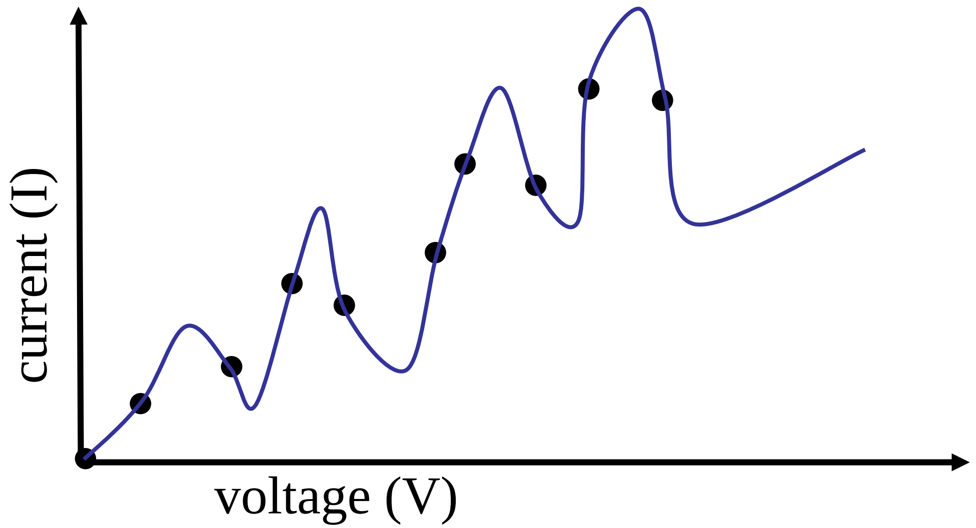


Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)

Experimentally measure 10 points

Fit a curve to the Resulting data.

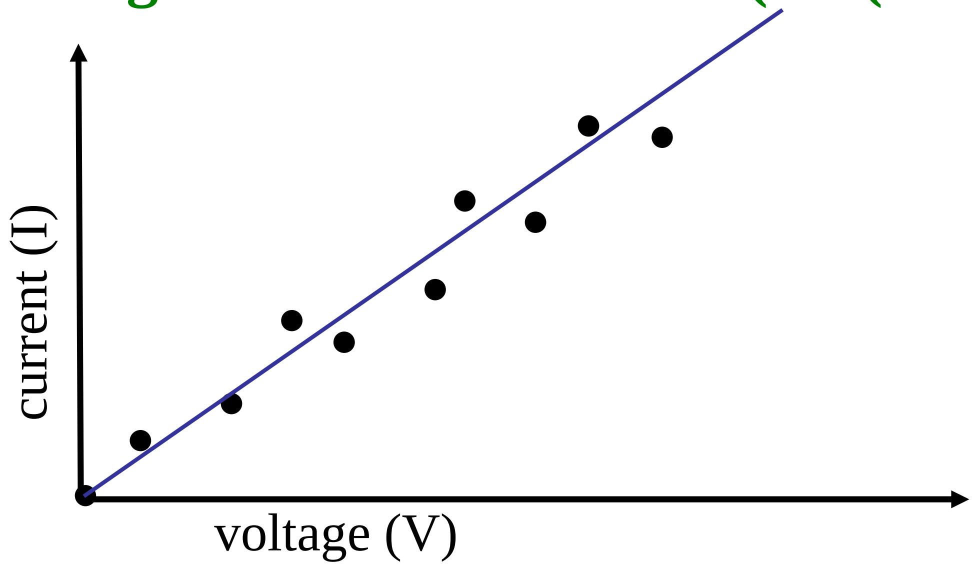


Perfect fit to training data with an 9th degree polynomial
(can fit *n* points exactly with an *n*-1 degree polynomial)

Ohm was wrong, we have found a more accurate function!

Overfitting Example

Testing Ohms Law: V = IR   (I = (1/R)V)



Better generalization with a linear function
that fits training data less accurately.

