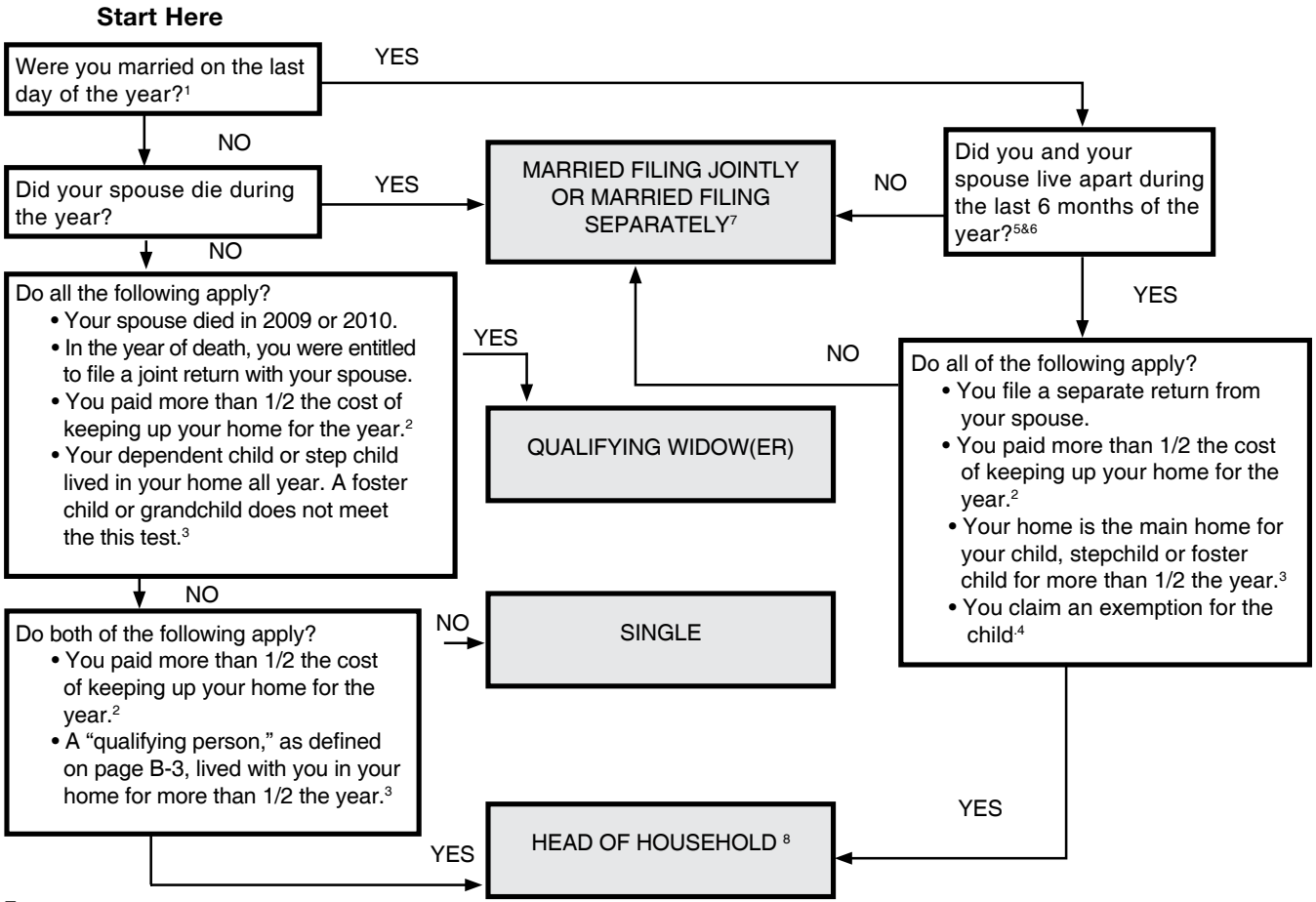
Decision Trees

(Slides inherited from Daniel Lowd, Martin Riedmiller, Pedro Domingos, Tom Mitchell, and others)

Reading: CIML chapter 1

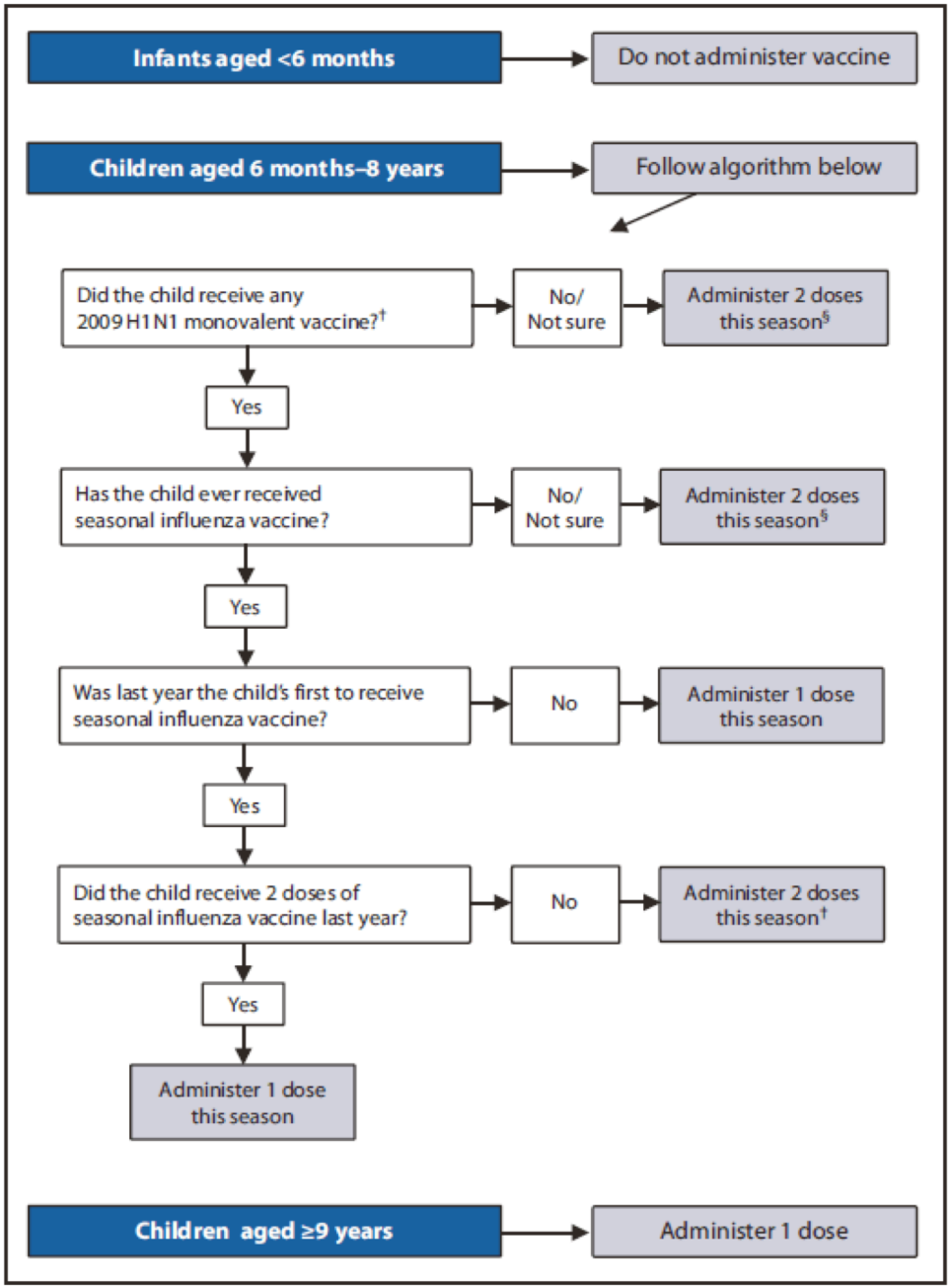
Note: These slides contain some information not in the readings!

