Decision trees

- decision tree learning is an inductive learning technique
- normally used for approximating discrete-valued target functions
- very mature technique, dates all the way back to the 1960s
- used for concept learning in assessing the credit risk of loan applications, diagnosing medical conditions, predicting river quality from biological properties, evaluating uranium fuel pellets and many more

Decision tree learning example: playing tennis (Mitchell)

- we want to decide whether to play tennis based on the condition of the weather
- the data we can use to make our decision are *outlook* (sunny, overcast, rain), *temperature* (cool, mild, hot), *humidity* (normal, high) and *wind conditions* (weak, strong)
- we have 14 examples of the form:

  (outlook=sunny, temperature=cool, humidity=normal, wind=weak, Yes)

  (outlook=rain, temperature=mild, humidity=high, wind=strong, No)

Decision tree algorithms

- decision trees are grown based on a core algorithm that employs top-down, greedy search through the space of possible trees
- classic algorithms: ID3 (Quinlan 1986) and C4.5 (Quinlan 1993)
- an implementation of C4.5 (release 8) is available from Quinlan's website `http://www.rulequest.com/Personal/`

General decision tree algorithm

- start with set of feature-value examples with classifications
- if all examples have the same classification, add the classification to the tree and terminate, otherwise
  1. choose an attribute to split the data on
  2. add a corresponding test to the tree
  3. create new branches for each value of the chosen attribute
  4. assign each example to the appropriate branch
  5. if some branch is assigned no examples, label the branch with the default value for the goal (default classification)
- carry on until all attributes have been used or all examples have been classified

How to choose which attribute to split on

- we want to choose an attribute to split on
- ID3 uses *information gain* to choose
- information gain is based on the idea on reducing the *entropy* of the examples
- the entropy of the examples is a measure of the amount of randomness/impurity

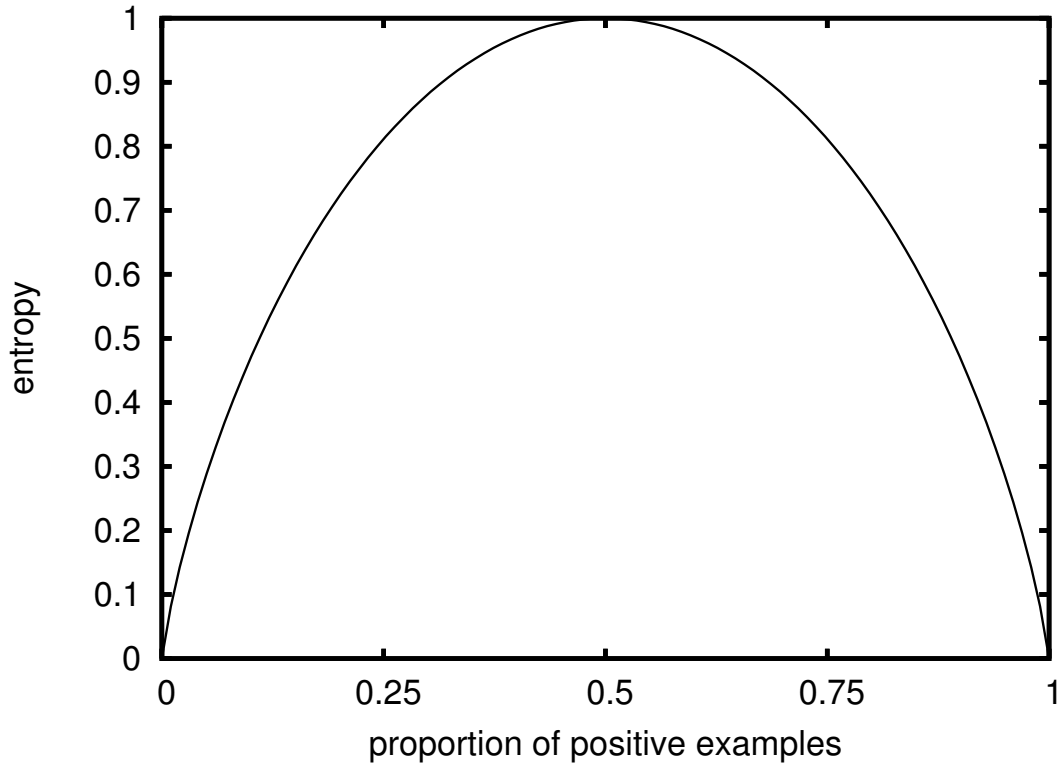
Entropy (1/3)

