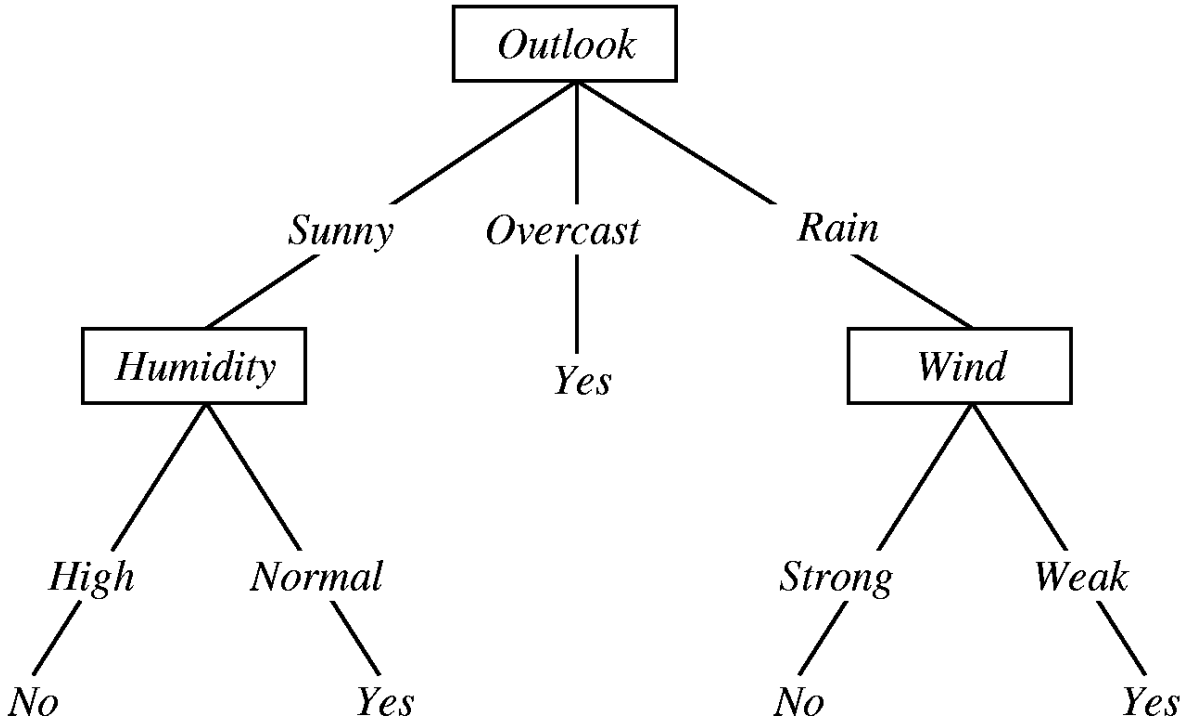
Decision Tree Learning

From multiple sources

Decision Trees

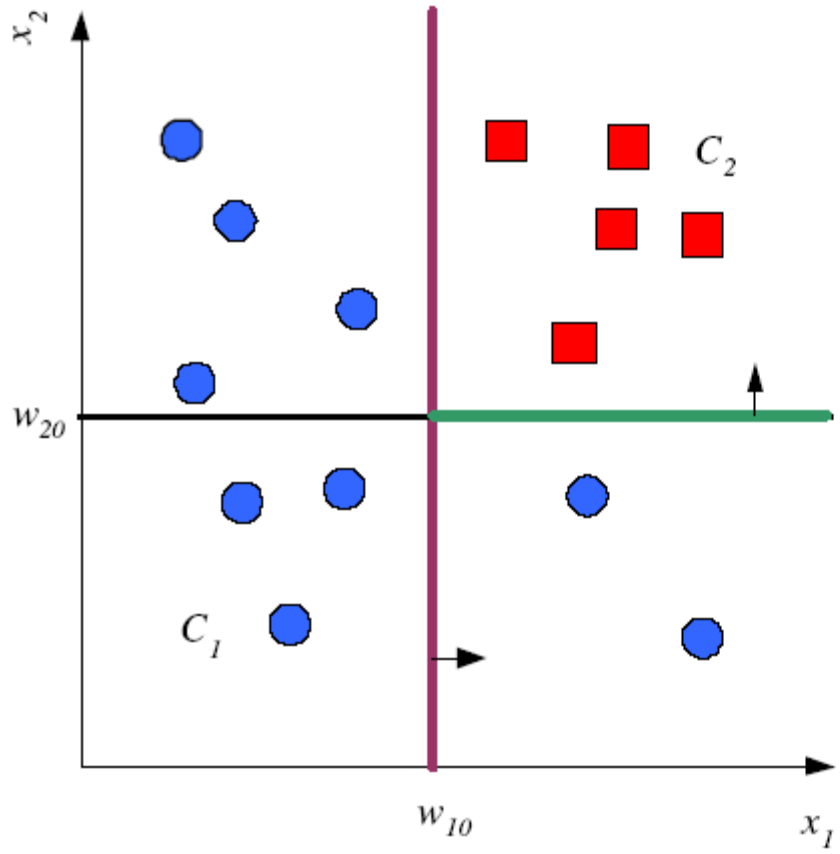
- One of the most widely used and practical methods for inductive inference
- Uses a completely expressive hypothesis space
- Capable of learning disjunctive expressions
- Approximates discrete-valued target functions as trees
- Robust to noisy data
- A family of decision tree learning algorithms include ID3, C4.5
- Can be used for classification (most common) or regression problems

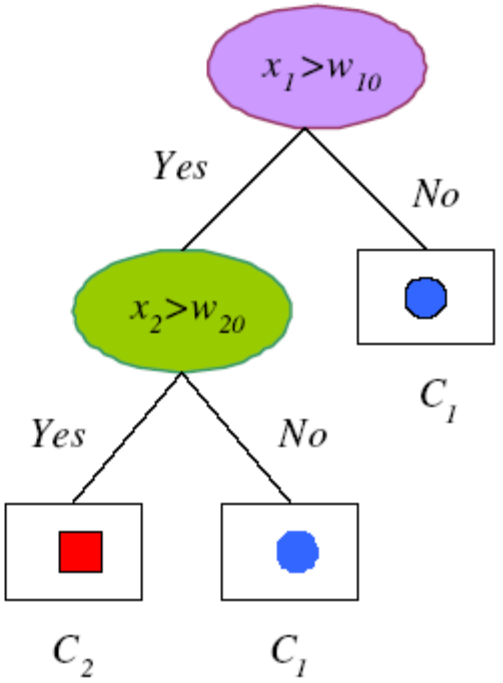
Decision Tree Example:Aldo playing Tennis



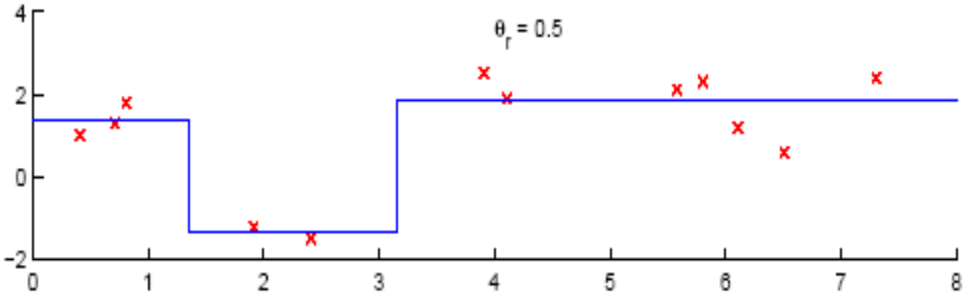
- Each internal node corresponds to a test
- Each branch corresponds to a result of the test
- Each leaf node assigns a classification