Determination of Filing Status – Decision Tree



Footnotes:

1. Answer "NO" to this question if, on the last day of the year, you were legally separated from your spouse under a divorce or separate maintenance decree.
2. Include in the cost of upkeep expenses such as rent, mortgage interest, real estate taxes, insurance on the home, repairs, utilities and food eaten in the home.
   - Payments received under TANF or other public assistance programs used to pay costs of keeping up the home cannot be counted as


Infants aged <6 months
Do not administer vaccine
Children aged 6 months–8 years
Follow algorithm below
Did the child receive any 2009 H1N1 monovalent vaccine?†
No/ Not sure
Administer 2 doses this season§
Yes
Has the child ever received seasonal influenza vaccine?
No/ Not sure
Administer 2 doses this season§
Yes
Was last year the child's first to receive seasonal influenza vaccine?
No
Administer 1 dose this season
Yes
Did the child receive 2 doses of seasonal influenza vaccine last year?
No
Administer 2 doses this season†
Yes
Administer 1 dose this season
Children aged ≥9 years
Administer 1 dose

A training dataset from the past experience

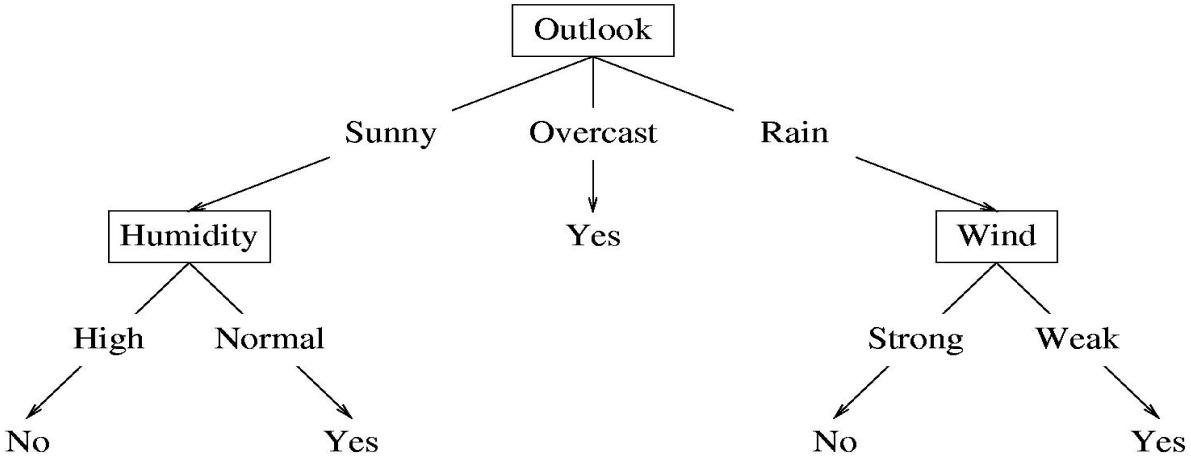
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Outlook, Temperature, Humidity and Wind are features/attributes

Should we play tennis today?

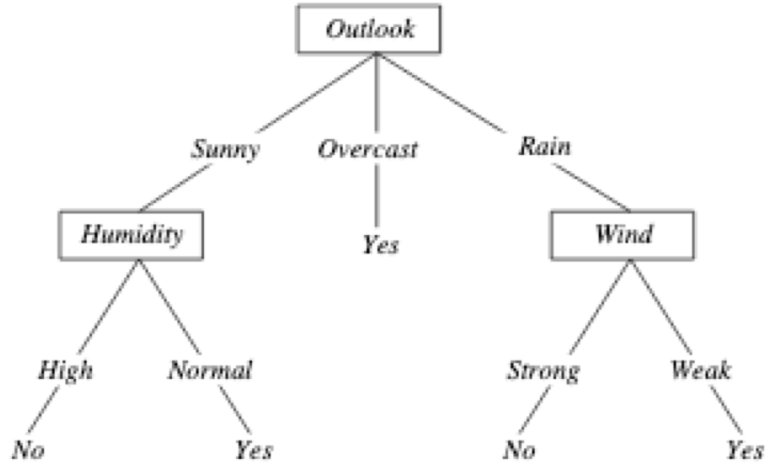
The decision tree we would learn

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

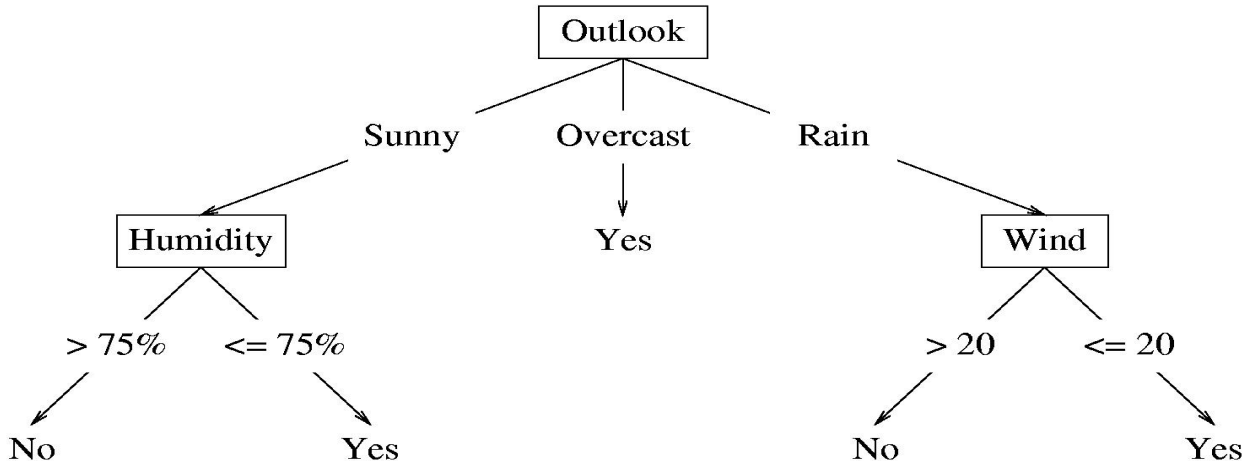
Converting a decision tree to rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$
IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$
. . .

Decision trees can handle real-valued features

If the features are continuous, internal nodes may test the value of a feature against a threshold.