Given a collection of examples S with positive and negative classifications of some (boolean) target concept, the entropy of S relative to the classification is:

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

where $p_{\oplus}$ is the proportion of examples that are classified as positive and $p_{\ominus}$ is the proportion of them that are classified as negative

Entropy (2/3)


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
entropy
0
0.25
0.5
0.75
1
proportion of positive examples

Entropy (3/3)

- if the set of examples is split exactly in two equal size classes, then the entropy is 1 as it is unpredictable if a random example is positive or negative
- if the set of examples are all positive (negative) then the entropy is zero as a random example will always be positive (negative)
- NB: in entropy calculations we define $0 log_2 0$ to be 0

Information gain (1/2)

Information gain is the expected amount of reduction in entropy that is caused by partitioning the data according to an attribute.

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

where S is the original data, A is the attribute we split on, $\text{Values}(A)$ are the possible values for attribute A and S_v are the resulting sets of data

Information gain (2/2)

- NB: $S_v = \{s \in S | A(s) = v\}$
- the first term of the equation is the entropy of the original examples
- the second term is the sum of the entropies of the sets that result from splitting using A as the attribute
- therefore $\text{Gain}(S)$ is the expected reduction in entropy

The tennis example (1/7)

Examples D1 – D7 (Mitchell) :

Day	Sunny	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

The tennis example (2/7)

Examples D8 – D14 (Mitchell) :

Day	Sunny	Temperature	Humidity	Wind	PlayTennis
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

The tennis example (3/7)

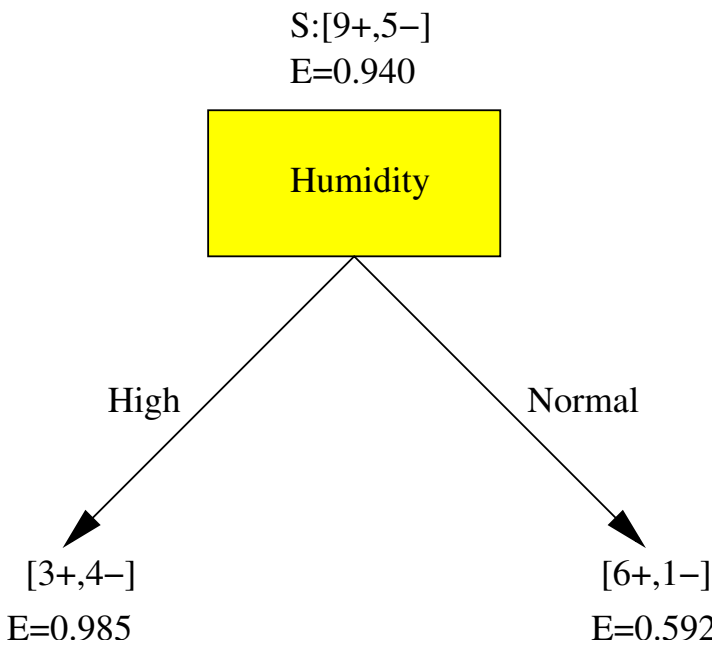
Calculate the information gain of all attributes at the root of the tree:

$$\begin{aligned}
\text{Gain}(S, \text{Outlook}) &= \mathbf{0.246} \\
\text{Gain}(S, \text{Humidity}) &= 0.151 \\
\text{Gain}(S, \text{Wind}) &= 0.048 \\
\text{Gain}(S, \text{Temperature}) &= 0.029
\end{aligned}$$