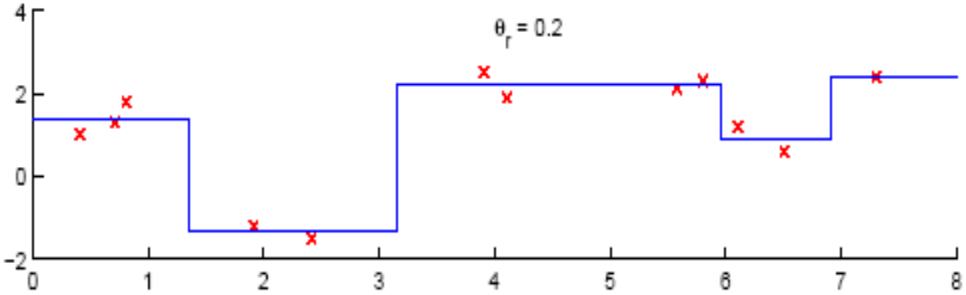
Decision Regions

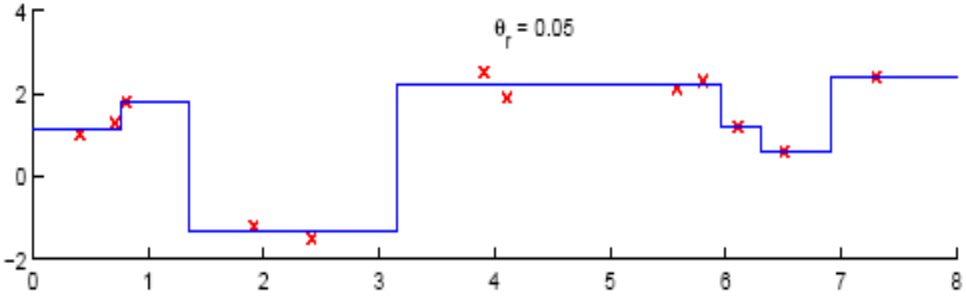


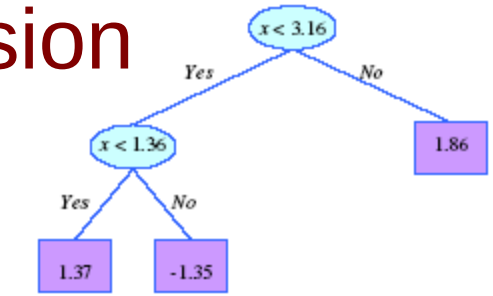


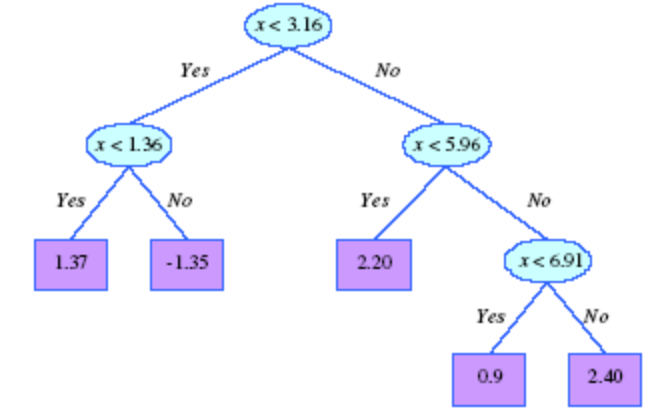
Decision Trees for Regression

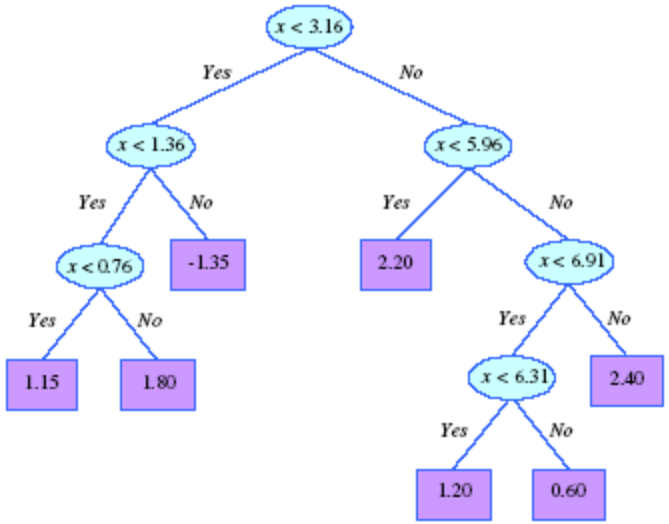












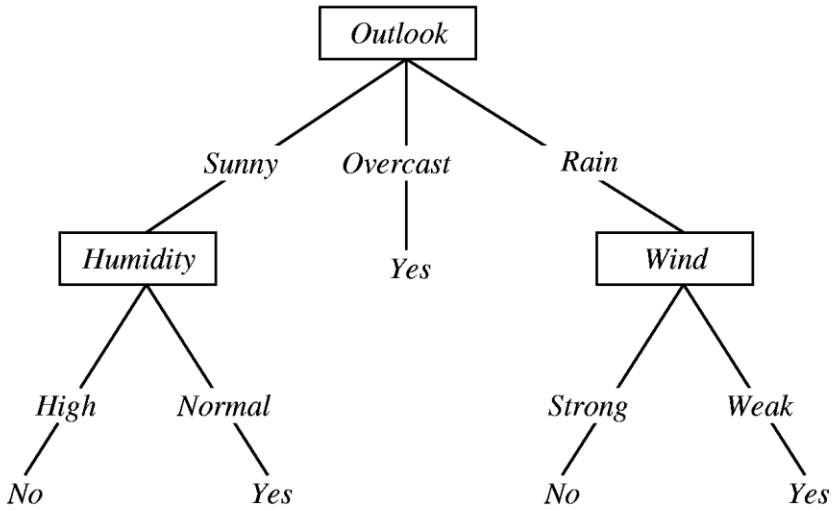
Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Discrete x_i : n-way split for n possible values
    - Continuous x_i : Binary split : $x_i > w_m$
  - Multivariate: Uses more than one attributes
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Once the tree is trained, a new instance is classified by starting at the root and following the path as dictated by the test results for this instance.

Expressiveness

- A decision tree can represent a disjunction of conjunctions of constraints on the attribute values of instances.
  - Each path corresponds to a conjunction
  - The tree itself corresponds to a disjunction

Decision Tree



If (O=Sunny AND H=Normal) OR (O=Overcast) OR (O=Rain AND W=Weak) then YES

- "A disjunction of conjunctions of constraints on attribute values"

- How expressive is this representation?
- How would we represent:
  - (A AND B) OR C
  - A XOR B
- It can represent any Boolean function

Decision tree learning algorithm

- For a given training set, there are many trees that code it without any error
- Finding the smallest tree is NP-complete (Quinlan 1986), hence we are forced to use some (local) search algorithm to find reasonable solutions
- Learning is greedy; find the best split **recursively** (Breiman et al, 1984; Quinlan, 1986, 1993)
- If the decisions are binary, then in the best case, each decision eliminates half of the regions (leaves).
- If there are b regions, the correct region can be found in $log_2 b$ decisions, in the best case.

The basic decision tree learning algorithm

- A decision tree can be constructed by considering attributes of instances one by one.
  - Which attribute should be considered first?

- The height of a decision tree depends on the order of the attributes that are considered.

- An attribute is good when
  - For one value we get all instances as positive
  - For other value we get all instances as negative
- An attribute is poor when
  - It provides no discrimination
  - Attribute is immaterial to the decision
  - For each value we have the same number of positive and negative instances

Top-Down Induction of Decision Trees

Main loop:

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 of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

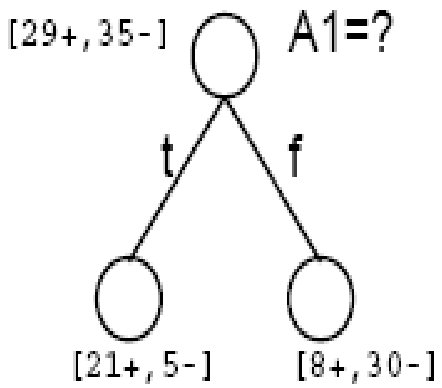
What Attribute to choose to "best" split a node?

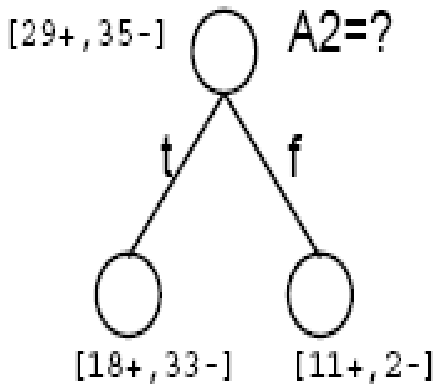
- Choose the attribute that minimize the **Disorder (or Entropy)** in the subtree rooted at a given node
- **Disorder** and **Information** are related as follows: the more disorderly a set, the more information is required to correctly guess an element of that set.
- **Information:** What is the best strategy for guessing a number from a finite set of possible numbers?
  - how many questions do you need to ask in order to know the answer (we are looking for the minimal number of questions)
  - Answer Log_2(S), where S is the set of numbers and |S|, its cardinality

E.g.: 0 1 | 2 3 4 5 6 | 7 8 9 10

Q2 Q1

Which attribute is best?



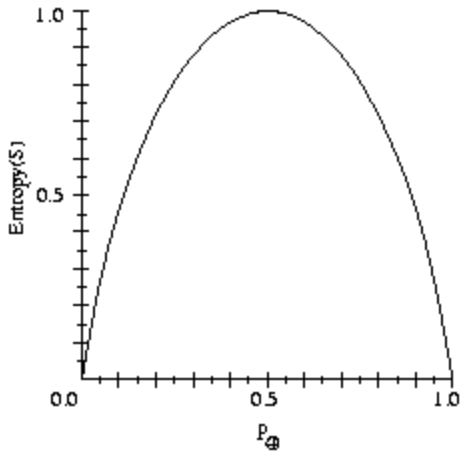


Entropy

- Entropy of a random variable with multiple possible values x is defined as:

$$\mathrm{H}[x] = -\sum_{x} p(x) \log_2 p(x)$$

- Characterizes the (im)purity of the arbitrary collection of examples
- Measure of uncertainty
- Measure of homogeneity of examples
- is 0 if all members belong to same class
- Entropy function relative to a boolean function



Entropy

Example from Coding theory:

Random variable x discrete with 8 possible states; how many bits are needed to transmit the state of x?

1. All states equally likely

$$\mathrm{H}[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

2. We have the following distribution for x?

x	a	b	c	d	e	f	g	h
$p(x)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$
code	0	10	110	1110	111100	111101	111110	111111

$$\begin{aligned} \mathrm{H}[x] &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - \frac{4}{64}\log_2\frac{1}{64} \\ &= 2 \text{ bits} \end{aligned}$$

Use of Entropy in Choosing the Next Attribute

- We will use the entropy of the remaining tree as our measure to prefer one attribute over another.

- In summary, we will consider
  - the entropy over the distribution of samples falling under each leaf node and
  - we will take a weighted average of that entropy – weighted by the proportion of samples falling under that leaf.

- We will then choose the attribute that brings us the biggest information gain, or equivalently, results in a tree with the lower weighted entropy.

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

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

Algorithm ID3(examples, target_attr, attributes)

- Examples are the training examples.
- target_attr is the attribute whose value is to be predicted
- attributes : list of attributes that may be tested by the learned decision tree.
- Returns a decision tree (the root node) that correctly classifies the given Examples.
- { Create a *Root* node for the tree
- 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 t*arget_attr* in *Examples*
- *%will return the name of a feature at this point*

ID3 contd.