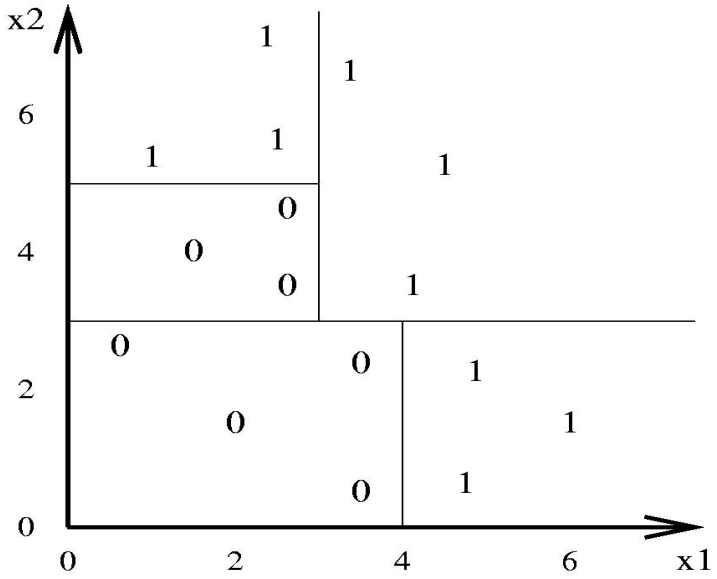


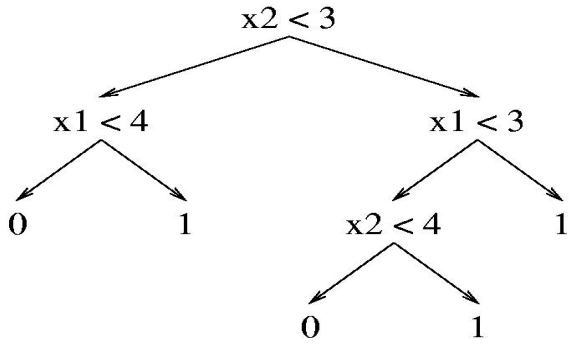
- Thresholds are decided based on the values of the features in the training dataset
  - Sort the feature values that occur in training set
  - Determine points where class changes

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

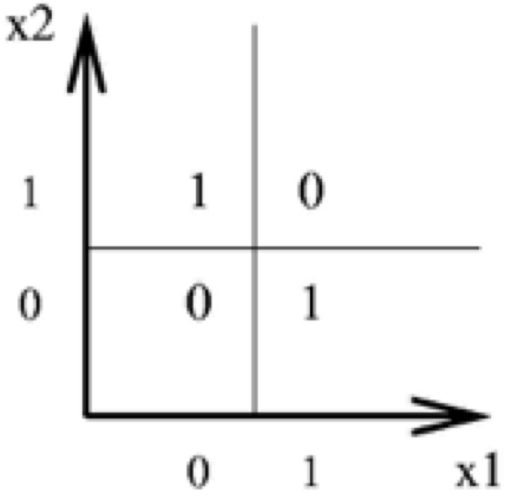
Decision Tree Decision Boundaries

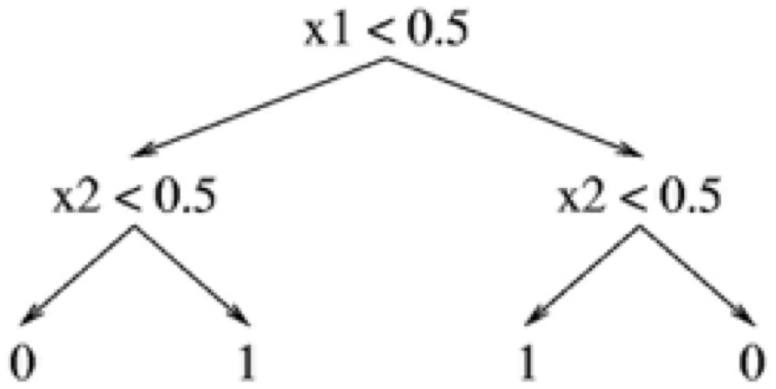
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision trees can represent any Boolean function





The tree will require exponentially many nodes in the worst case, however.

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$
- **etc.**

The ID3 algorithm (Iterative Dichotomiser 3) to learn decision trees

Function ID3

- **Input:** Example set S
- **Output:** Decision Tree DT

- If all examples in S belong to the same class c
  - return a new leaf and label it with c
- Else
  - i. Select an attribute A according to some heuristic function
  - ii. Generate a new node DT with A as test
  - iii. For each Value v_i of A
    - (a) Let S_i = all examples in S with $A = v_i$
    - (b) Use ID3 to construct a decision tree DT_i for example set S_i
    - (c) Generate an edge that connects DT and DT_i

