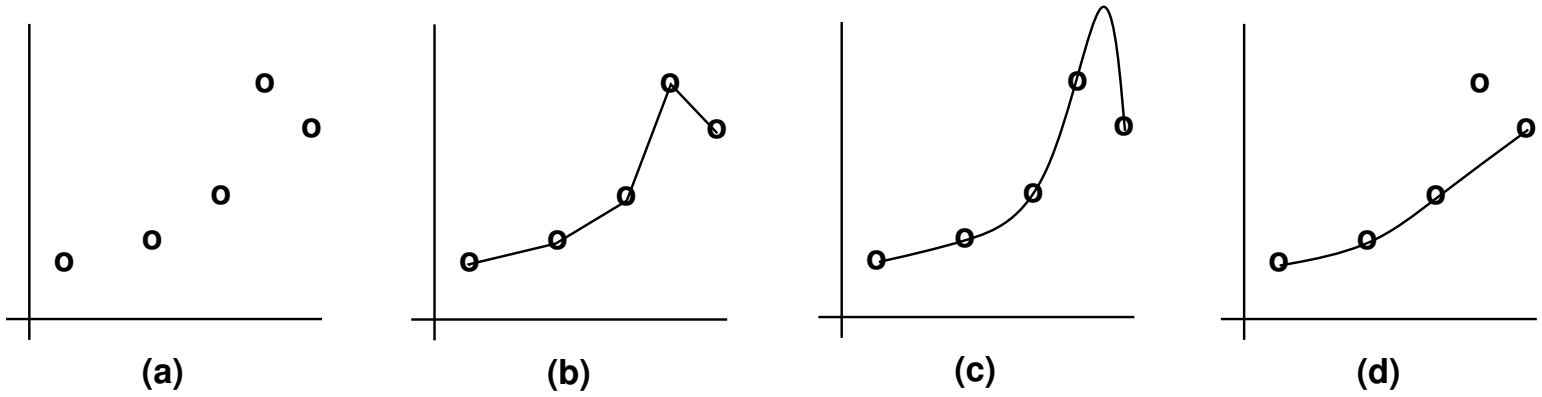
Decision Tree Learning

- Blue slides: Mitchell
- Olive slides: Alpaydin

Inductive Learning

- Given **example** pairs $(x, f(x))$, return a function h that approximates the function f:
  - **pure inductive inference,** or **induction**.
- The function h is called a **hypothesis**.

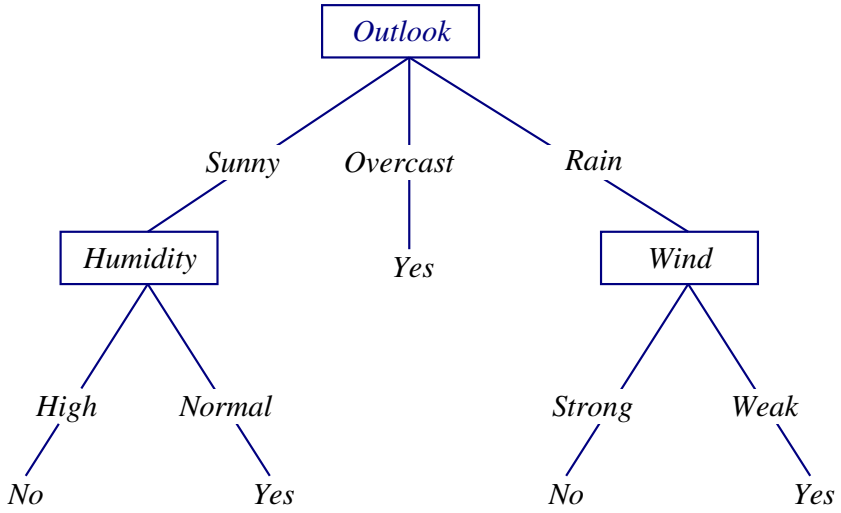
Inductive Learning and Bias



Given (a) as the training data, we can come up with several different hypotheses: (b) to (d)

- selection of one hypothesis over another is called a **bias**.
  - exact match to training data
  - prefer imprecise but smooth approximation
  - etc.

Decision Tree Learning



- Learn to approximate **discrete-valued** target functions.
- Step-by-step decision making: It can learn **disjunctive expressions**: Hypothesis space is **completely expressive**, avoiding problems with restricted hypothesis spaces.
- Inductive bias: **small trees** over large trees.