- Otherwise Begin
  - A ←the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute (feature) for *Root*←A
  - For each possible value *vi*, of A,
  - Add a new tree branch below *Root*, corresponding to test A = *vi*
  - Let $Examples_{vi}$ the subset of *Examples* that have value *vi* for A
  - If $Examples_{vi}$ is empty then
    - below this new branch, add a leaf node with label = most common value of *target_attr* in *Examples*
  - Else
    - below this new branch add the subtree

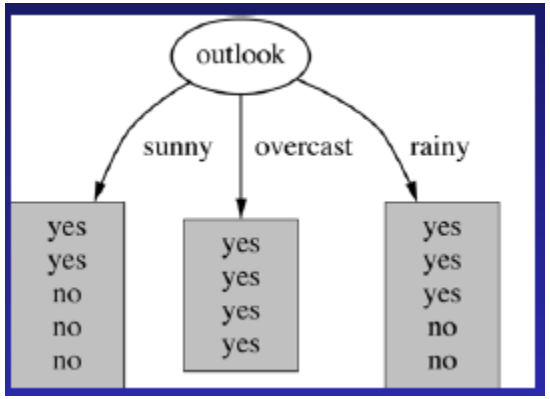
      ID3($Examples_{vi}$, *target_attr*, *attributes*-{A})
- End
- Return *Root }*
- The best attribute is the one with the highest *information gain*, as defined in

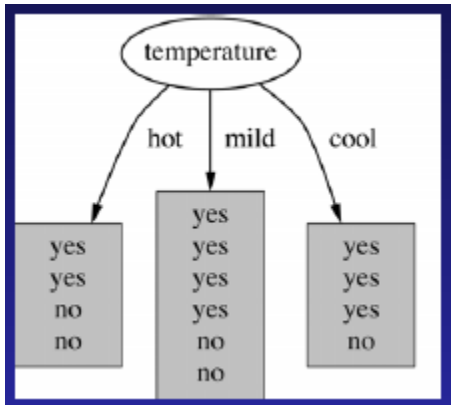
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

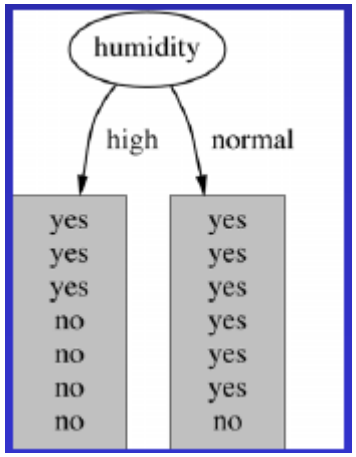
Training Examples

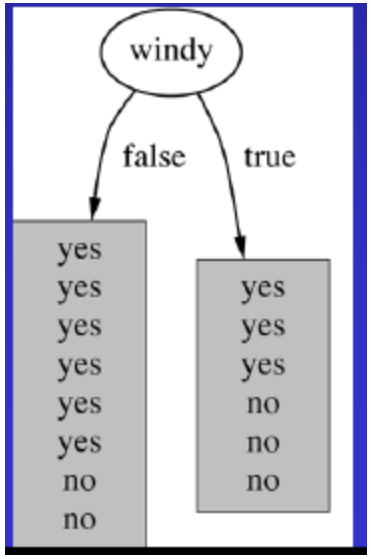
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

Four Tree stumps



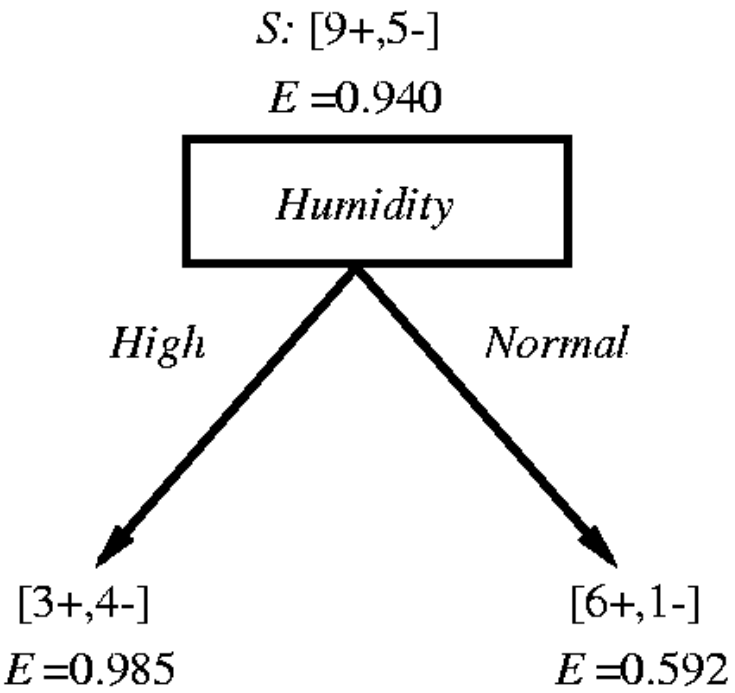


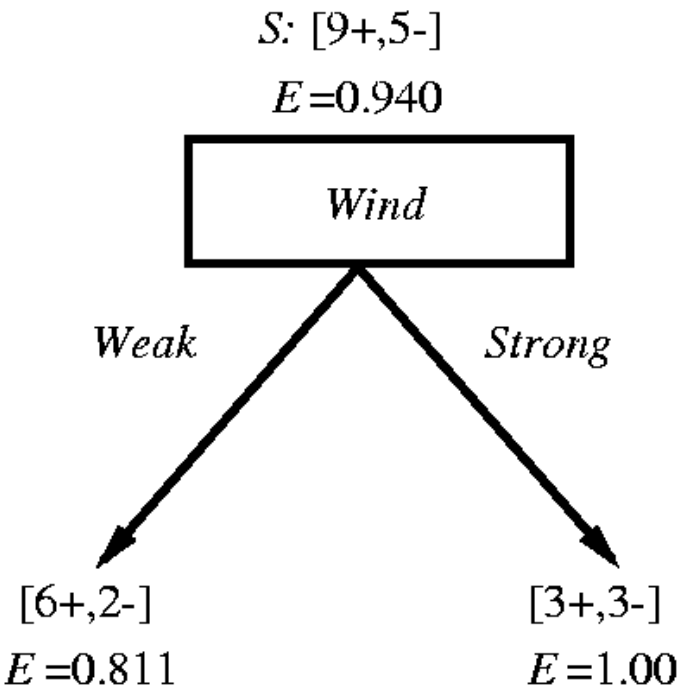




- Gain(S, Outlook) = 0.246
- Gain(S,Humidity)= 0.151
- Gain(S,Wind) = 0.048
- Gain(S,Temperature) = 0.029
- Outlook is the root node

Which attribute is the best classifier?





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

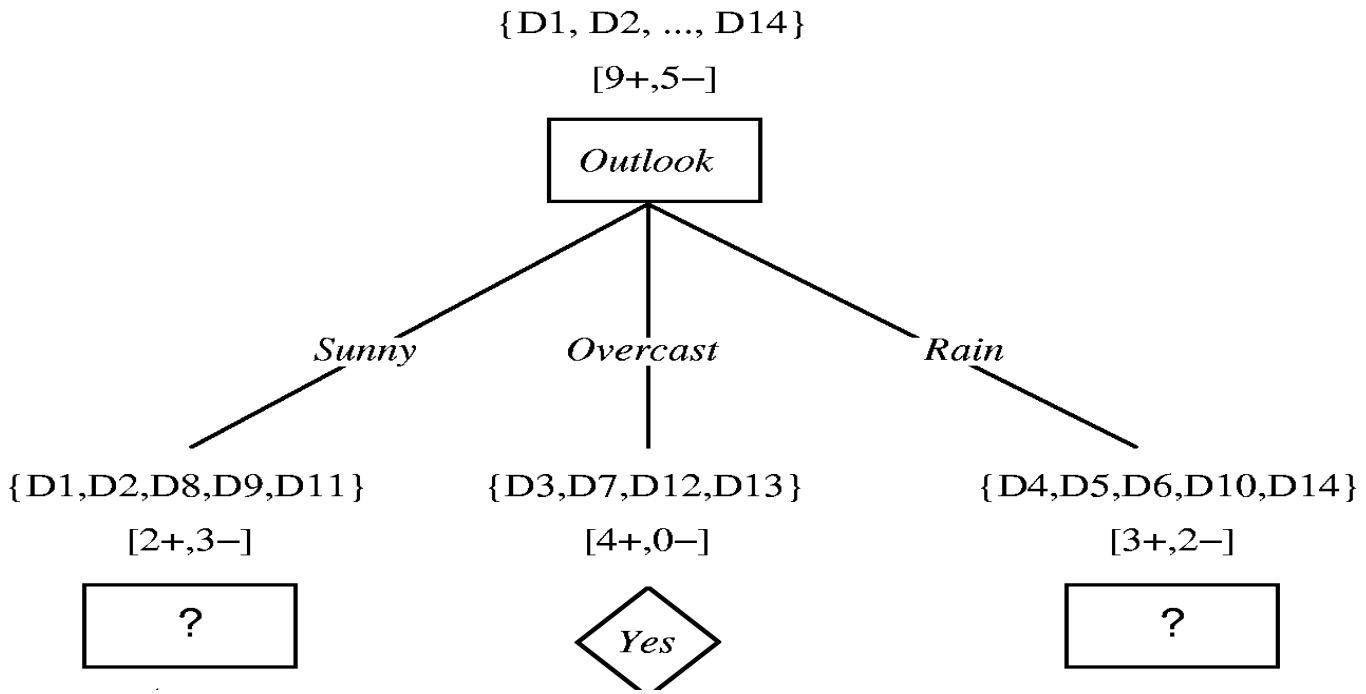
Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

