Decision Trees

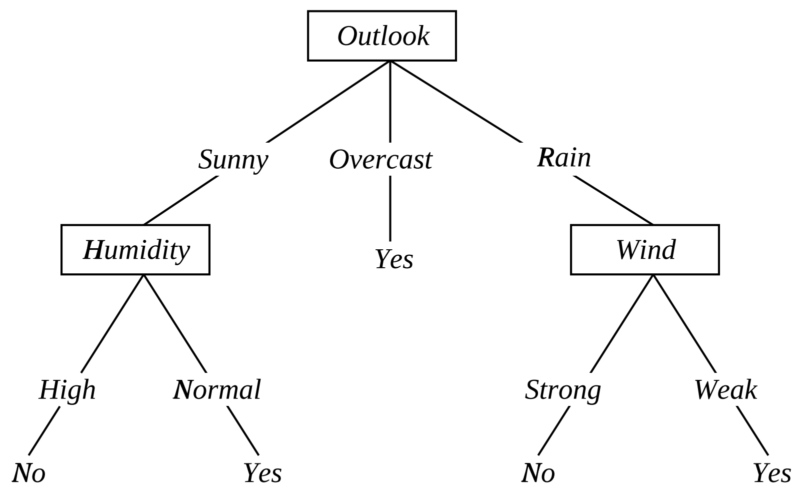
- *Decision tree learning* is a form a supervised learning for concept classification. A decision tree represents a procedure for classifying objects based on their attributes.
- Decision trees are perhaps the most popular ML/KDD algorithm.
- Internal node: an attribute to test. Leaf node: a classification.
- To classify an object: start at the root node, traverse the branches corresponding to the attribute values of the object.

A Concept Learning Task

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	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 Tree Representation

- Internal nodes test the value of a particular feature, and branch according to the outcome of the test
- Leaf nodes specify the class h(x).



Example: $< Outlook = sunny, Temperature = Hot, Humidity = High, Wind = Strong >$ Class:

Decision Trees Represent Disjunction of Conjunctions

- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests.
- The tree itself corresponds to a disjunction of these conjunctions.
- Example – decision tree of last slide corresponds to the expression:

  $(Outlook = Sunny \wedge Humidity = Normal) \vee$

  $(Outlook = Overcast) \vee$

  $(Outlook = Rain \wedge Wind = Weak)$

- A decision tree can be converted to an equivalent rule set.

Appropriate Problems for Decision Tree Learning

- Instances represented by attribute-value pairs
- Target function has a discrete number of output values
- Disjunctive descriptions may be required
- Training data may contain errors
- Training data may contain missing attribute values

Designing a Decision Tree Learning Algorithm

Goal: Construct a decision tree that agrees (is consistent) with the training set.

Trivial solution: construct a decision tree that has one path to a leaf for every example.

Problem with trivial solution?

Non-trivial solution: find a concise decision tree that agrees with the training data.

Problem?

Solution?

ID3

- ID3 is a well-known decision tree algorithm that uses a top-down greedy search through the hypothesis space.
- ID3 was designed to handle large training sets with many attributes.
- ID3 tries to generate fairly simple trees, but is not guaranteed to produce the best one.
- Two of the most widely used decision tree algorithms, C4.5 and C5.0, are descendents of ID3.

Basic Decision Tree Algorithm

Approach: construct tree from the top down, starting with the question: which attribute should be tested at the root of the tree.

Algorithm:

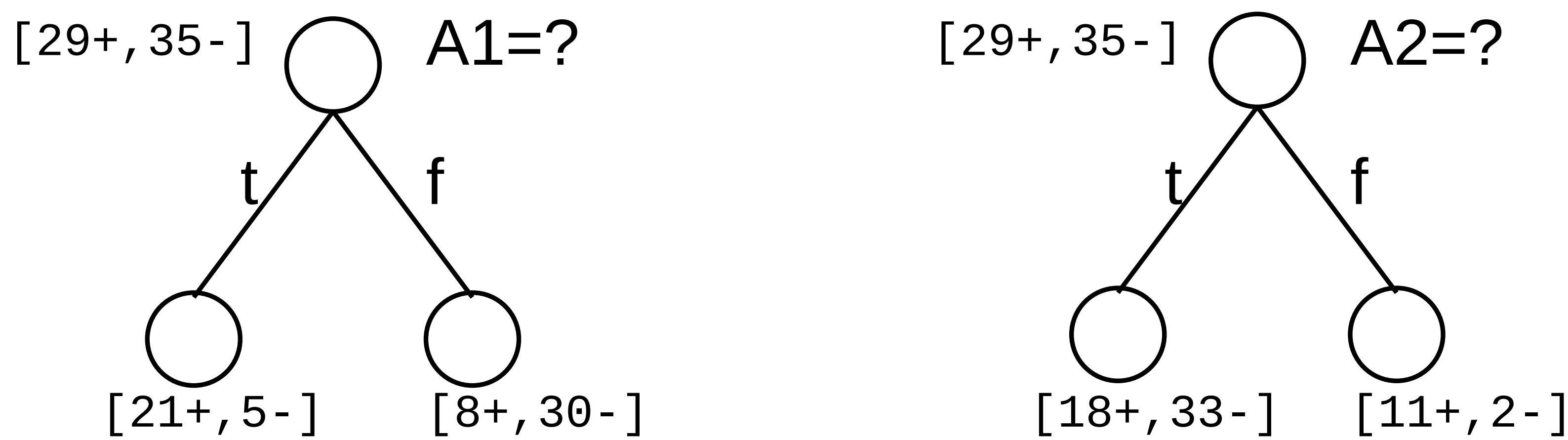
If all instances from same class
    then tree is leaf with that class name
    else
        pick test for decision node
        partition instances by test outcome
        construct one branch for each possible outcome
        build subtrees recursively

Example

CS Major Database

Height	Eyes	Class
short	brown	hacker
tall	blue	theorist
tall	brown	hacker
short	blue	theorist

Which attribute is best?



Characteristics of Tests

Imagine that we have examples from two classes P and N. How do we decide which attribute to split on?

Lets first take a look at some characteristics of tests:

Let S contain 20 occurrences of P and 20 of N.

Imagine a Boolean test that splits the data into two subsets S_1 and S_2,

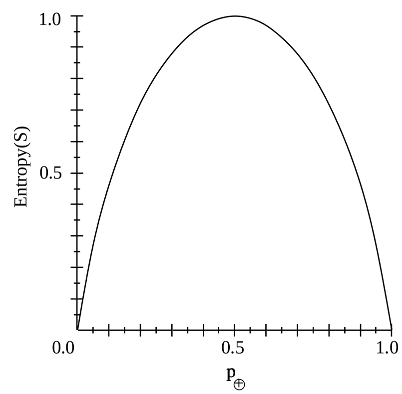
best case: $S_1 = 20P\text{'s} \textit{ and } S_2 = 20N\text{'s}$

worst case: $S_1 = 10P, 10N \textit{ and } S_2 = 10P, 10N$

intermediate case: $S_1 = 17P, 1N \textit{ and } S_2 = 3P, 19N$

Q: Why is the third case better than the second?

Entropy



- S is a sample of training examples
- p is the proportion of positive examples in S
- n is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p \log_2 p - n \log_2 n$$

Some Examples

- All points of class P
  - $\frac{p}{p+n} = 1.0$
  - $\frac{n}{p+n} = 0.0$
  - $-1log_2(1.0) - 0log_2(0) = 0.0$
  - No disorder
- Half and Half
  - $\frac{p}{p+n} = 0.5$
  - $\frac{n}{p+n} = 0.5$
  - $-0.5log_2(0.5) - 0.5log_2(0.5) = 1.0$
  - Maximum disorder

Multiclass

Thus far we have assumed that the target class is Boolean. More generally, the class can take on c values, then the entropy of S relative to this c-wise classification is defined as:

$$Entropy(S) = \sum_{i=1}^{c} -\frac{\mid S_i \mid}{\mid S \mid} \log_2\left(\frac{\mid S_i \mid}{\mid S \mid}\right)$$

Where S_i is the proportion of S belonging to class i.

Information Gain

Measures the expected reduction in entropy caused by partitioning the examples according to the attribute.