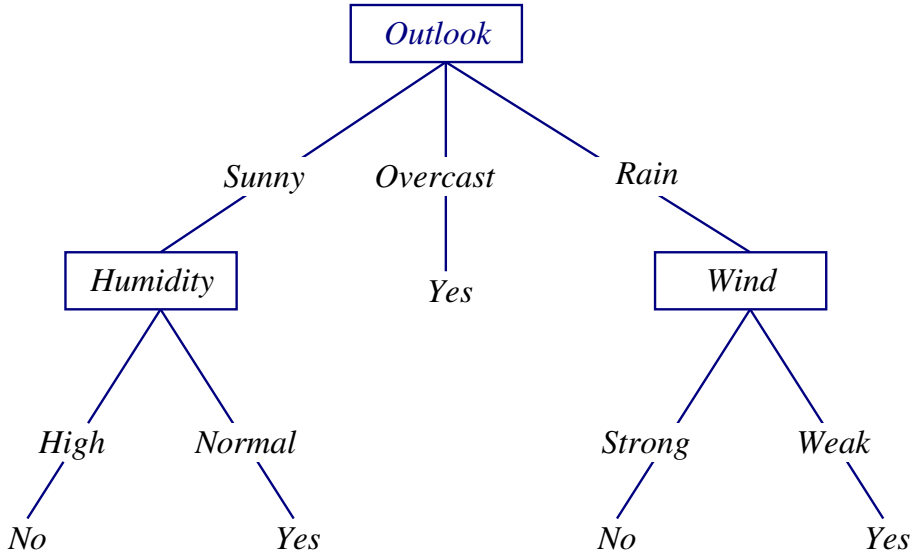
Example

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

Decision Trees

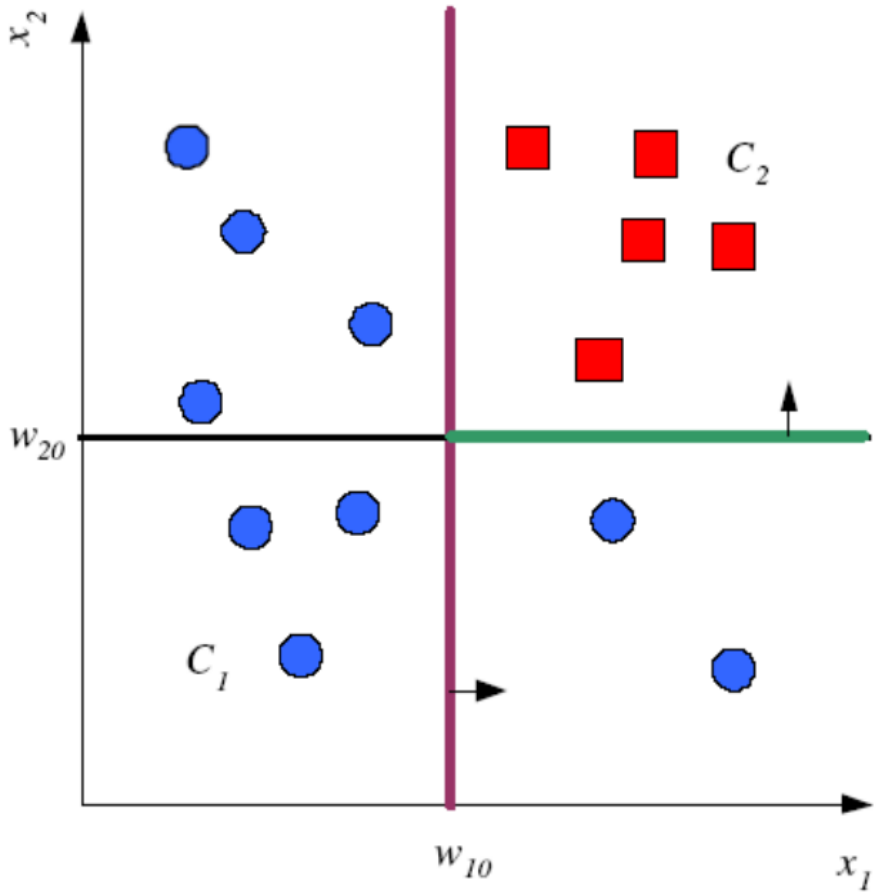
- A popular **inductive inference** algorithm.
- Algorithms: ID3, ASSISTANT, C4.5, etc.
- Applications: medical diagnosis, assess credit risk of loan applicants, etc.