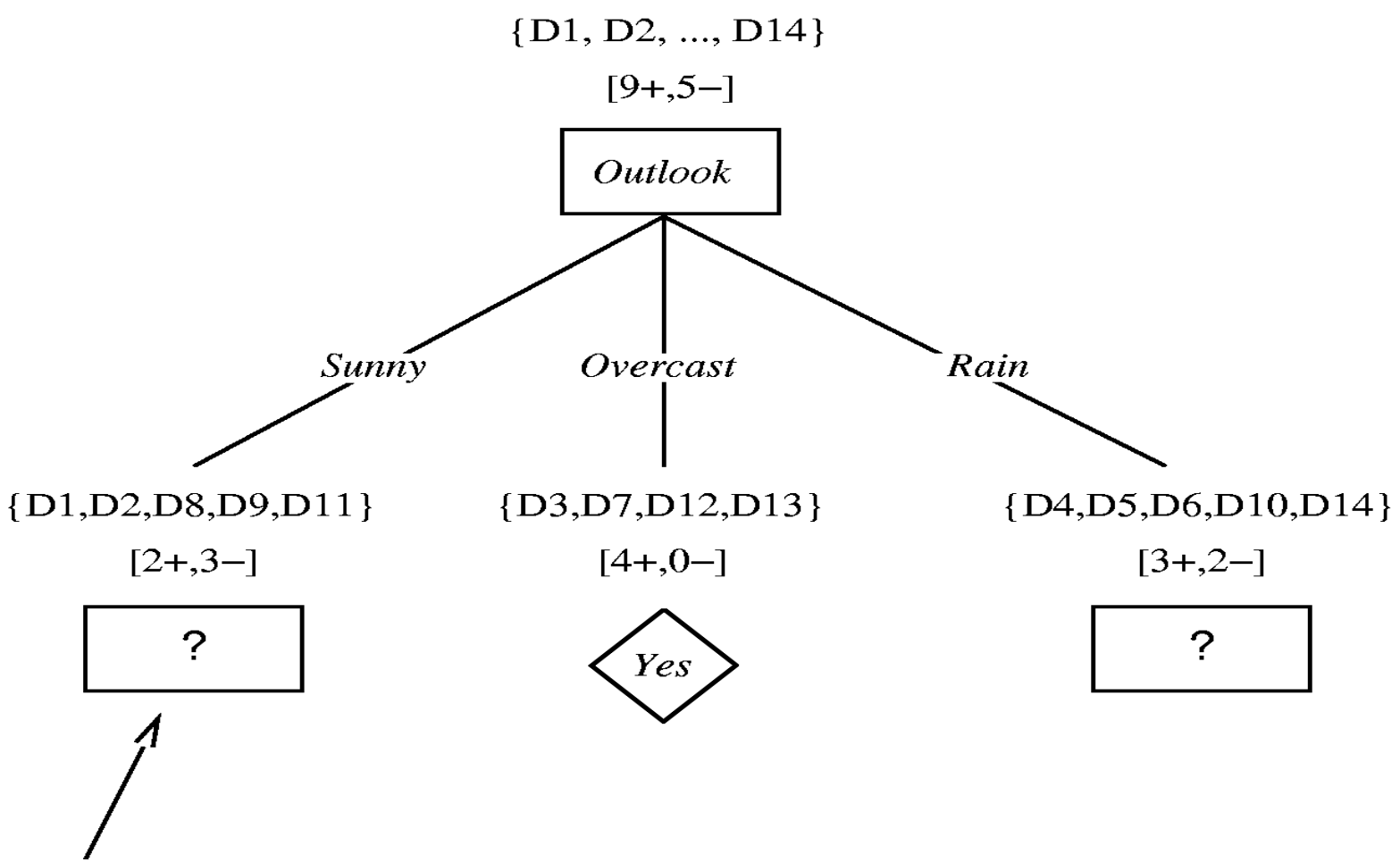
We would select the Outlook attribute to split the root node as it has a higher Information Gain

Selecting the Next Attribute

- Computing the information gain for each attribute, we selected the *Outlook* attribute as the first test, resulting in the following partially learned tree:



- We can repeat the same process recursively, until Stopping conditions are satisfied.



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Until stopped:

- Select one of the unused attributes to partition the remaining examples at each non-terminal node
- using only the training samples associated with that node

Stopping criteria:

- each leaf-node contains examples of one type
- algorithm ran out of attributes

Hypothesis Space Search in Decision Tree Learning

- **Hypothesis Space**: Set of possible decision trees (i.e., complete space of finite discrete-valued functions).
- **Search Method**: Simple-to-Complex *Hill-Climbing* Search (only a single current hypothesis is maintained **No Backtracking!!!**
- **Evaluation Function**: Information Gain Measure
- **Batch Learning:** ID3 uses all training examples at each step to make statistically-based decisions ==> the search is less sensitive to errors in individual training examples.

Inductive Bias in Decision Tree Learning

- **ID3's Inductive Bias:** *Shorter* trees are preferred over longer trees. Trees that place *high information gain attributes close to the root* are preferred over those that do not.

Why Prefer Short Hypotheses?

- **Occam's razor:**
  Prefer the simplest hypothesis that fits the data
  [William of Occam (Philosopher)]
- Scientists seem to do that: Eg., Physicist seem to prefer simple explanations for the motion of planets, over more complex ones
- **Argument:** Since there are fewer short hypotheses than long ones, it is less likely that one will find a short hypothesis that coincidentally fits the training data.
- **Problem with this argument:** it can be made about many other constraints. Why is the "short description" constraint more relevant than others?
- **Nevertheless:** Occam's razor was shown experimentally to be a successful strategy!

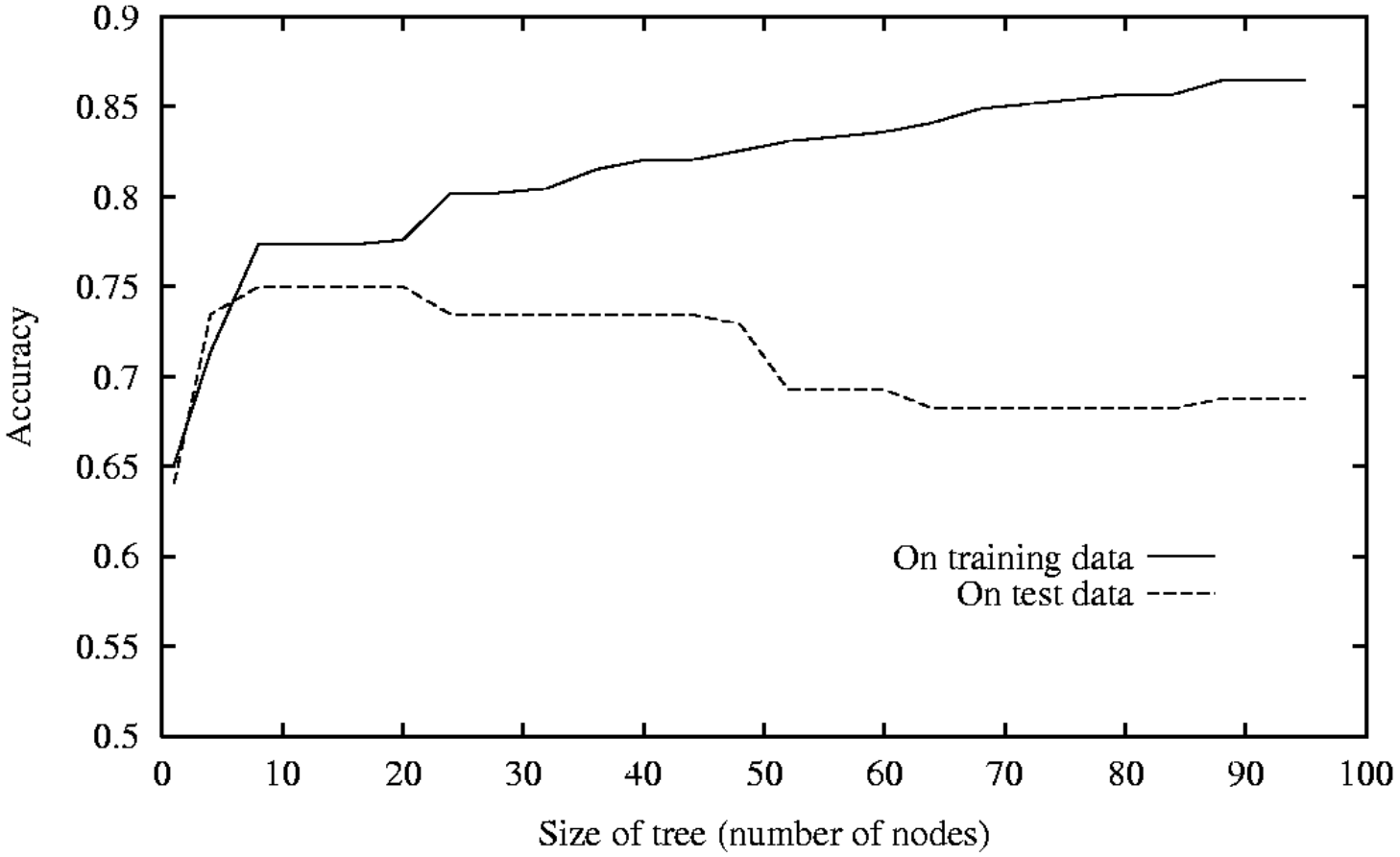
Appropriate Problems for DT Learning

- Instances are represented by discrete attribute-value pairs
  - Each attribute takes on a small number of disjoint possible values (hot, mild, cold)
  - Extensions allow real-valued attributes as well eg. temperature
- The target function has discrete output values
  - Eg. Boolean classification (yes/no)
  - Easily extended to multiple value functions
  - Can be extended to real-valued outputs as well
- Disjunctive hypothesis descriptions may be required
- The training data may contain errors
- The training data may contain missing attribute values
- Examples: Medical Diagnosis, equipment malfunction, loan approval etc.

Issues in Decision Tree Learning: 1. Overfitting the Data

- Why "over"-fitting?
  - A model can become more complex than the true target function (concept) when it tries to satisfy noisy data as well.
- **Definition:** Given a hypothesis space *H*, a hypothesis $h \in H$ is said to ***overfit*** the training data if there exists some alternative hypothesis $h' \in H$, such that *h* has smaller error than *h'* over the training examples, but *h'* has a smaller error than *h* over the entire distribution of instances.

Over fitting in Decision Trees



Causes for overfitting

- ID3 (the Greedy algorithm that was outlined) will make a new split and will classify future examples following the new path as negative.

- Problem is due to ”overfitting” the training data which may be thought as insufficient generalization of the training data
  - Coincidental regularities in the data
  - Insufficient data (Training set too small)
  - Noisy data
  - Differences between training and test distributions

Problems due to Overfitting :

Classification Errors

Decision trees may obtain zero error on a set of training examples, but may misclassify examples not contained in the training set.