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

Calculation for Attribute Humidity

branch	value	n_{bp}	n_{bn}	disorder
1	high	3	4	.99
2	normal	6	1	.58

Entropy (high) $= -\frac{3}{7}log_2(\frac{3}{7}) - \frac{4}{7}log_2(\frac{4}{7}) = .99$

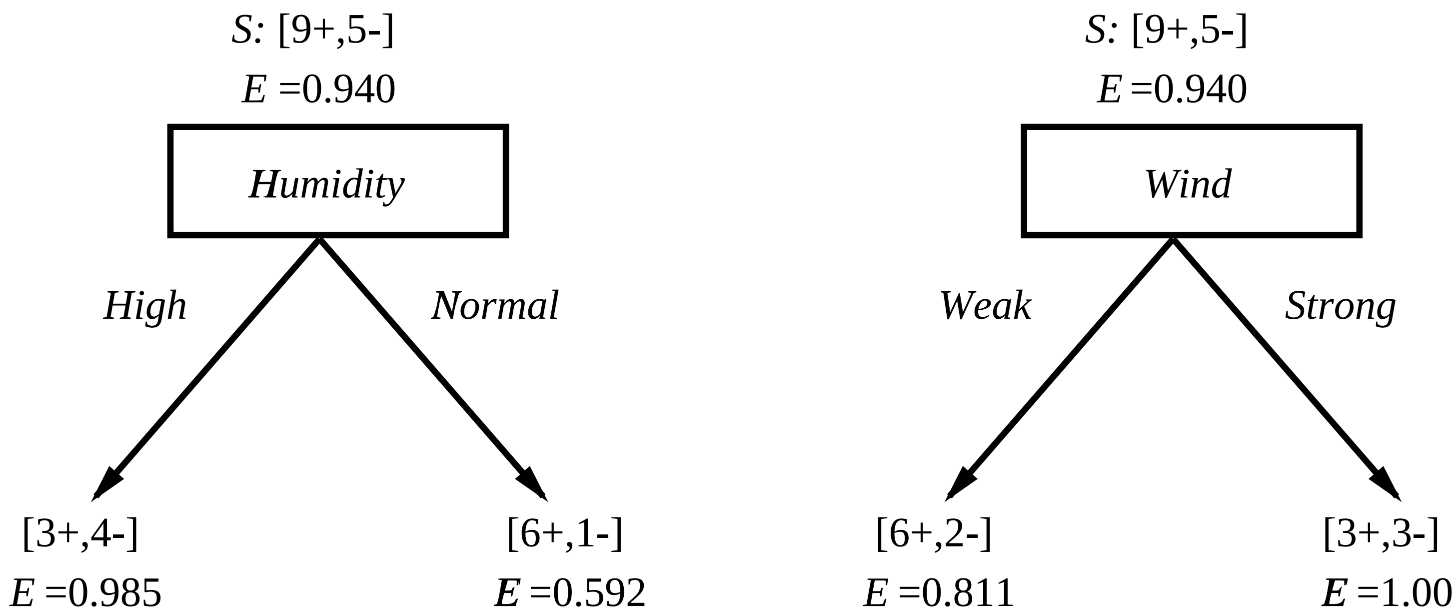
Entropy (normal)$= -\frac{6}{7}log_2(\frac{6}{7}) - \frac{1}{7}log_2(\frac{1}{7}) = .58$

Gain(S, Humidity) =

Entropy(S) - $\frac{7}{14}$ Entropy(high) - $\frac{7}{14}$ Entropy(normal) =

$0.940 - \frac{7}{14}(.99) - \frac{7}{14}(.58) = 0.151$

Which attribute is the best classifier?



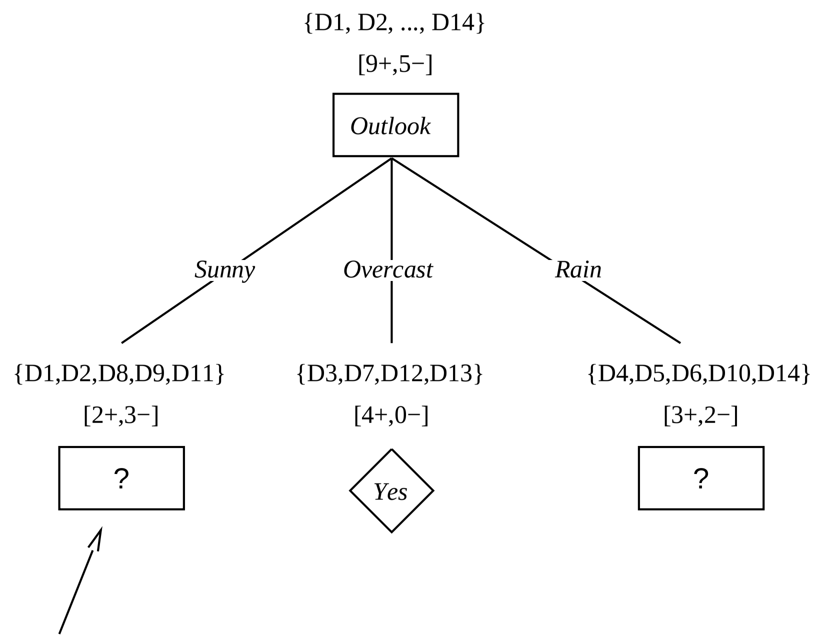
Gain (S, Humidity)

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

Gain (S, Wind)

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

Example of Tree Construction



Which attribute should be tested here?