The algorithm is greedy so the highest of these is selected: Outlook

The tennis example (4/7)

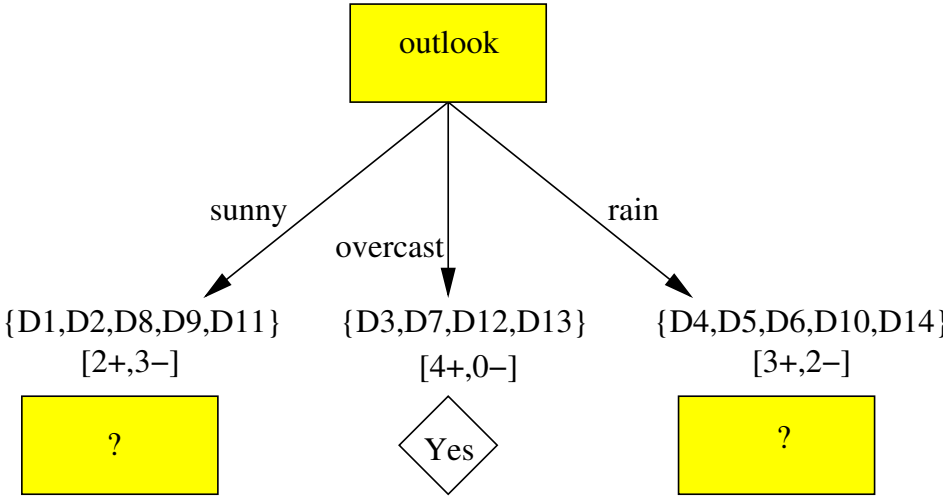
How the information gain computation happens. Example using humidity with the whole set of 14 examples:



$$\text{Gain}(S, \text{Humidity}) = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$$

The tennis example (5/7)

We've splt the data based on Outlook:

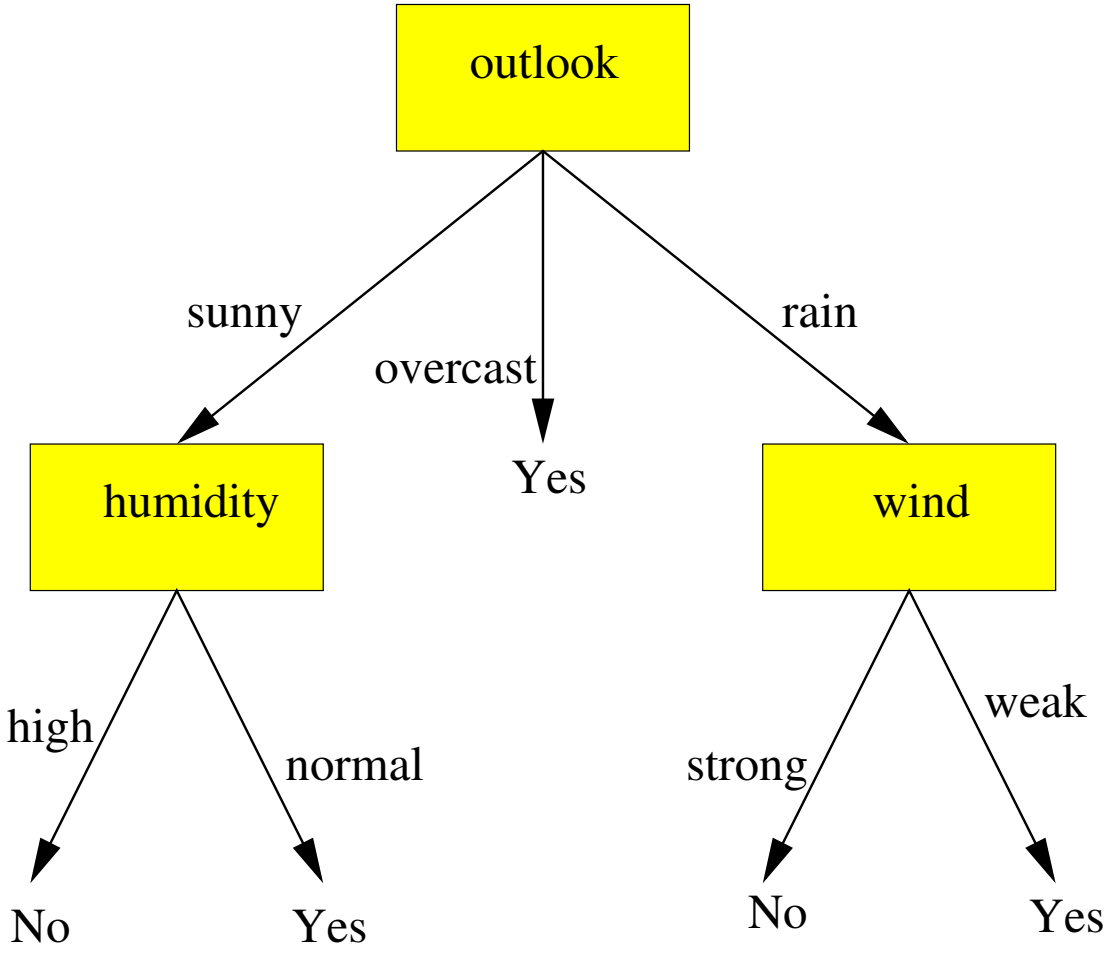


The tennis example (6/7)

Which attribute should we split on next?

- we calculate the information gain for all attributes based on the new sets of examples
- the highest of those is selected
- new splits are made until no examples are left unclassified or all attributes have been used

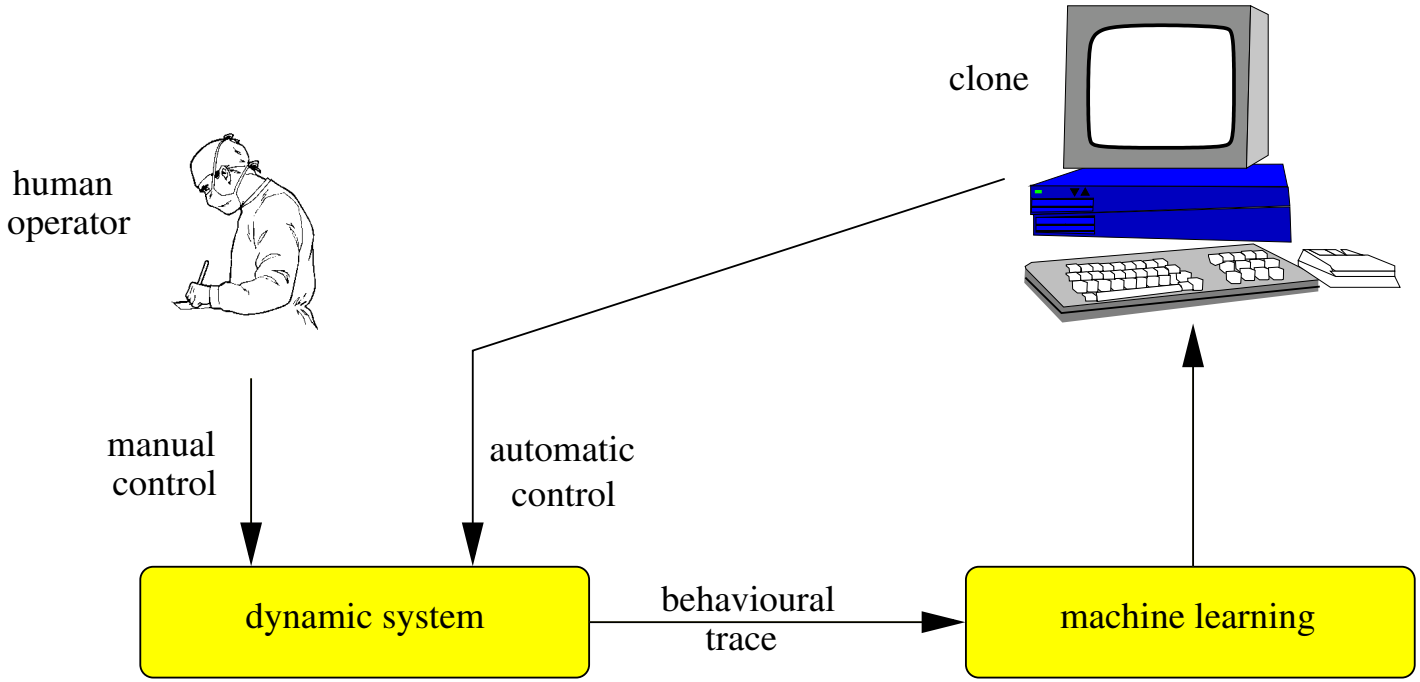
The tennis example (7/7)