Training error: probability of error for training examples

True error: probability of error for unrestricted examples

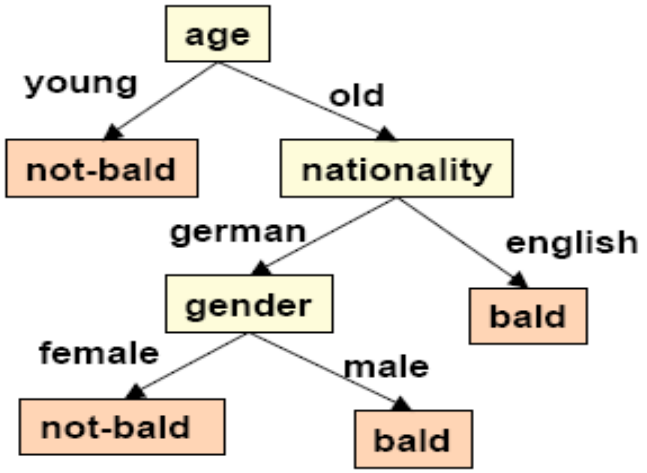
Overfitting of a decision tree to training data is an important source for errors.

Example:

male	young	german	not-bald
female	old	german	not-bald
male	old	english	bald
male	old	german	bald

Unfortunately, old english females will be classified as bald.

Overfitting is due to insufficient generalization of examples.



10

From: http://kogs-www.informatik.uni-hamburg.de/~neumann/WMA-WS-2007/WMA-10.pdf

Problems due to overfitting : noisy data

Inconsistent Example Sets

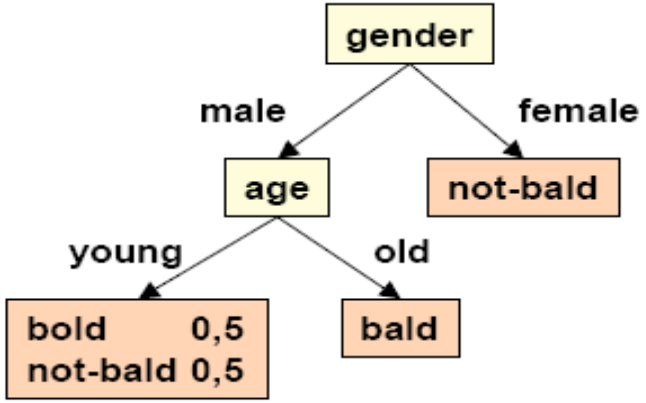
An example set is inconsistent if it contains examples which differ only by the value of the goal attribute.

Reasons for inconsistency:

- too few or irrelevant attributes
- noisy data

Decision trees may reflect inconsistent examples by assigning probability values to conflicting outcomes at leaf nodes.

e1	male	young	not-bald
e2	female	old	not-bald
e3	male	old	bald
e4	female	young	not-bald
e5	male	young	bald

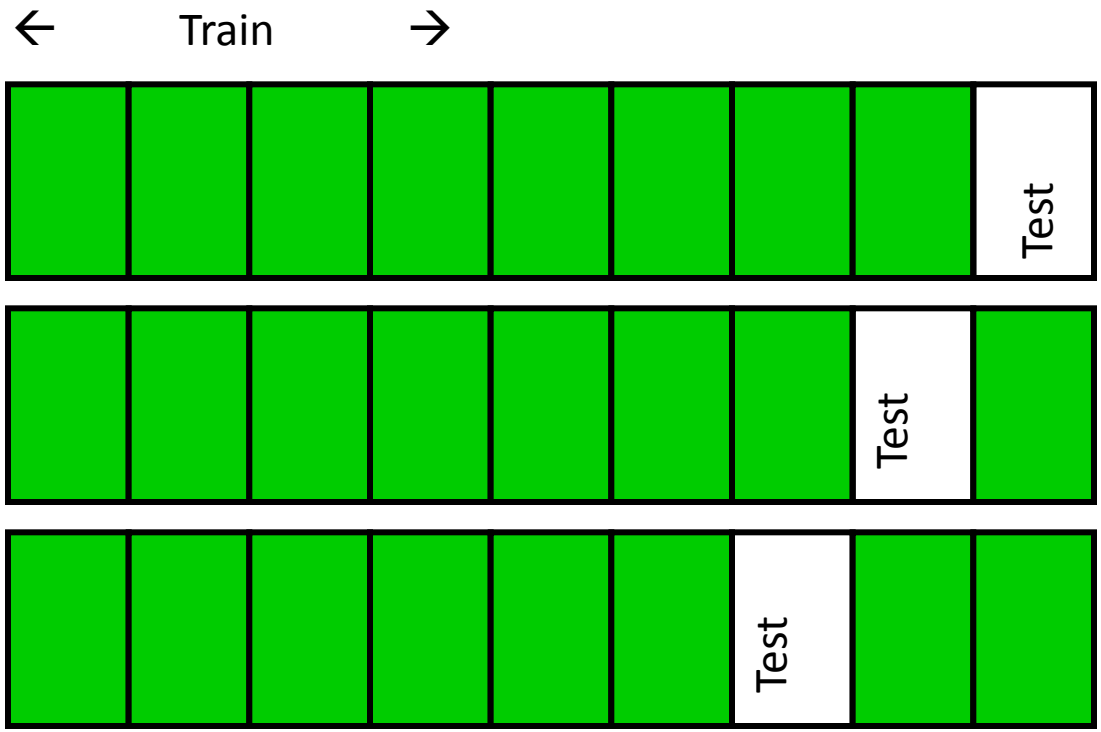


Avoiding over-fitting the data

- How can we avoid overfitting? There are 2 approaches:
  1. **Early stopping**: stop growing the tree before it perfectly classifies the training data
  2. **Pruning:** grow full tree, then prune
     - Reduced error pruning
     - Rule post-pruning
  - Pruning approach is found more useful in practice.
- Whether we do pre or post-pruning, the important question is how to select “best” tree:
  - Measure performance over separate validation data set
  - Measure performance over training data
    - apply a statistical test to see if expanding or pruning would produce an improvement beyond the training set (Quinlan’86)
  - Mininimum Description Length : minimize *size*(*tree*) + *size*(*misclassifications*(*tree*))

Evaluation: Cross Validation

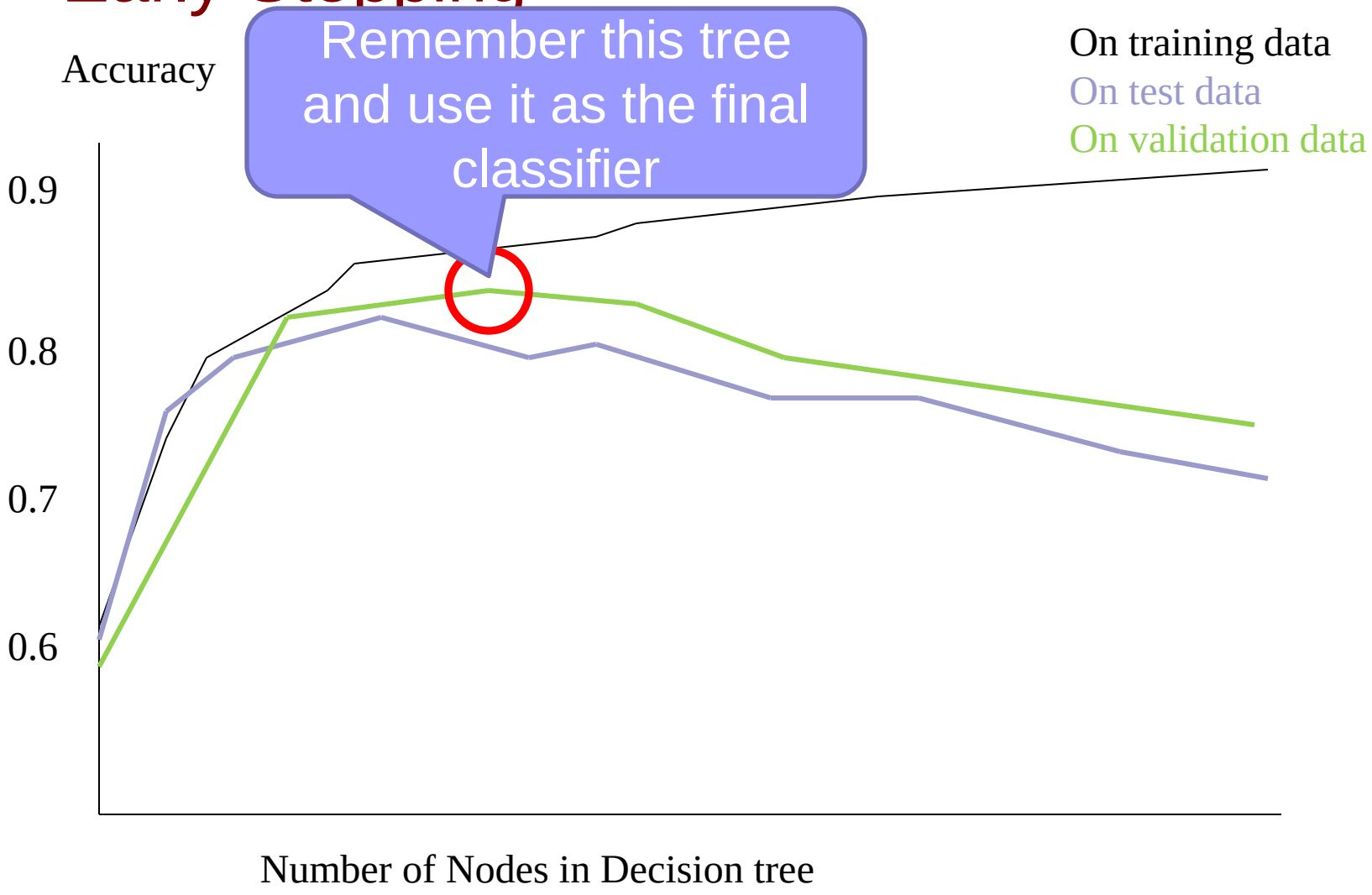
- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



Cross-Validation (2)

- Leave-one-out
  - Use if < 100 examples (rough estimate)
  - Hold out one example, train on remaining examples
- M of N fold
  - Repeat M times
  - Divide data into N folds, do N fold cross-validation