The ID3 algorithm - Example

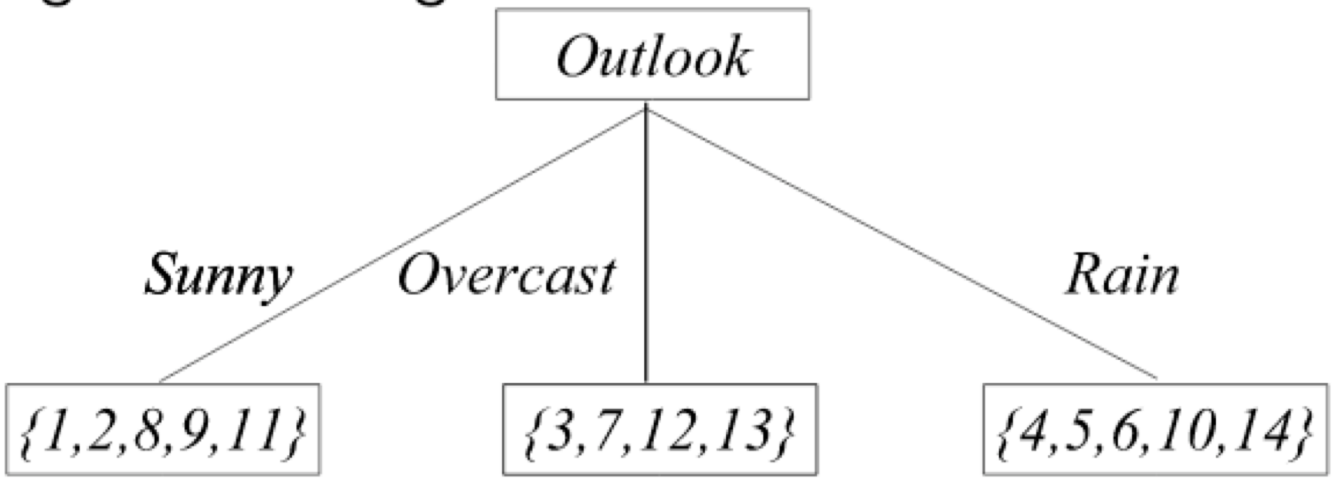
- Look at current training set S

$S=\{1,\ldots,14\}$

- Determine best attribute

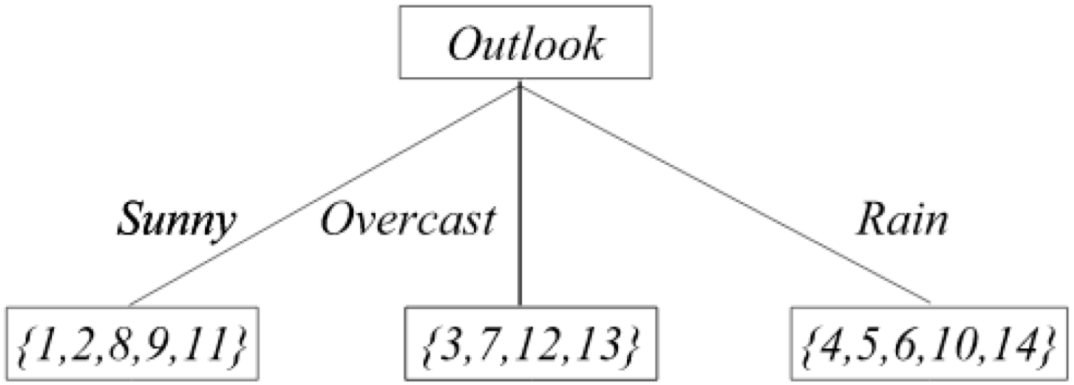
Outlook

- Split training set according to different values

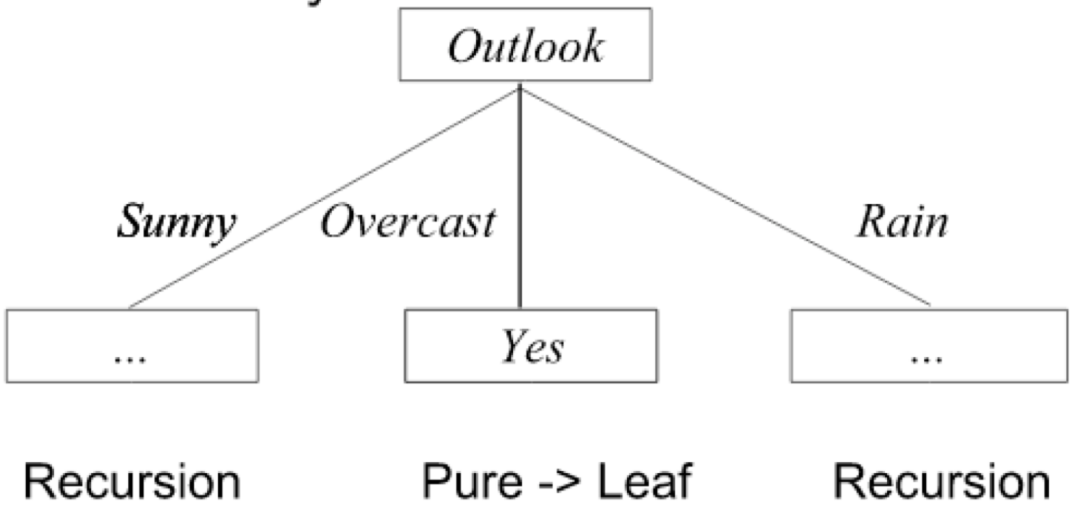


The ID3 algorithm - Example

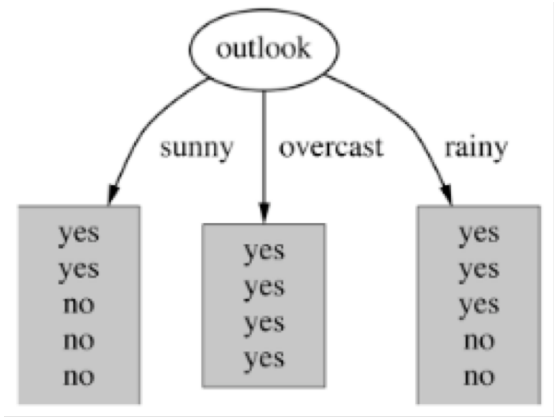
- Tree

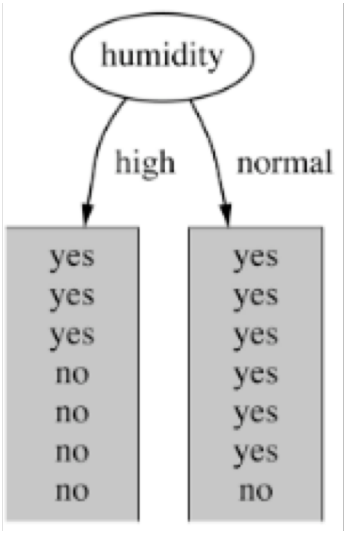


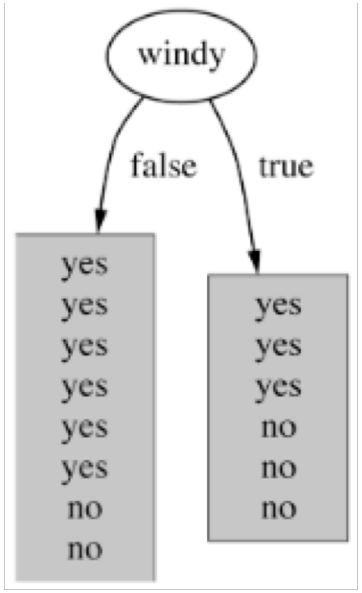
- Apply algorithm recursively

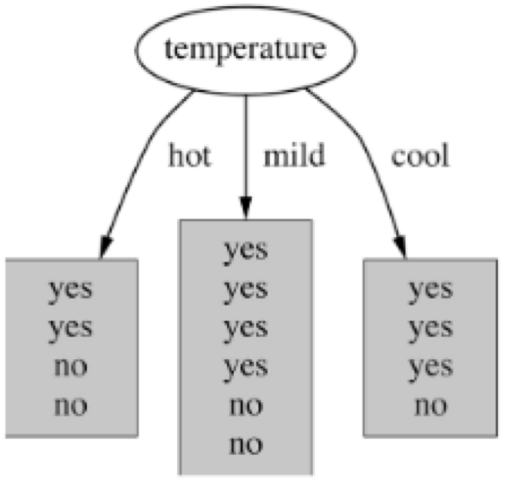


Which attribute is the best?





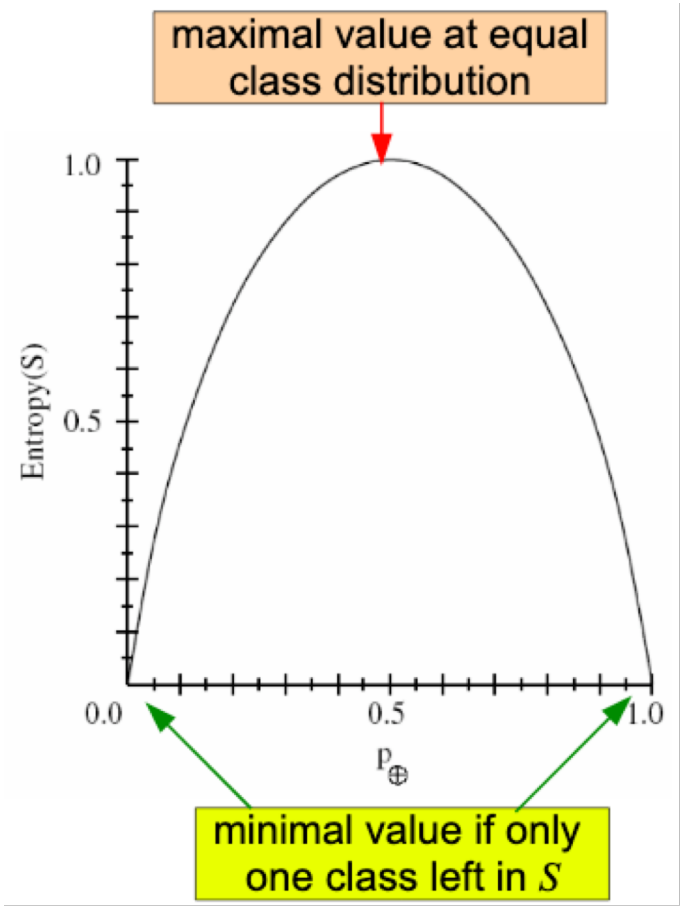




Which attribute is the best?