Many extensions for decision trees

- the algorithm we just traced with the tennis data is Quinlan's ID3
- Quinlan's extensin to ID3, C4.5, is essentially the same as ID3 but it also goes through a rule post-pruning phase
- decision trees have been extended to allow continuous valued attributes, missing attribute values for some examples and costs

Behavioural Cloning



Behavioural cloning

1. *behavioural cloning* is a method by which human *subcognitive* skills can be captured and reproduced in a computer program.
2. As the human subject performs the skill, his or her actions are recorded along with the situation that gave rise to the action.
3. a log of these records is used as input to a learning program
4. the learning program outputs a set of rules that reproduce the skilled behaviour.

Behavioural cloning: intuition

- control skills are often acquired through experience and are sub-cognitive
- they are hard to reconstruct and represent in a computer program
- some means of automatically acquiring skills is needed

Behavioural trace

- the trace is a set of examples of correct control by a human expert
- each example is essentially a pair (State,Action)
- problem: time delay between state and action to account for time needed by operator for state recognition and physical action
- example 1: moving the rudder of a jumbo set is very unresponsive
- example 2: pilots anticipate situations and plan their actions accordingly

Learning to fly: setup

- flight simulation program logs the actions taken by three human subjects as they fly an aircraft
- the program also logs the state of the simulation as the experiment unfolds
- states are taken from earlier time points to account for delay (delay set by experimentation)
- states are represented as feature-value vectors (natural representation for decision trees!)

State representation (1/2)

on_ground	boolean
g_limit	boolean
wing_stall	boolean
twist	integer
elevation	integer
azimuth	integer
roll_speed	integer
elevation_speed	integer
azimuth_speed	integer
airspeed	integer

State representation (2/2)

climbspeed	integer
E/W distance	real
altitude	real
N/S distance	real
fuel	integer
rollers	real
elevator	real
rudder	real
thrust	integer
flaps	integer

Training data

- a decision tree is constructed with C4.5 for each control variable
- control variables: elevator, rollers, thrust and flaps
- the effect of the other three control variables is seen as part of the state description for the decision tree for one variable
- numerical control variables: discretise to serve as class indicators

Hacking...

- lots of "hacks" are used to get the data in a usable state
- circular buffers are used in many places to simulate delays between a state change and the corresponding action
- also used to simulate the analogue nature of some instruments such as the rudder and rollers

Results

- performance of cloned auto-pilot was similar to human trainer but more “dependable” (clean-up effect)
- that meant that the clone flew the plane more “mechanically” than the human experts
- induced decision trees are quite complex
- in some cases, they are readable by a human and understandable
- however, many complex tasks are represented in very complicated decision trees

Behavioural cloning: caveats

- data can be sparse: clone learns from limited experience and overspecialises
- clone may fails in extreme situations
- state representation is not a good indication of what information the human uses to solve the problem
- clone's actions are purely reactive, not goal-oriented
- behavioural cloning for games?