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

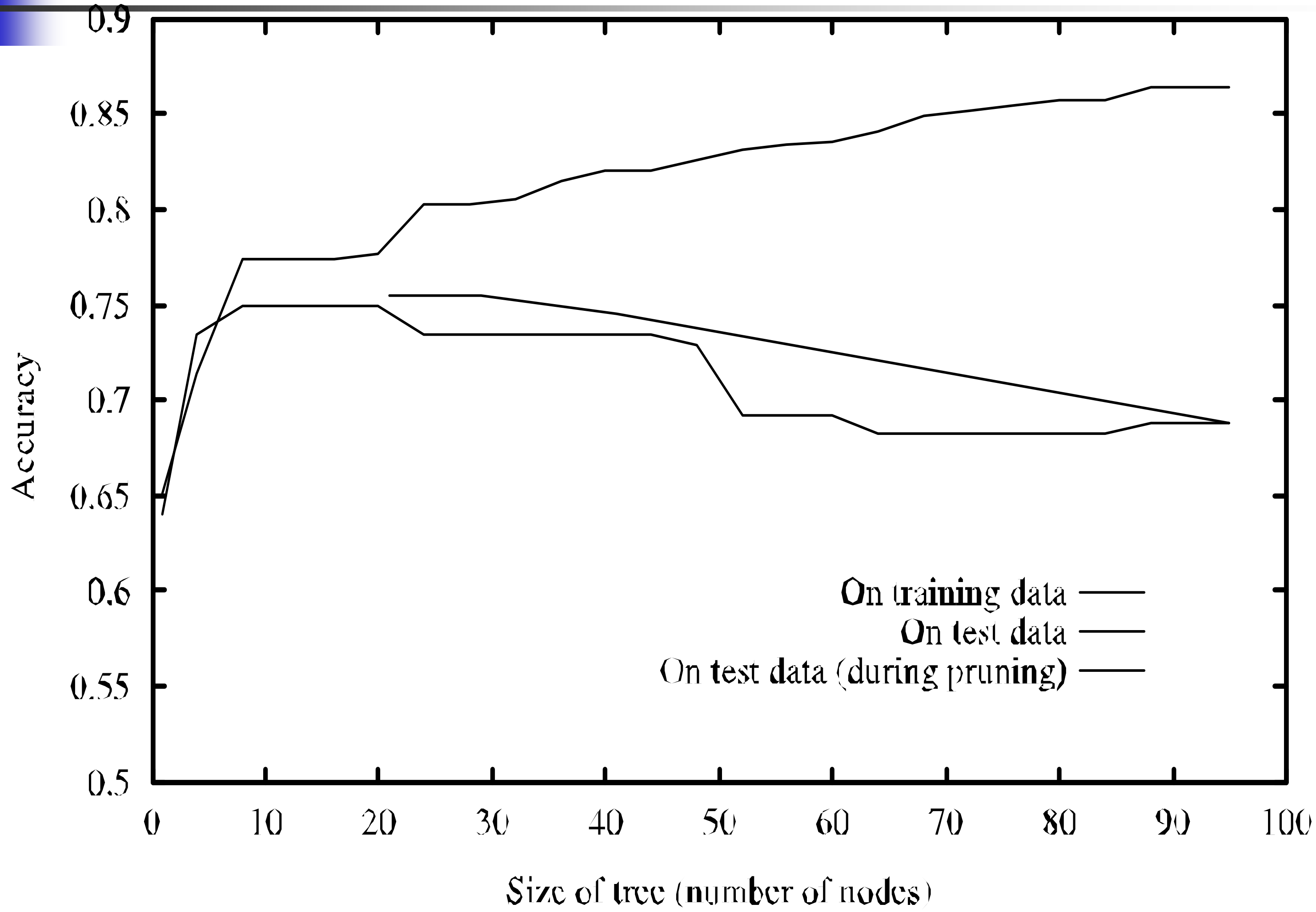
Split data into *training* and validation set