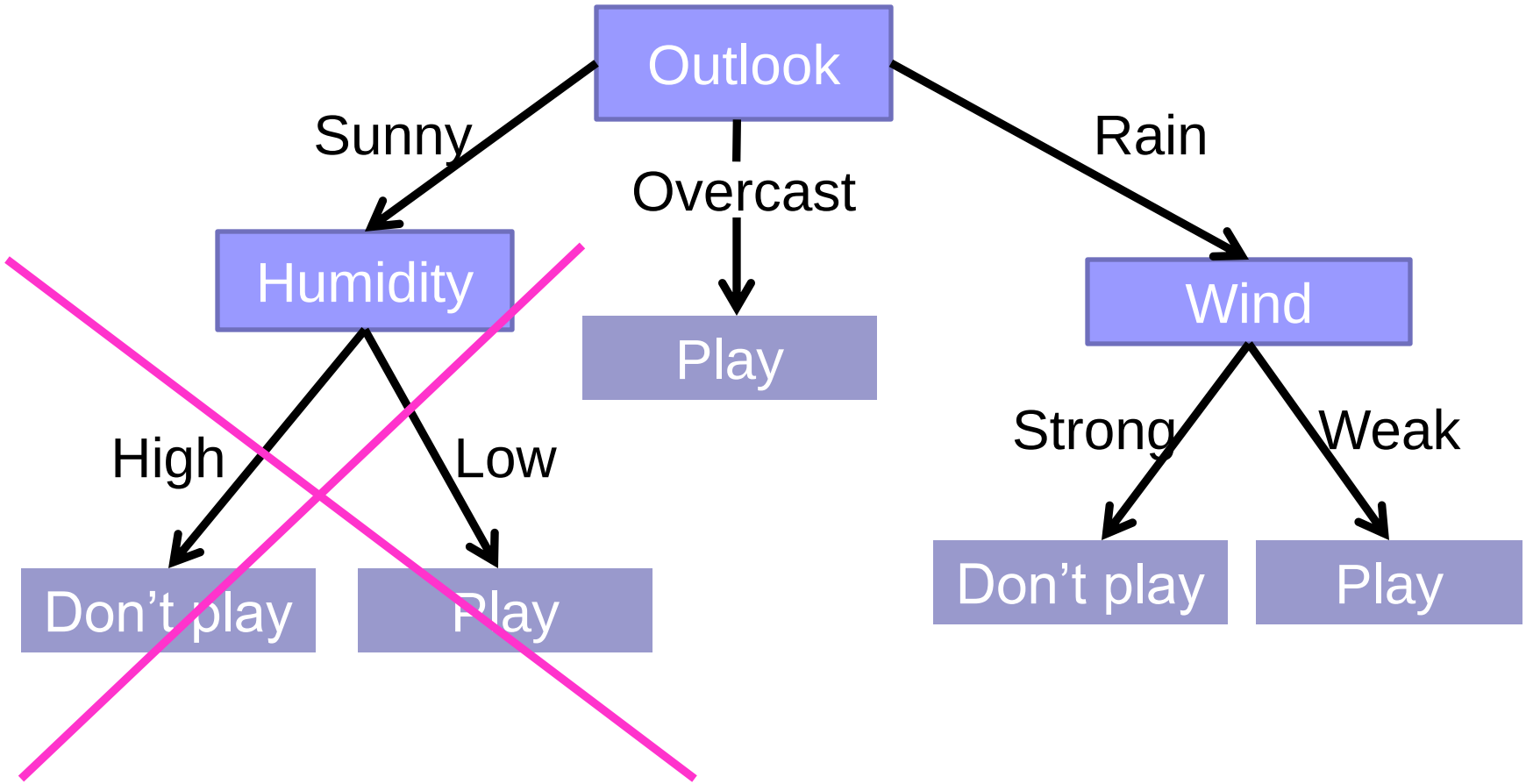
Early Stopping



Reduced-Error Pruning (Quinlan 1987)

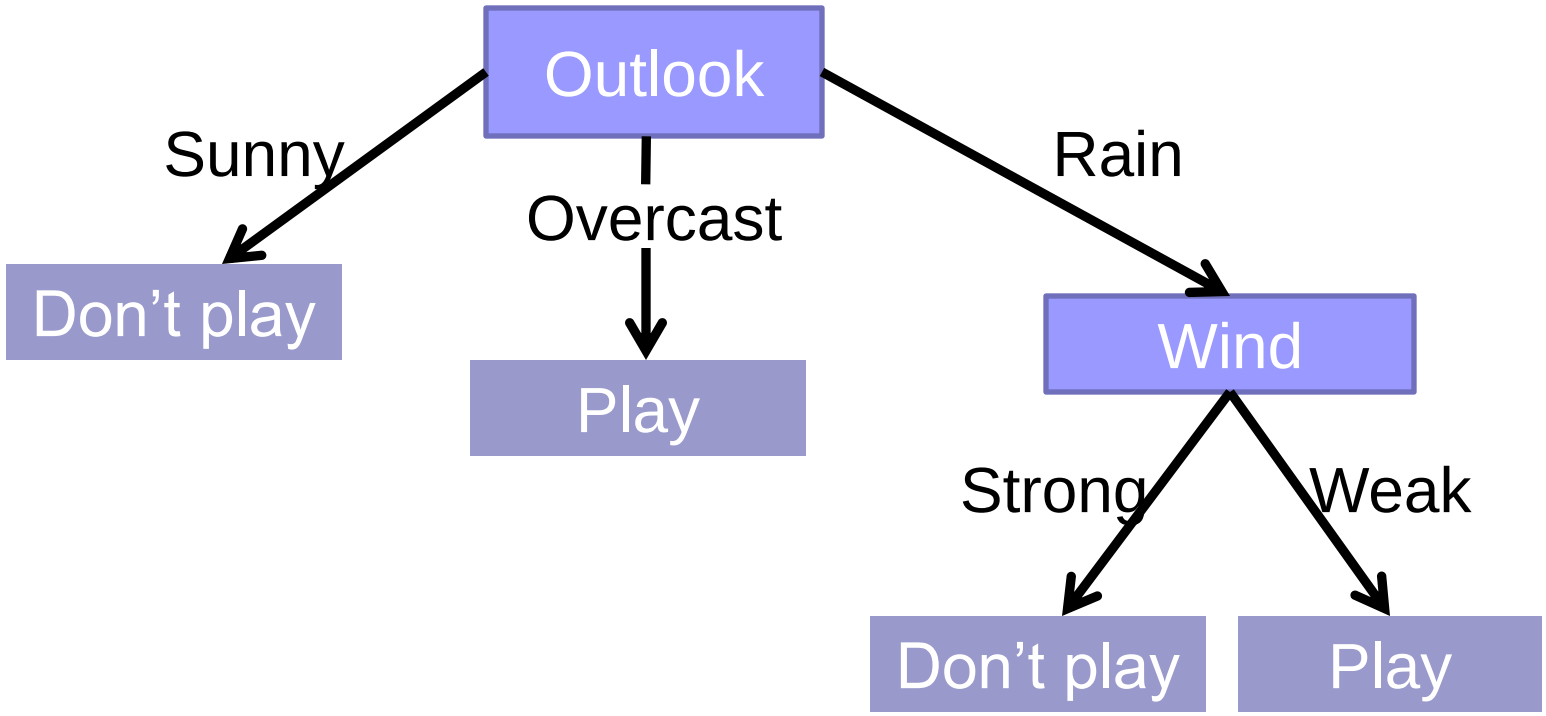
- Split data into *training* and *validation* set
- Examine each decision node to see if pruning decreases the tree's performance over the evaluation data
- "Pruning" here means replacing a subtree with a leaf with the most common classification in the subtree.
- Do until further pruning is harmful:
  - Remove each subtree and replace it with majority class and evaluate on validation set
  - Remove subtree that leads to largest gain in accuracy
- Produces smallest version of the (most accurate) tree
- What if data is limited?
  - We would not want to separate a validation set.