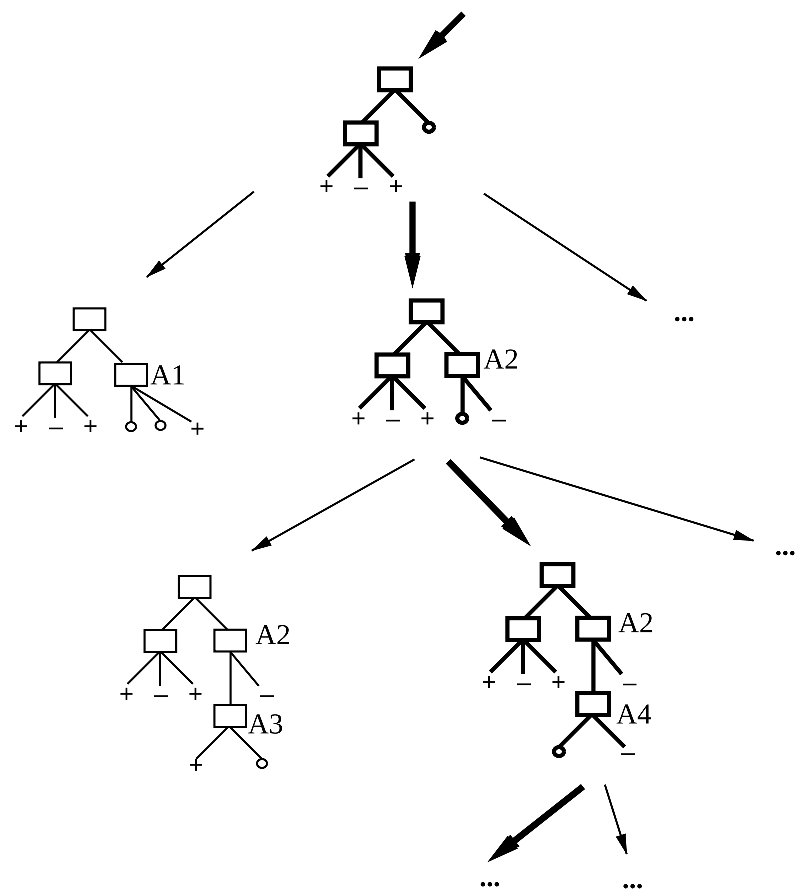
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain(S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain(S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

ID3's Hypothesis Space



ID3's Hypothesis Space

- ID3's hypothesis space is the set of all possible decision trees. This is a <u>complete</u> space of finite discrete-valued functions (relative to the given attributes).

  BOTTOM LINE: The hypothesis space is guaranteed to contain the target concept.

- Outputs a *single* hypothesis.
- Performs a simple-to-complex hill-climbing search: information gain metric guides the search.
- No backtracking: Can get stuck in local maxima!
- Statistically-based search heuristic: robust to noise.

ID3's Inductive Bias

- prefers shallow trees over deeper ones.

  ID3 can be viewed a heuristic approximation of breadth-first search for decision tree generation!

- prefers to branch on attributes with highest information gain first.

Preference vs. Restriction Biases

Preference Bias: preference for certain hypotheses over others, but no restriction on the hypothesis space.

ID3 searches a complete hypothesis space, but searches it incompletely.

Restriction Bias: a priori restriction on the hypothesis space, but no preference for one hypothesis over another.

The candidate elimination algorithm searches an incomplete hypothesis space, but searches it completely.

Some learning algorithms use both preference and restriction biases, such as evaluation function learning methods.

Occam's Razor Bias

Occam's Razor heuristic: Prefer the simplest hypothesis that fits the data.

Why prefer simpler hypotheses over complex hypotheses?

Common rationale: There are fewer simpler hypotheses, so it is less likely to be coincidence if a simple hypothesis fits the data.

Counter arguments:

- But there are lots of meaningless small hypothesis sets!
- The complexity of a hypothesis depends on the representation used by the learner.