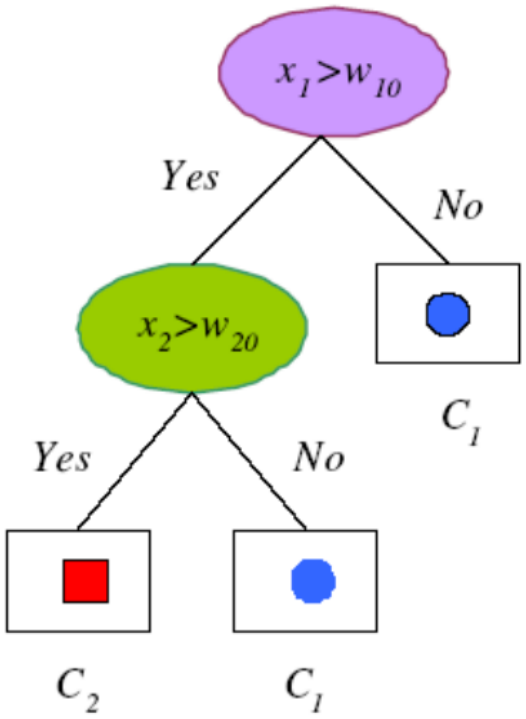
Decision Trees: Operation



- Each instance holds attribute values.
- Instances are classified by filtering the attribute values down the decision tree, down to a leaf which gives the final answer.
- Internal nodes: attribute names or attribute values. Branching occurs at attribute nodes.

[Alp] Trees Use Nodes and Leaves

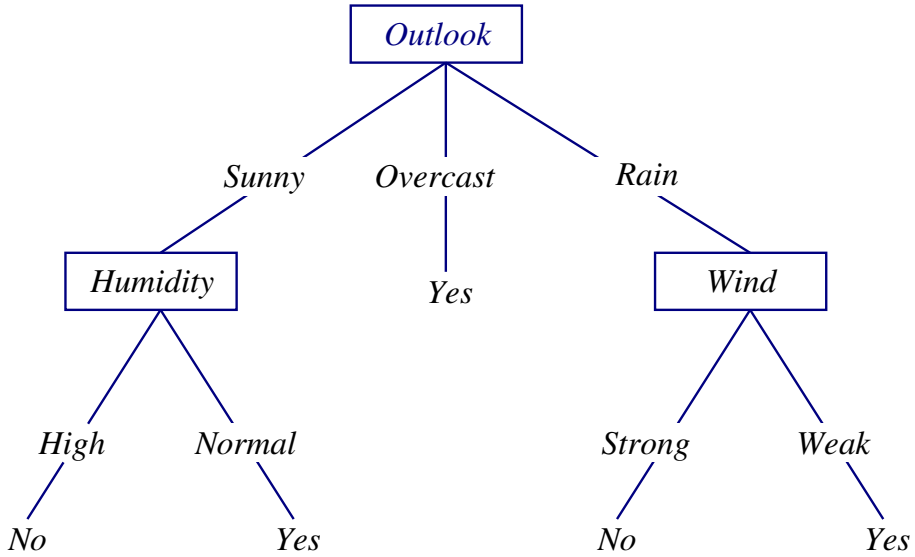




[Alp] Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\mathbf{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

Decision Trees: What They Represent



- Each path from root to leaf is a **conjunctions of constraints** on the attribute values.

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

Appropriate Tasks for Decision Trees

Good at **classification problems** where:

- Instances are represented by **attribute-value pairs**.
- The target function has **discrete output values**.
- Disjunctive descriptions may be required.
- The training data **may contain errors**.
- The training data **may contain missing attribute values**.

Constructing Decision Trees from Examples

- Given a set of examples (**training set**), both **positive** and **negative**, the task is to construct a decision tree that describes a concise decision path.
- Using the resulting decision tree, we want to **classify** new instances of examples (either as **yes** or **no**).

Constructing Decision Trees: Trivial Solution

- A trivial solution is to explicitly construct paths for each given example. In this case, you will get a tree where the number of leaves is the same as the number of training examples.
- The problem with this approach is that it is not able to deal with situations where, some attribute values are missing or new kinds of situations arise.
- Consider that some attributes may not count much toward the final classification.

Finding a Concise Decision Tree

- Memorizing all cases may not be the best way.
- We want to extract a decision pattern that can describe a large number of cases in a **concise** way.
- In terms of a decision tree, we want to make as few tests as possible before reaching a decision, i.e. the depth of the tree should be shallow.

Finding a Concise Decision Tree (cont'd)

- Basic idea: pick up attributes that can clearly separate positive and negative cases.
- These attributes are more important than others: the final classification heavily depend on the value of these attributes.

Decision Tree Learning Algorithm: ID3

```
function DECISION-TREE-LEARNING(examples, attributes, default) returns a decision tree
  inputs: examples, set of examples
          attributes, set of attributes
          default, default value for the goal predicate

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MAJORITY-VALUE(examples)
  else
      best ← CHOOSE-ATTRIBUTE(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← DECISION-TREE-LEARNING(examples_i, attributes − best,
                                           MAJORITY-VALUE(examples))
          add a branch to tree with label v_i and subtree subtree
      end
      return tree
```

Choosing the Best Attribute

$A1$ or $A2$?