Reduced Error Pruning Example



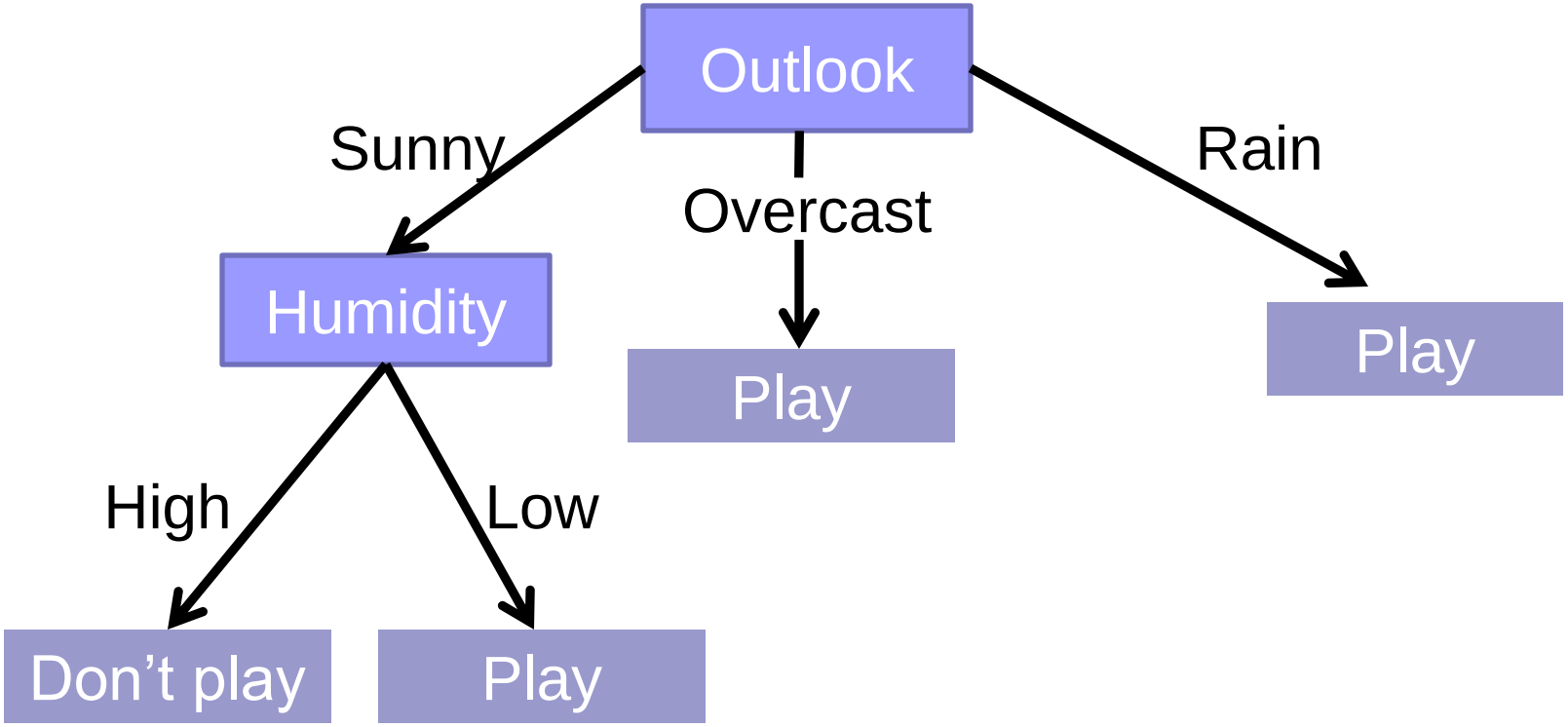
Validation set accuracy = 0.75

Reduced Error Pruning Example



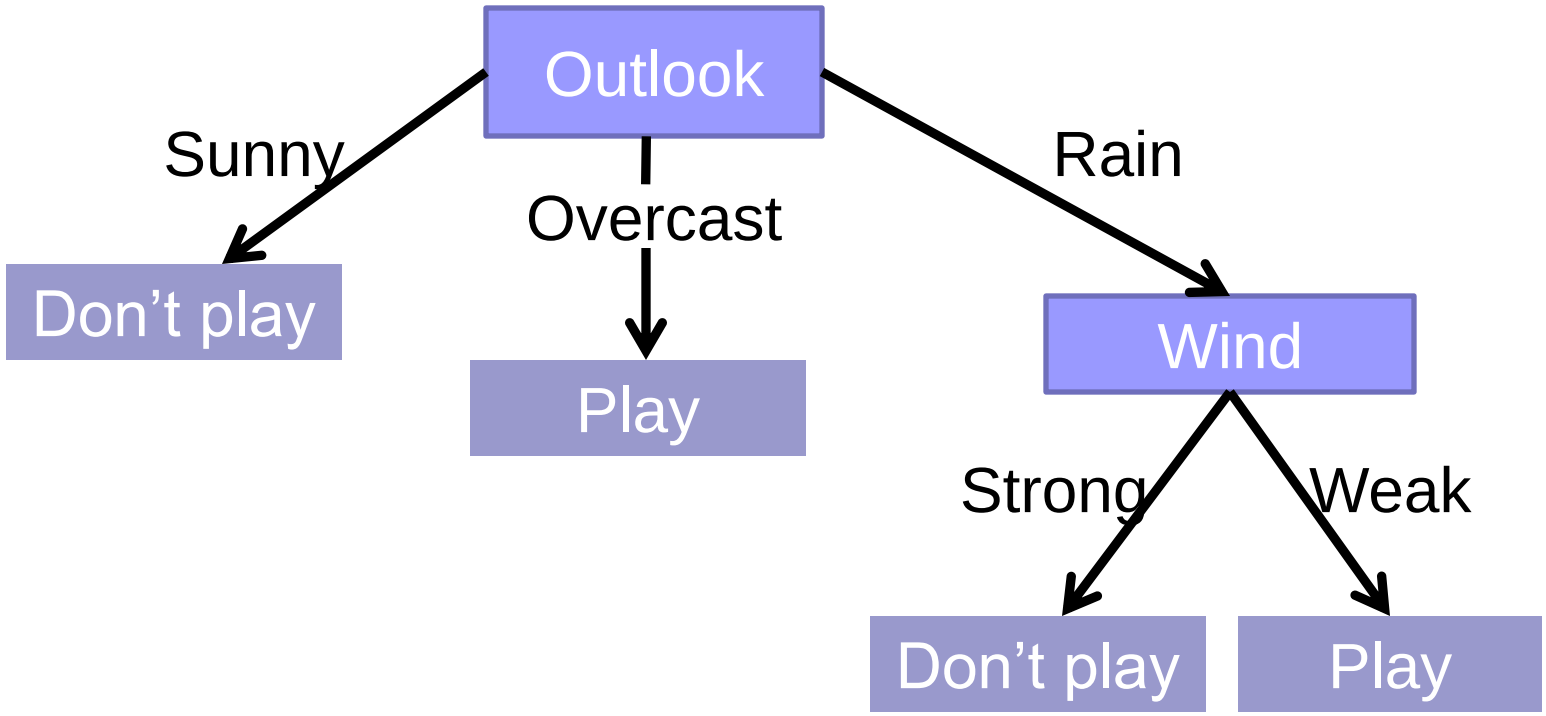
Validation set accuracy = 0.80

Reduced Error Pruning Example



Validation set accuracy = 0.70

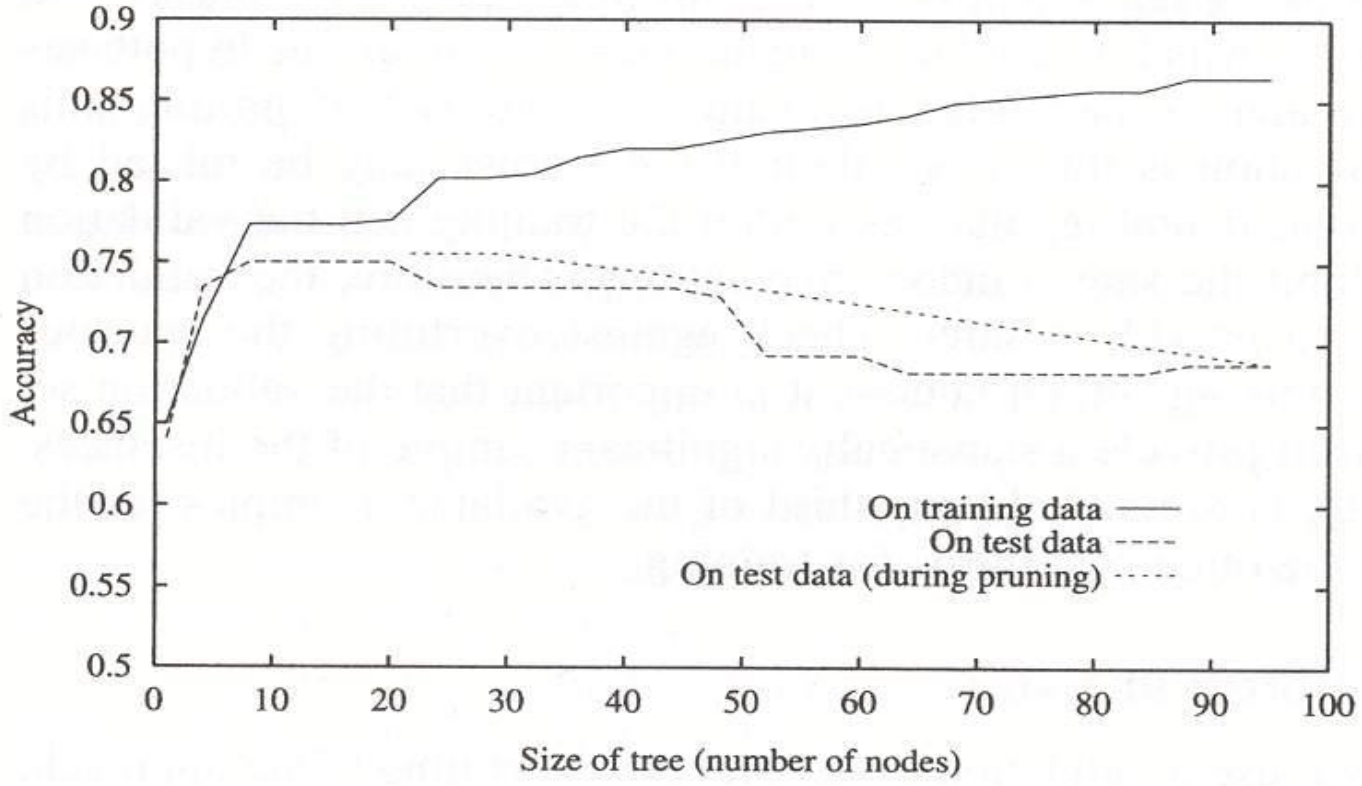
Reduced Error Pruning Example



Validation set accuracy = 0.80

Use this as final tree

Reduced error pruning



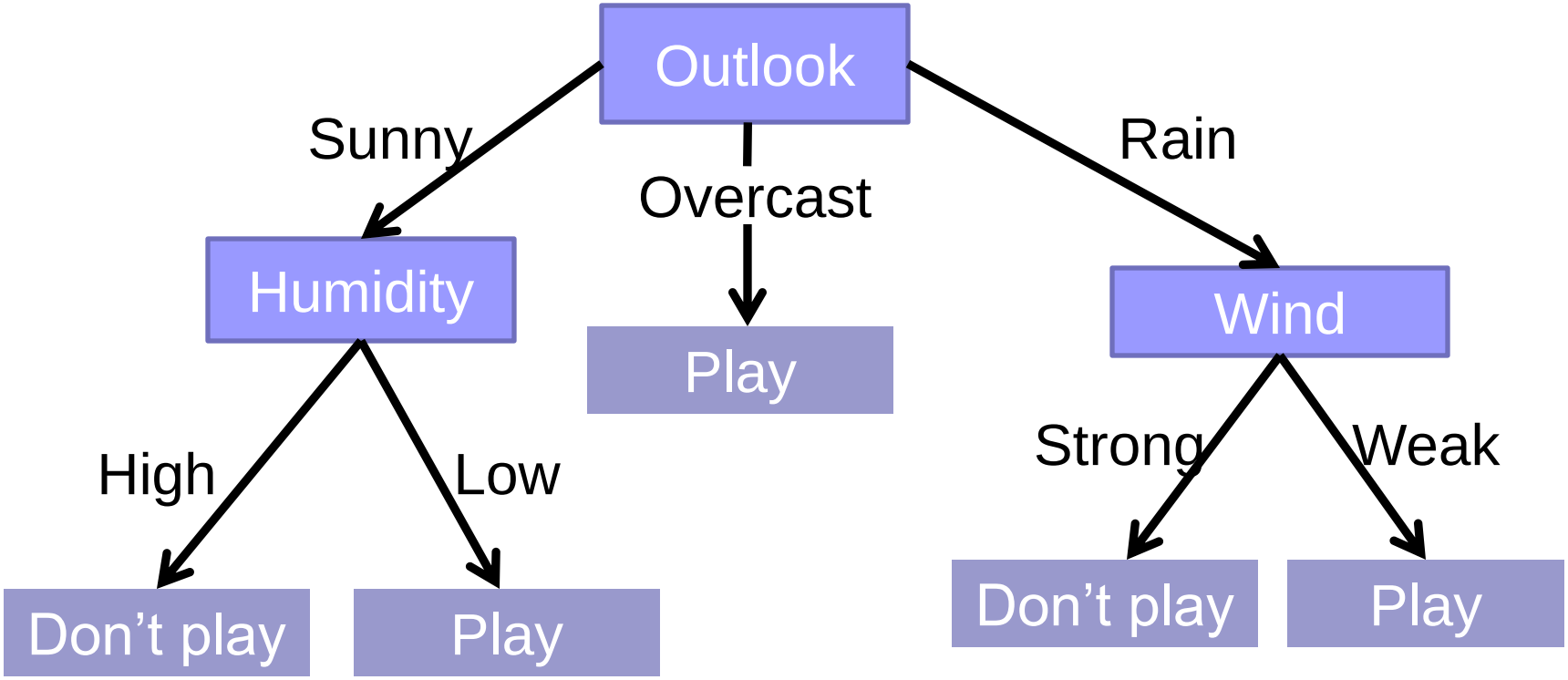
Fighting Overfitting: Using Rule Post-Pruning

1. Grow decision tree. Fit as much data as possible. Allow overfitting.
2. Convert tree to equivalent set of rules. One rule for each path from root to leaf.
3. Prune (generalize) each rule independently of others. i.e. delete preconditions that improve its accuracy.
4. Sort final rules into desired sequence for use depending on accuracy.
5. Use ordered sequence for classification.

Post Rule Pruning

- Split data into train and validation set
- Prune each rule independently
  - Remove each pre-condition and evaluate accuracy
  - Pick pre-condition that leads to largest improvement in accuracy
- Note: ways to do this using training data and statistical tests

Conversion to Rule



Outlook = Sunny ∧ Humidity = High ⇒ Don't play

Outlook = Sunny ∧ Humidity = Low ⇒ Play

Outlook = Overcast ⇒ Play

…

Example

Outlook = Sunny ∧ Humidity = High ⇒ Don't play

Validation set accuracy = 0.68

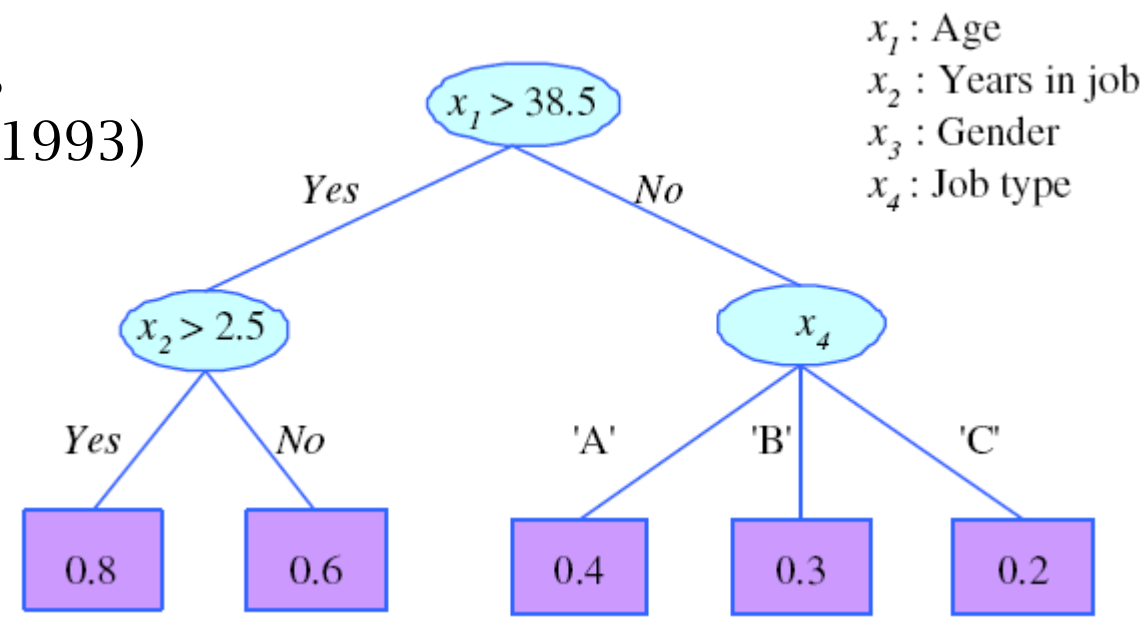
→ Outlook = Sunny ⇒ Don't play Validation set accuracy = 0.65

→ Humidity = High ⇒ Don't play Validation set accuracy = 0.75

Keep this rule

Rule Extraction from Trees:Post pruning

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

3. Attributes with many values

- Problem:
  - If attribute has many values, Gain will select it
  - Imagine using Date = June_6_1996
- So many values
  - Divides examples into tiny sets
  - Sets are likely uniform => high info gain
  - Poor predictor
- Penalize these attributes

One Solution: Gain Ratio

$$\text{Gain Ratio}(S,A) = \text{Gain}(S,A)/\text{SplitInfo}(S,A)$$

$$\text{SplitInfo} = \sum_{v \in \text{Values}(A)} (|S_v| / |S|) \, \text{Log2}(|S_v|/|S|)$$

SplitInfo ≅ entropy of S *wrt* values of A