- ID3 Heuristics:
  - choosing the attribute that produces the "purest" nodes: the distribution of examples in each node is so that it mostly contains examples of a single class
  - best case: all examples in a node belong the the same class
  - worst case: all classes are equally likely
  - using entropy and information gain to achieve this property

Entropy



- $p_\oplus$ is the proportion of positive examples
- $p_\ominus$ is the proportion of negative examples
- Entropy measures the impurity of S
- $Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$
- Information can be seen as the negative of entropy

Information Gain

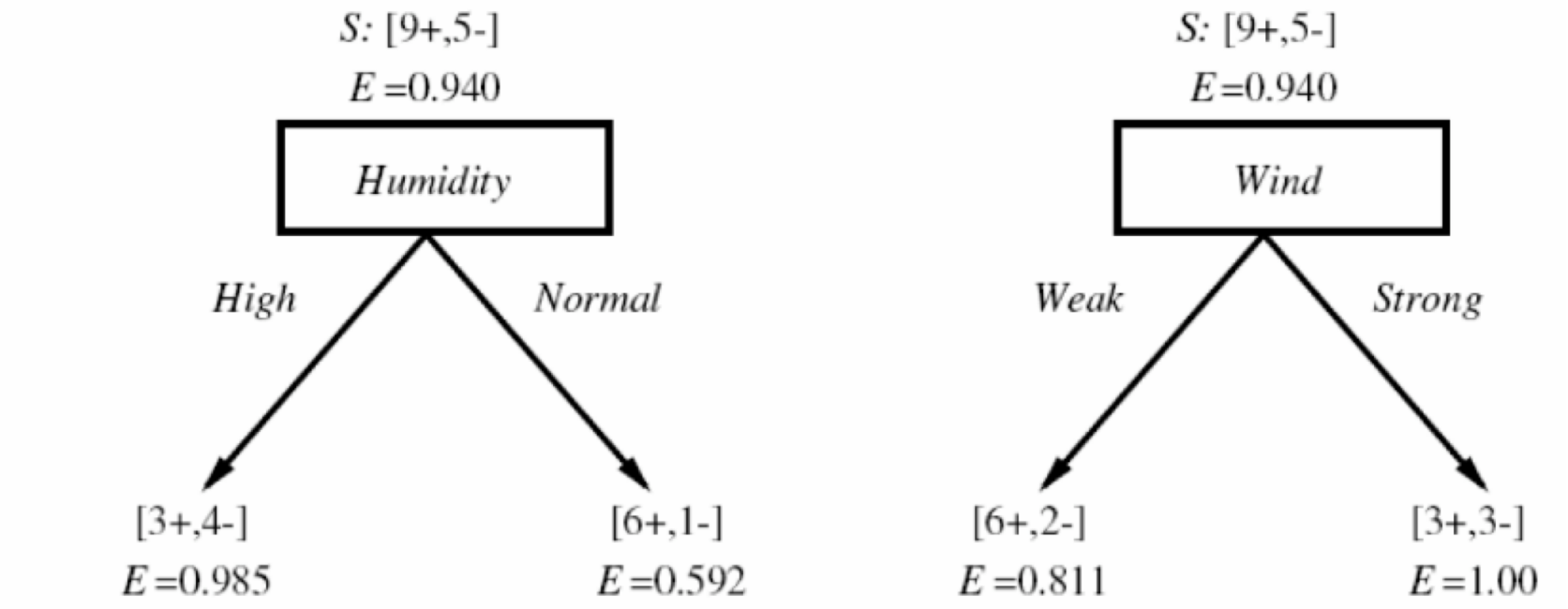
- Measuring attribute x creates subsets S_1 and S_2 with different entropies
- Taking the mean of $Entropy(S_1)$ and $Entropy(S_2)$ gives conditional entropy $Entropy(S|x)$, i.e. in general: $Entropy(S|x) = \sum_{v \in Values(x)} \frac{|S_v|}{|S|} Entropy(S_v)$
- $\rightarrow$ Choose that attribute that maximizes difference:

$$Gain(S, x) := Entropy(S) - Entropy(S|x)$$

- $Gain(S, x)$ = expected reduction in entropy due to partitioning on x

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

Example



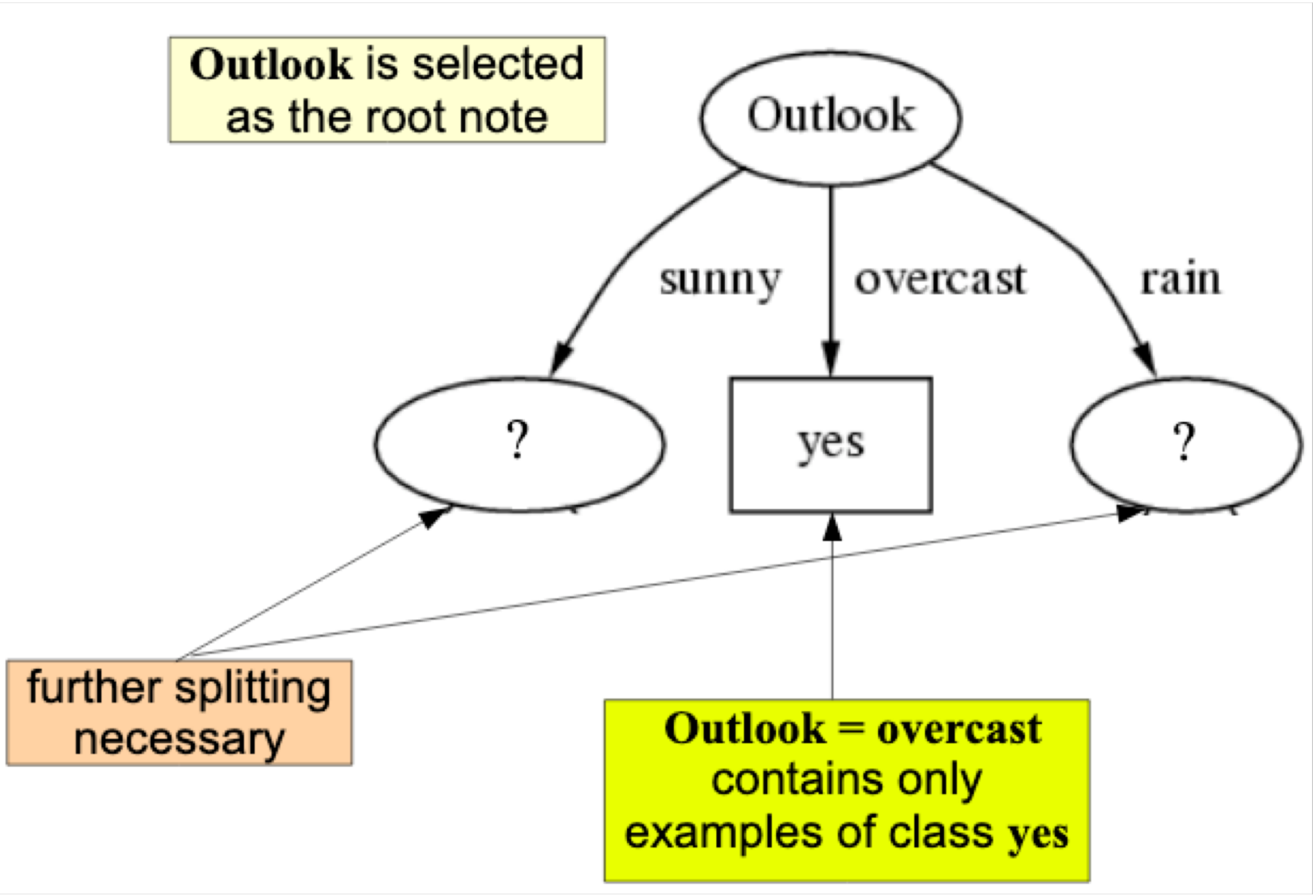
Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