- With initial and final number of positive and negative examples based on the attribute just tested, we want to decide which attribute is **better**.
- How to quantitatively measure which one is better?

Choosing the Best Attribute to Test First

Use Shannon's information theory to choose the attribute that give the maximum **information gain**.

- Pick an attribute such that the information gain (or entropy reduction) is maximized.
- Entropy measures the **average surprisal** of events. Less probable events are more surprising.

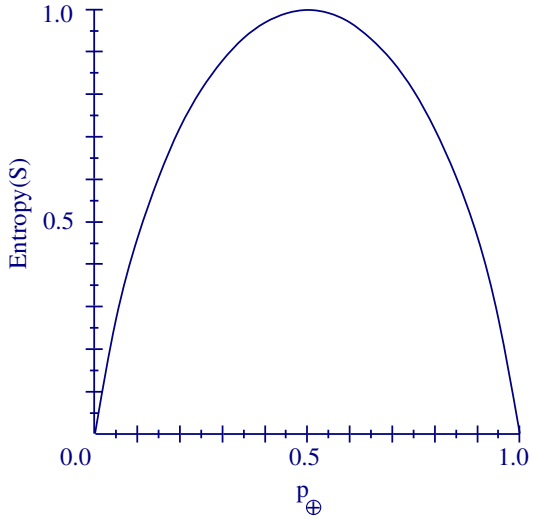
Information Theory (Informal Intro)

Given two events, H and T (Head and Tail):

- Rare (uncertain) events give more surprise:
  H more surprising than T if $P(H) < P(T)$
  H more uncertain than T if $P(H) < P(T)$
- How to represent "more surprising", or "more uncertain"?
  $Surprise(H) > Surprise(T)$ if

$$
\begin{aligned}
& P(H) < P(T) \\
\leftrightarrow \quad & \frac{1}{P(H)} > \frac{1}{P(T)} \\
\leftrightarrow \quad & \log\left(\frac{1}{P(H)}\right) > \log\left(\frac{1}{P(T)}\right) \\
\leftrightarrow \quad & -\log\left(P(H)\right) > -\log\left(P(T)\right)
\end{aligned}
$$

Information Theory (Cont'd)



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the average uncertainty in S

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Uncertainty and Information

- By performing some query, if you go from state S_1 with entropy $E(S_1)$ to state S_2 with entropy $E(S_2)$, where $E(S_1) > E(S_2)$, your **uncertainty has decreased**.
- The amount by which uncertainty decreased, i.e., $E(S_1) - E(S_2)$, can be thought of as information you gained (**information gain**) through getting answers to your query.

Computing Entropy of a Data Set D

Given a data set D, we can compute its entropy based on:

- Number of positive samples = $n_{\oplus}$
- Number of negative samples = $n_{\ominus}$
- From this, we can compute:

$$P_{\oplus} = \frac{n_{\oplus}}{n_{\oplus} + n_{\ominus}}$$

$$P_{\ominus} = \frac{n_{\ominus}}{n_{\oplus} + n_{\ominus}}$$

Then,

$$E(D) = -P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}$$

Entropy and Information Gain

$$Entropy(S) = \sum_{i \in C} -P_i log_2(P_i)$$

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

- C: categories (classifications)
- S: set of examples
- A: a single attribute
- S_v: set of examples where attribute $A = v$.
- $|X|$: cardinality of arbitrary set X.