(Contrast with entropy of S *wrt* **target** value)

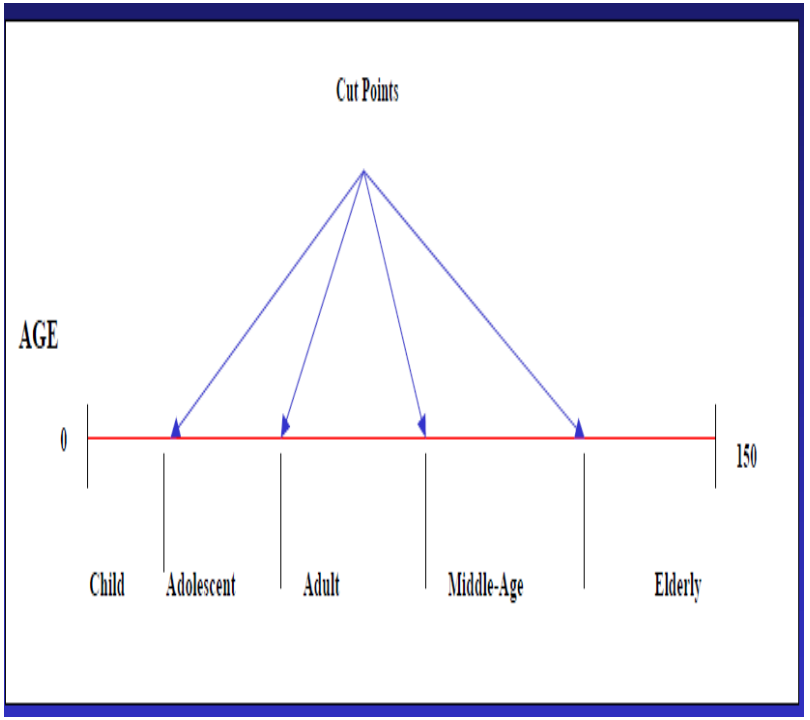
⇓ attribs with many uniformly distrib values

e.g. if A splits S uniformly into n sets

SplitInformation = $\log_2(n)$… = 1 for Boolean

4. Data Reduction

- Feature Discretization
  - Discretize values of continuous features into a small number of intervals
  - Each interval is mapped into a discrete symbol
  - Each feature handled independently of other features



- Benefits
  - Simplified data description
  - Easy to understand data
  - Easy to understand data mining results
  - Applicability of many ML algorithms
  - Quality of the analysis improved

Missing Data 1

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	?	weak	yes
d11	m	n	s	yes

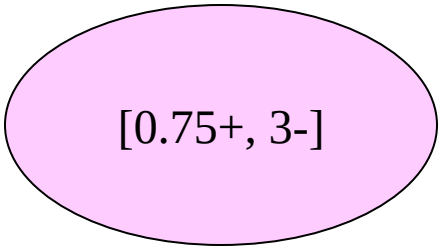
Assign most common value at this node
?=>h

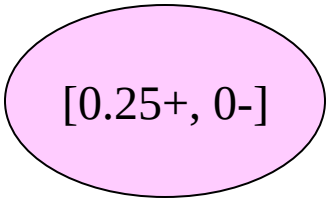
Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	?	weak	yes
d11	m	n	s	yes

Assign most common value for class
?=>n

Missing Data 2

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	?	weak	yes
d11	m	n	s	yes





- 75% h and 25% n
- Use in gain calculations
- Further subdivide if other missing attributes
- Same approach to classify test ex with missing attr
  - Classification is most probable classification
  - Summing over leaves where it got divided

Summary

- Decision tree learning provides a practical method for concept learning and for learning discrete-valued functions.
- ID3 searches a complete hypothesis space (i.e., the space of decision trees can represent any discrete-valued function defined over discrete-valued instances).
- The inductive bias implicit in ID3 includes a *preference* for smaller trees.
- Overfitting the training data is an important issue in decision tree learning.
- A large variety of extensions to the basic ID3 algorithm has been developed by different researchers.

Methodology Citations

- Dietterich, T. G., (1998). **Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms**. *Neural Computation, 10* (7) 1895-1924
- Densar, J., (2006). **Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets**. The Journal of Machine Learning Research, pages 1-30.