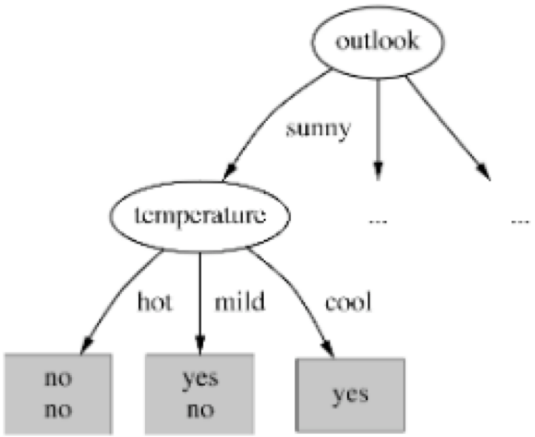
$Gain(S, Outlook) = 0.246$

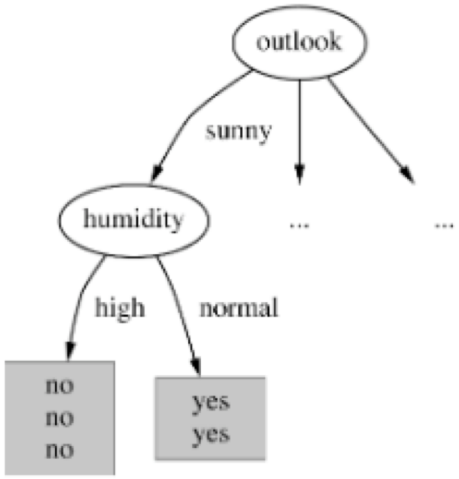
$Gain(S, Temperature) = 0.029$

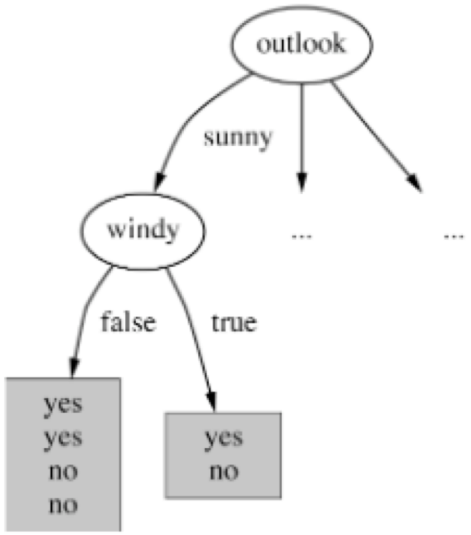
Example - continue



Example - continue



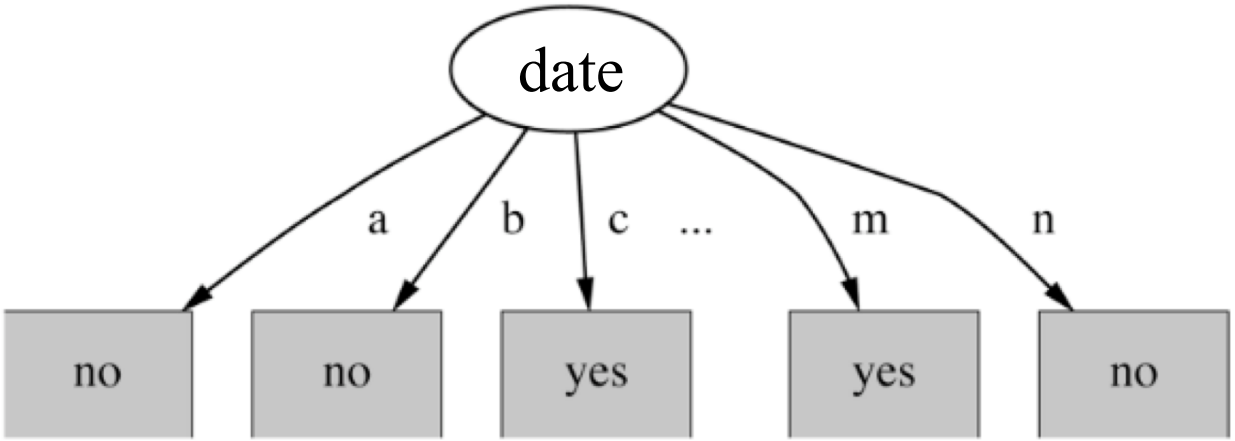




Gain(*Temperature*) = 0.571 bits
Gain(*Humidity*) = 0.971 bits
Gain(*Windy*) = 0.020 bits

Humidity is selected

Attributes with many values



- E.g., the dates of the examples
- Subsets are more likely to be pure if there is a large number of values
- Information gain is biased towards choosing attributes with a large number of values

→ generalization suffers!

C4.5 heuristics: Gain Ratio

Idea: Measure how broadly and uniformly A splits the data:

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i and c is the number of different values.

Example:

- Attribute 'Date': n examples are completely seperated. Therefore: $SplitInformation(S,'Date') = \log_2 n$
- other extreme: binary attribute splits data set in two even parts: $SplitInformation(S,'Date') = 1$

By considering as a splitting criterion the

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}$$

one relates the Information gain to the way, the examples are split

Gain Ratio - Example

Outlook		Temperature	
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019
Humidity		**Windy**	
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

$$SplitInformation(date) = log_2(14) = 3.807$$

$$GainRatio(date) = \frac{0.94}{3.807} = 0.246$$

→ "date" still wins → one has to be careful about which attributes to add

- However, in general, gain ratio is more reliable than information gain

Hypothesis Space of Decision Trees

- Hypothesis space is a general concept in machine learning, specifying the set of decision functions/hypotheses that a learning algorithm would search for a given training dataset (i.e., the possible hypotheses to be returned).
- The hypothesis space of decision trees is complete
  - Every possible finite discrete-valued function can be represented by some decision tree.
  - Given a training dataset, the target function is surely included.

Properties of ID3, C4.5

- Maintaining a single current hypothesis when searching through the space of decision trees, so only a single hypothesis is returned (some other algorithms maintain the set of all hypotheses consistent with the available training dataset)
- No backtracking during the search (risk of converging to a locally optimal solution).
- Using all training examples at each step to refine the current hypothesis, so less sensitive to errors/noises in training data

Inductive Bias in Decision Tree Learning

- Inductive bias of a learning algorithm: the explicit or implicit prior assumptions on the hypothesis (e.g., restrictions on the hypothesis space – restriction bias).
- For decision tree learning, the inductive bias is imposed on the search strategy, i.e., how to explore the hypothesis space (i.e., preference bias)
  - Preference for short trees over longer trees
  - and for those with high information gain/gain ratio attributes near the root
- Many other learning algorithms (e.g., linear classifiers) exhibit both restriction and preference bias.

Why prefer short hypotheses?

- Occam's razor (1320): Prefer the simplest hypothesis that fits the data

→ among many hypotheses that are consistent/fit with the training data, choose the simplest one.