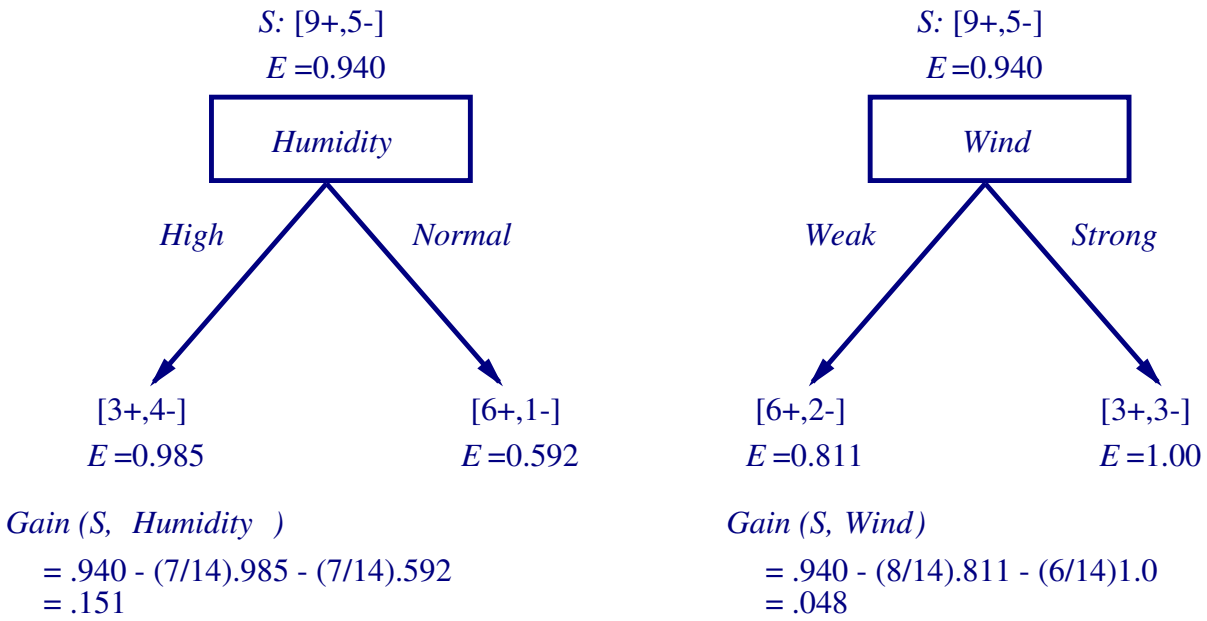
Example

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

- Which attribute to test first?

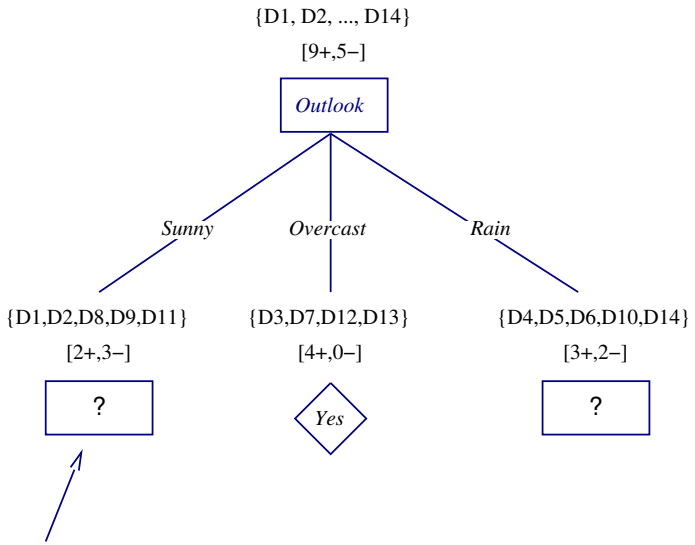
Choosing the Best Attribute

Which attribute is the best classifier?



- $+$: # of positive examples; $-$: # of negative examples
- Initial entropy $= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$.
- You can calculate the rest.
- Note: $0.0 \times \log_2 0.0 \equiv 0.0$ even though $\log_2 0.0$ is not defined.

Partially Learned Tree



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

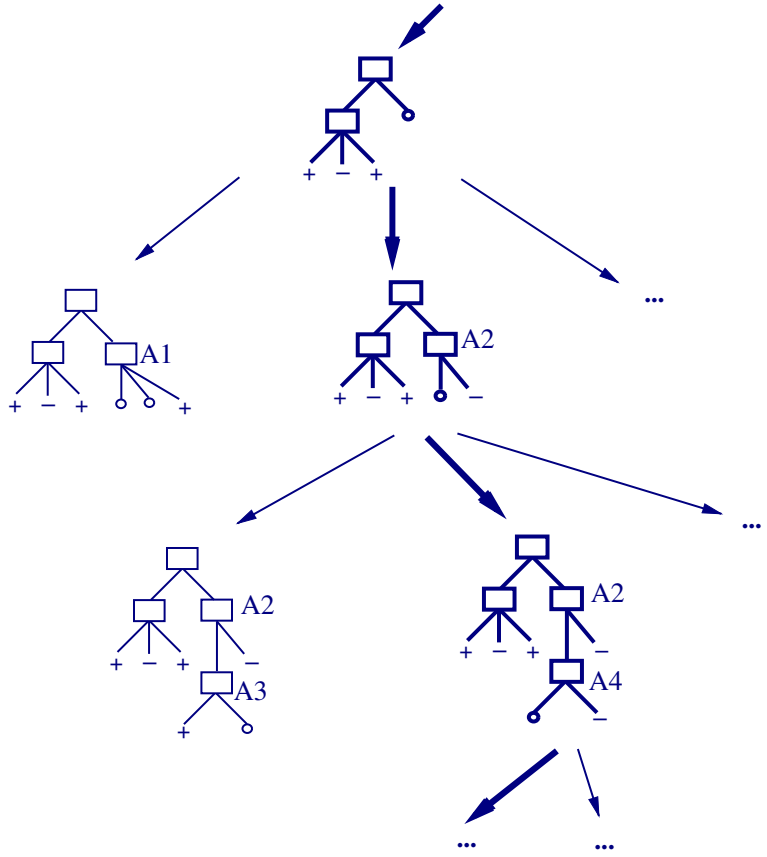
Gain (S_{sunny} , *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