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

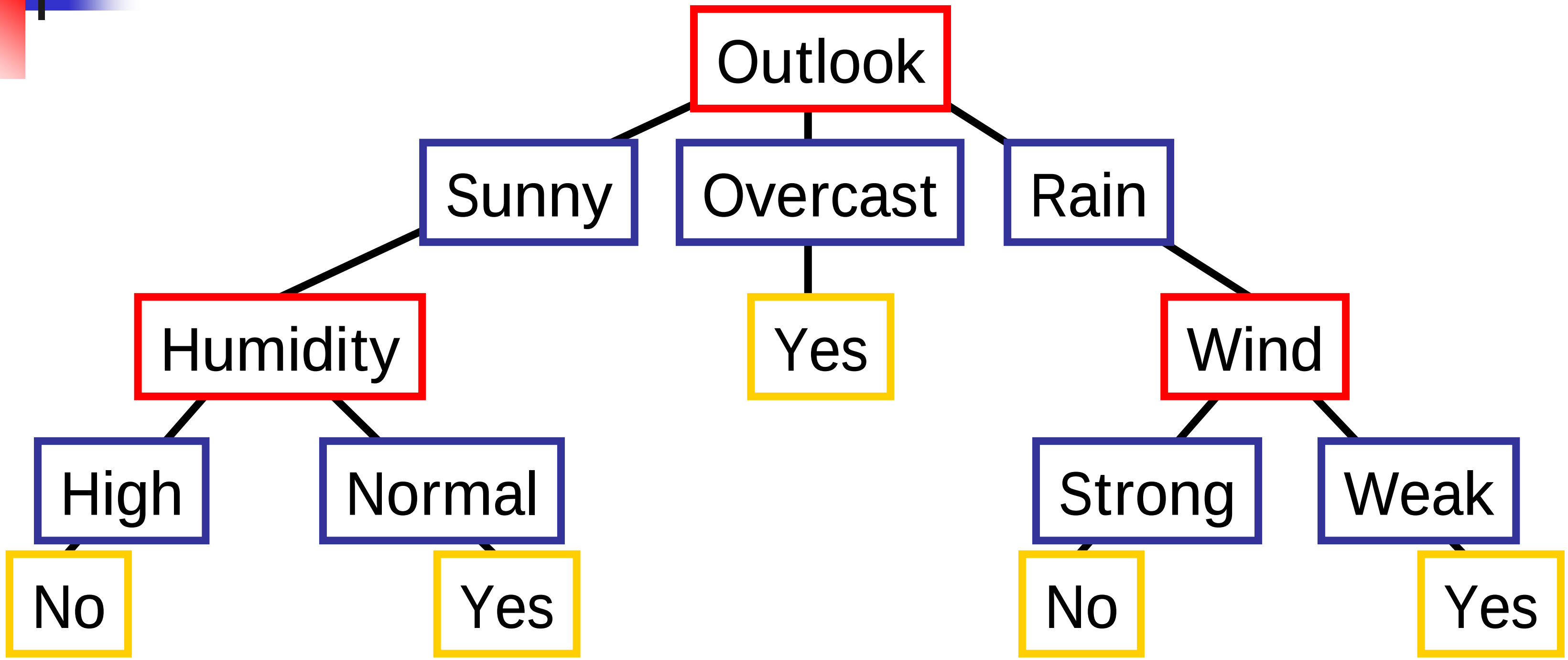


Rule-Post Pruning

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

Additional Decision Tree Issues

- Better splitting criteria
  - Information gain prefers features with many values.
- Continuous features
- Predicting a real-valued function (regression trees)
- Missing feature values
- Features with costs
- Misclassification costs
- Incremental learning
  - ID4
  - ID5
- Mining large databases that do not fit in main memory

Continuous Valued Attributes

Create a discrete attribute to test continuous

- Temperature = 24.5^0C
- (Temperature > 20.0^0C) = {true, false}

Where to set the threshold?

Temperatur	15^0C	18^0C	19^0C	22^0C	24^0C	27^0C
PlayTennis	No	No	Yes	Yes	Yes	No

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:

$GainRatio(S,A) = Gain(S,A) / SplitInformation(S,A)$

$SplitInformation(S,A) = -\sum_{i=1..c} |S_i|/|S| \log_2 |S_i|/|S|$

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 :

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

$2^{Gain(S,A)}-1/(Cost(A)+1)^w$ $w \in [0,1]$ [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 pi to each possible value vi of A
  - Assign fraction pi of example to each descendant in tree

Classify new examples in the same fashion