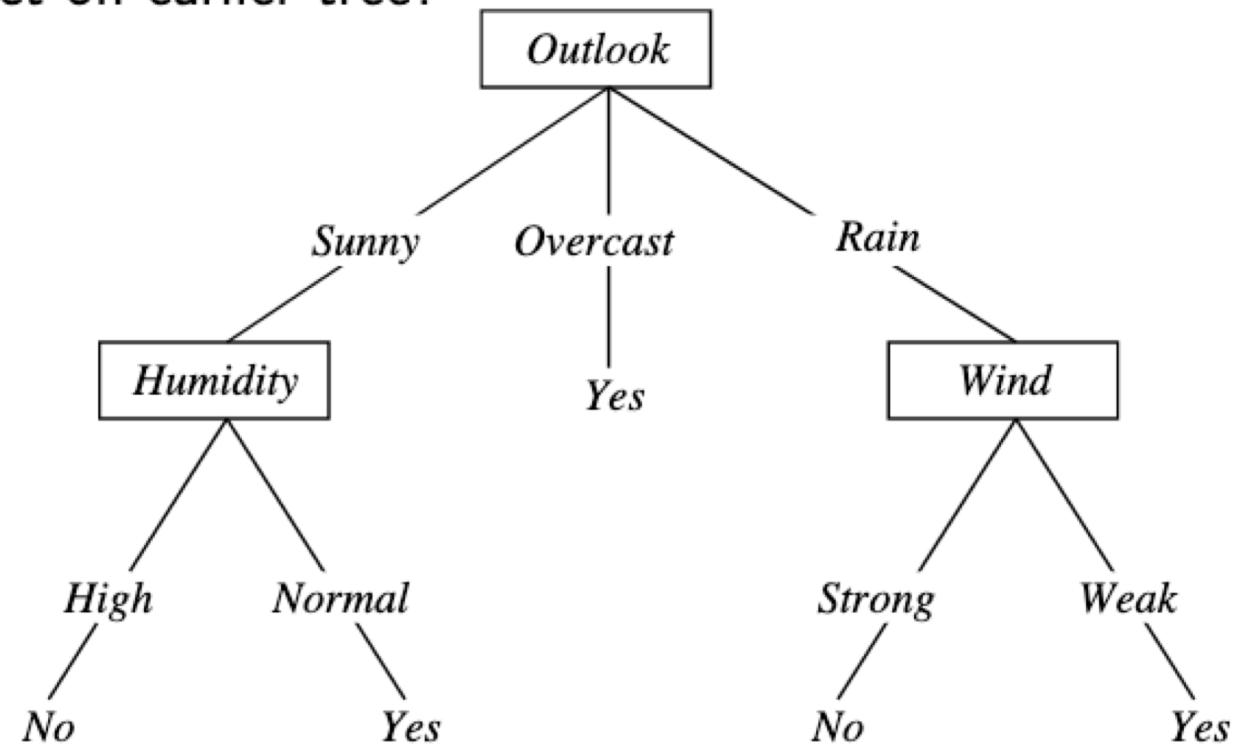
- Argument in favor:
  - Typically, fewer short hypotheses than long ones
  - A short hypothesis that fits data unlikely to be coincidence
  - A long hypothesis that fits data might be coincidence

Noise

Consider adding noisy training example #15:

$$Sunny,\ Mild,\ Normal,\ Weak,\ PlayTennis = No$$

What effect on earlier tree?



Overfitting in Decision Trees

- Algorithm will introduce new attribute test
- The new tree perfectly fits the new training data while the old tree does not.
- However, as the additional training example is a noise, we expect that the old tree still outperforms the new tree on future examples drawn from the same distribution.

→ overfitting: corresponds to learning coincidental regularities

- Unfortunately, we generally don't know which examples are noisy …

Overfitting

Consider error of hypothesis h over

- training data $(\mathbf{x}_1, k_1), \ldots, (\mathbf{x}_d, k_d)$: training error

$$error_{train}(h) = \frac{1}{d} \sum_{i=1}^{d} L(h(\mathbf{x}_i), k_i)$$

with loss function $L(c, k) = 0$ if $c = k$ and $L(c, k) = 1$ otherwise

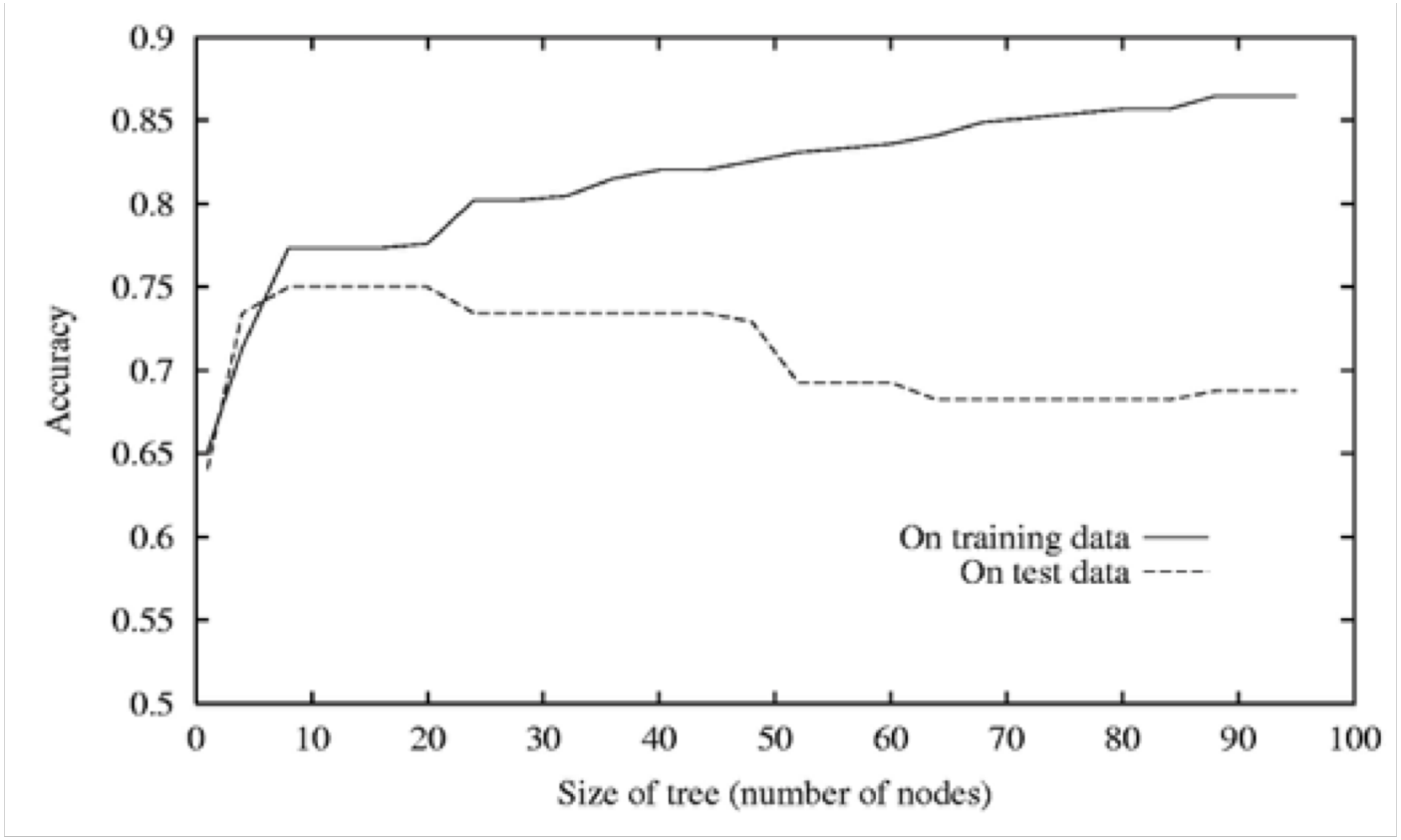
- entire distribution $\mathcal{D}$ of data $(\mathbf{x}, k)$: true error

$$error_{\mathcal{D}}(h) = P(h(\mathbf{x}) \neq k)$$

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

$$error_{train}(h) < error_{train}(h') \quad \text{and} \quad error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