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

- Select next attribute, based on the remaining examples.

Hypothesis Space Search in ID3



- At each branch, we make a decision regarding a particular attribute. Choice of an attribute directs the search toward a certain final hypothesis.

Hypothesis Space Search in ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

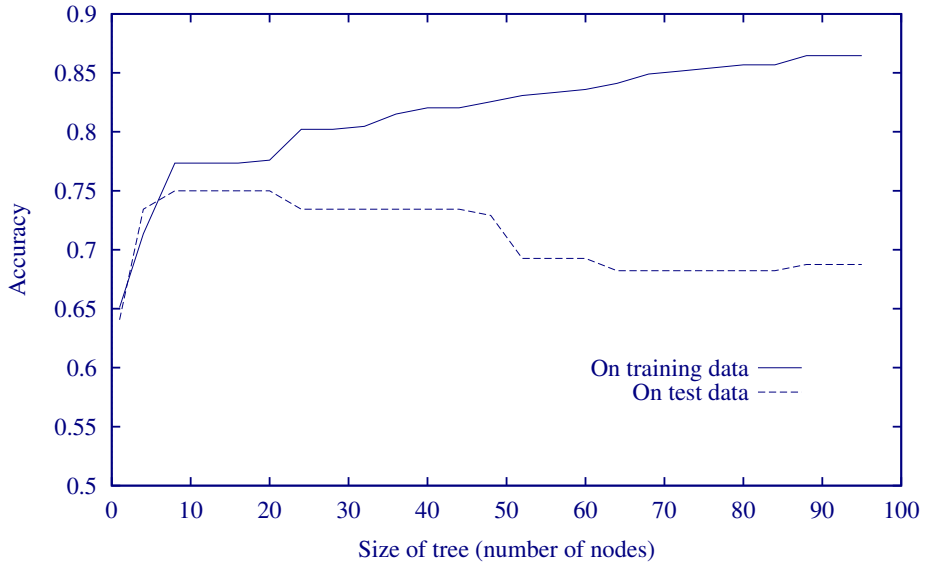
Inductive Bias in ID3

- ID3 is biased:
  - Not because of the restriction on the hypothesis space, but
  - Because of the **preference** for a particular hypothesis.
- Such an inductive bias is called **Occam's razor**: *The most likely hypothesis is the* **simplest one** *that is consistent with all observations.*

Accuracy of Decision Trees

- Divide examples into training and test sets.
- Train using the training set.
- Measure accuracy of resulting decision tree on the test set.

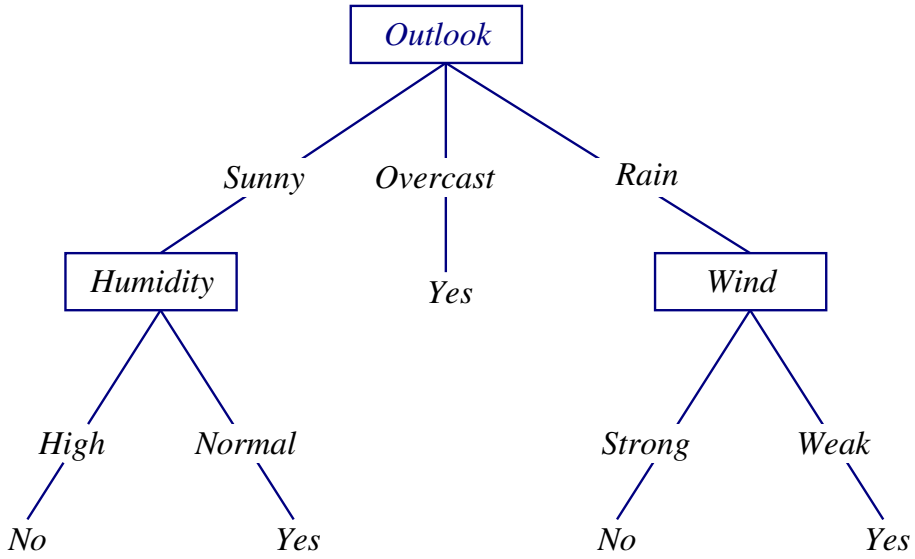
Issue: Overfitting



Overfitting:

- 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' is worse than h on the training set but h' is better than h over the entire distribution of instances.
- Can be due to **noise** in data.

Issue: Noise



- What if
  $\langle Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, Play = No \rangle$ was added as a training example?
- Further elaboration of the above tree becomes necessary.
- The resulting tree will fit the training data plus the noise, but it may perform poorly on the true instance distribution.

Overcoming Overfitting

- Stop early.
- Allow overfitting, then post-prune tree.
- Use separate set of examples not used in training to monitor performance on unobserved data (validation set).
- Use all available data, but perform statistical test to estimate chance of improving.
- Use explicit measure of complexity of encoding, and put a bound on tree size.

[Alp] Regression Trees

- Error at node m:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_m : \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

$$E_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m\left(\mathbf{x}^t\right) \qquad g_m = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$$

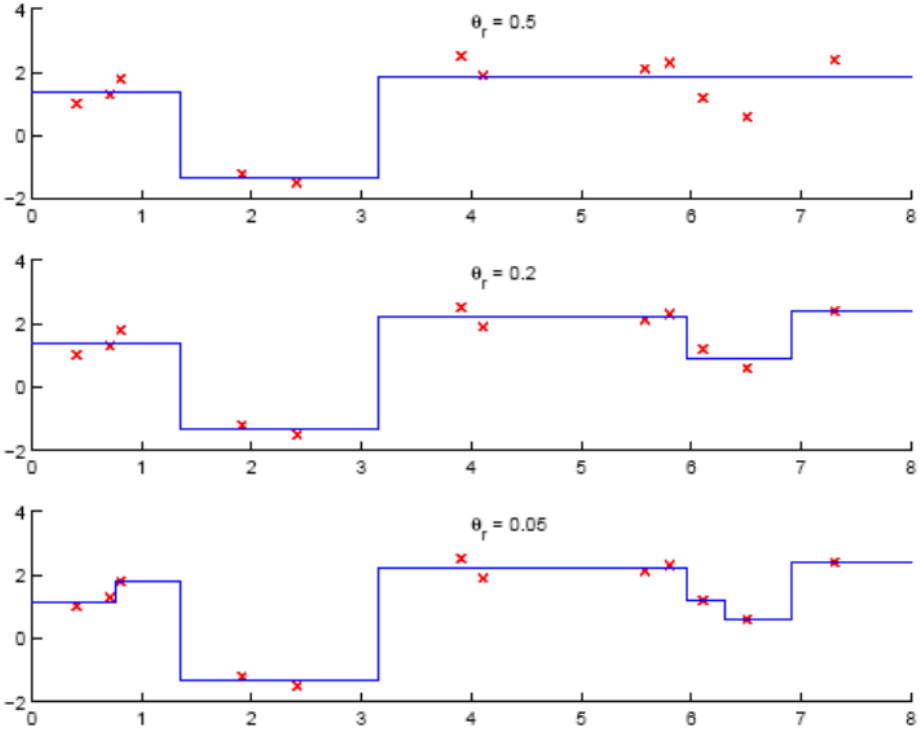
- After splitting:

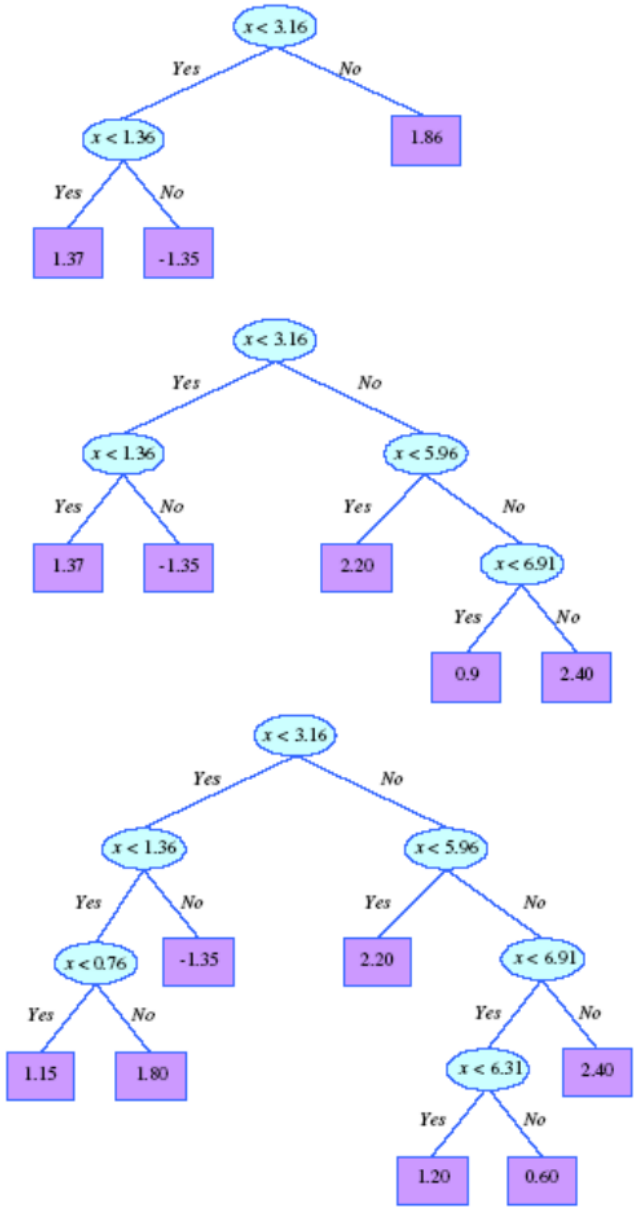
$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_{mj} : \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$E'_m = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}\left(\mathbf{x}^t\right) \qquad g_{mj} = \frac{\sum_t b_{mj}\left(\mathbf{x}^t\right) r^t}{\sum_t b_{mj}\left(\mathbf{x}^t\right)}$$

[Alp] Model Selection in Trees

Model Selection in Trees



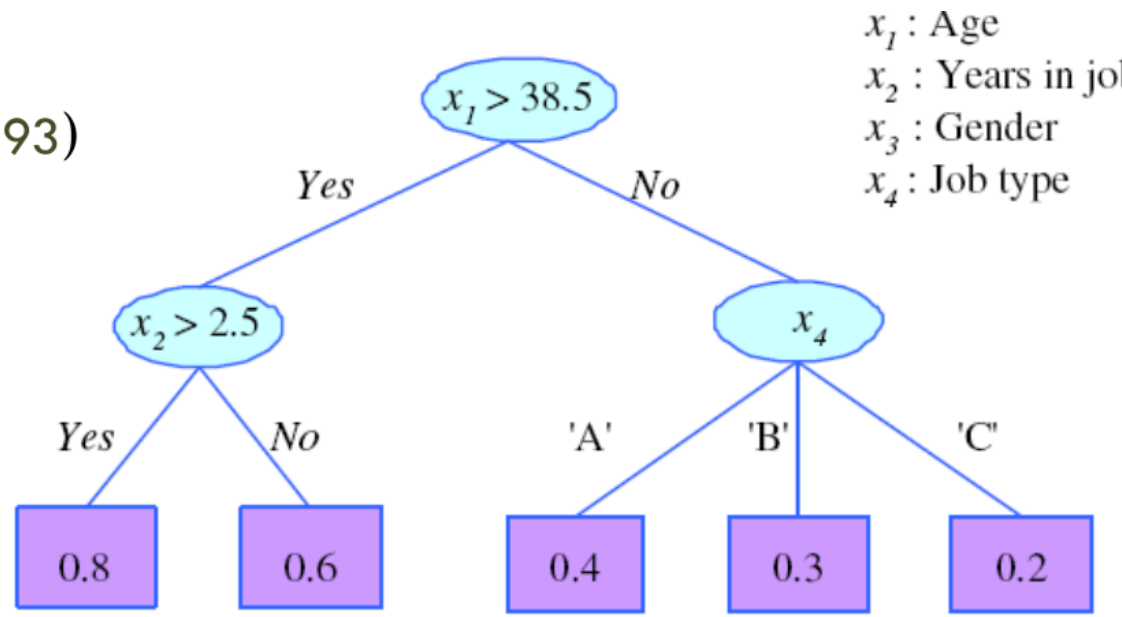


[Alp] Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

[Alp] Rule Extraction from Trees

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

[Alp] Rule Learning

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule covers an example if all terms of the rule evaluate to true for the example
- Sequential covering: Generate rules one at a time until all positive examples are covered
- IREP (Fürnkrantz and Widmer, 1994), Ripper (Cohen, 1995)

Other Issues

- Continuous-valued attributes: dynamically define new discrete-valued attributes
- Multi-valued attributes with large number of possible values: Use measures other than information gain.
- Training examples with missing attribute values: Assign most common value, or assign with the occurring frequency.
- Attributes with different cost/weighting: Scale using the cost.