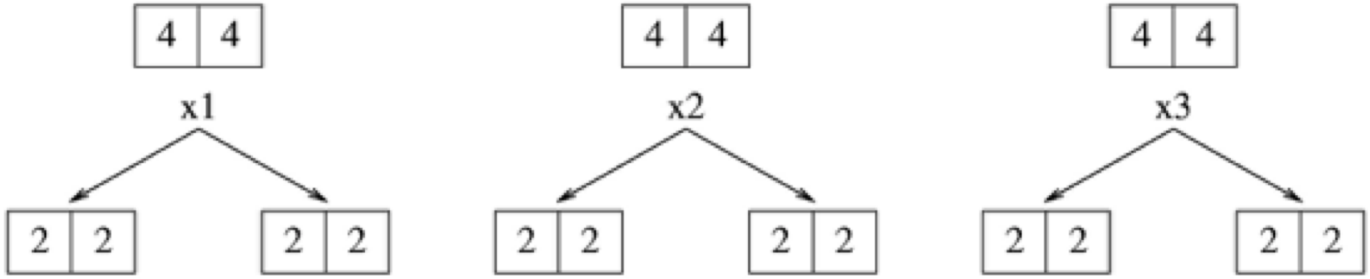
Overfitting in Decision Tree Learning



Learning Parity with Noise

When learning exclusive-or (2-bit parity), all splits look equally good. If extra random boolean features are included, they also look equally good. Hence, decision tree algorithms cannot distinguish random noisy features from parity features.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



Avoiding Overfitting

1. How can we avoid overfitting?
   - Stop growing when data split not statistically significant (pre-pruning)
     - e.g. in C4.5: Split only, if there are at least two descendant that have at least n examples, where n is a parameter
   - Grow full tree, then post-prune (post-prune)
2. How to select "best" tree:
   - Measure performance over training data
   - Measure performance over separate validation data set
   - Minimum Description Length (MDL): minimize $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

1. An example for post-pruning

2. Split data into *training* and *validation* set

3. Do until further pruning is harmful:

   (a) Evaluate impact on *validation* set of pruning each possible node (plus those below it)
   (b) respective node is labeled with most frequent class
   (c) Greedily remove the one that most improves *validation* set accuracy

4. Produces smallest version of most accurate subtree

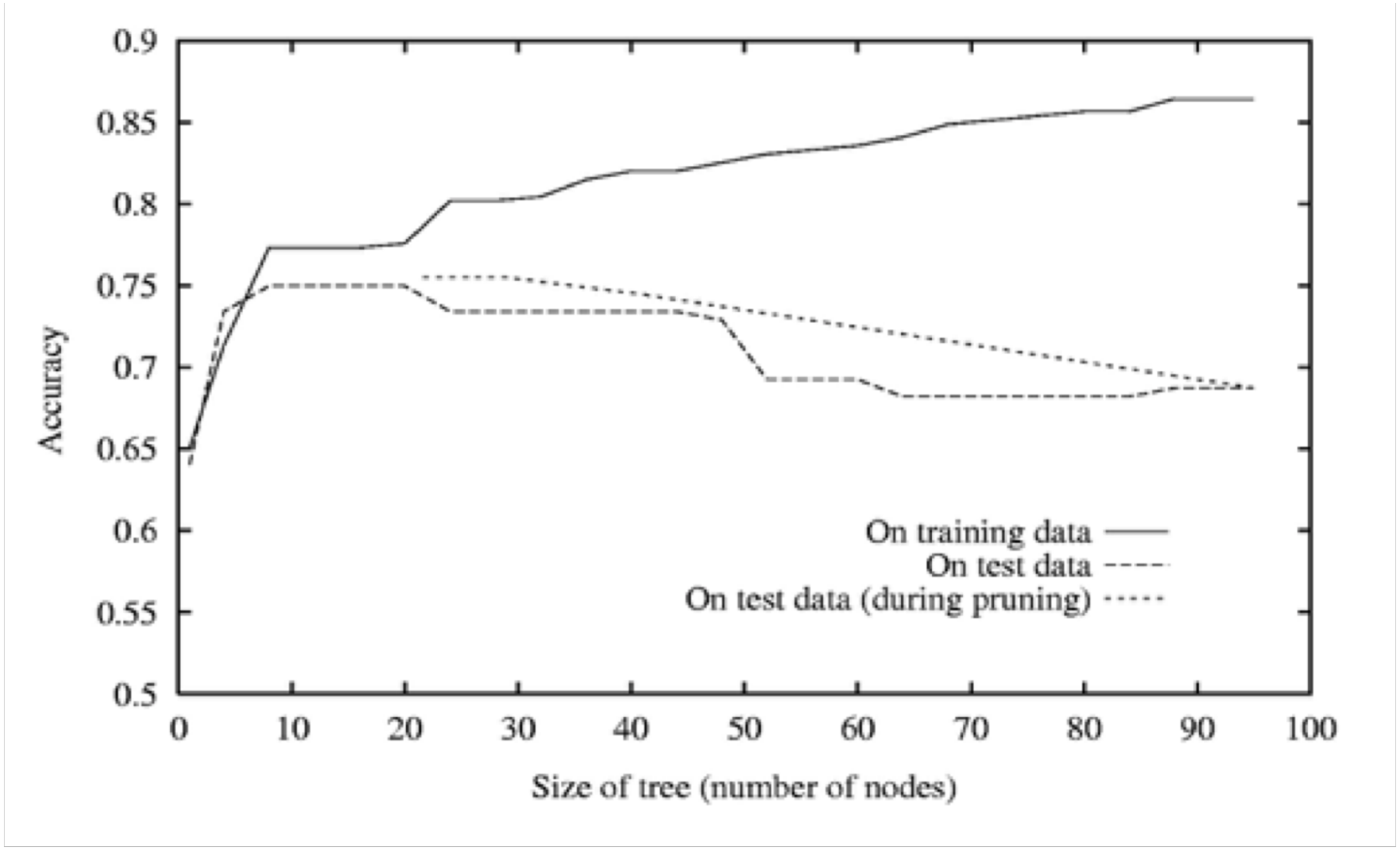
5. What if data is limited?

Rule Post-Pruning

1. Grow tree from given training set that fits data best, and allow overfitting
2. Convert tree to equivalent set of rules
3. Prune each rule independently of others
4. Sort final rules into desired sequence for use

- Perhaps most frequently used method (e.g., C4.5)
- allows more fine grained pruning
- converting to rules increases understandability

Effect of Reduced-Error Pruning



When to consider decision trees

- Instances describable by attribute-value pairs, typically when each attribute takes on a small number of disjoint possible values
- The target outputs have discrete/categorical values (real-valued outputs are possible but less common)
- Disjunctive descriptions may be required
- The training data may contain errors
- Interpretable result of learning is required
- Examples: medical diagnosis, credit risk analysis

Unknown Attribute Values

What if some examples are 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 same fashion