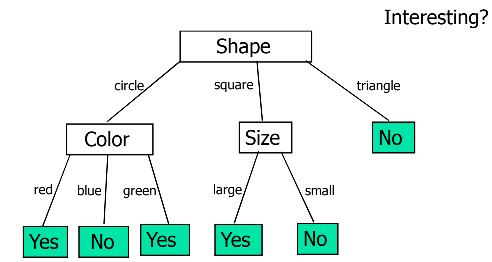
Decision Tree Learning

Ronald J. Williams
CSU520, Spring 2008

Decision Tree Example



Interesting=Yes ⇔ ((Shape=circle)^((Color=red)V(Color=green)))
V ((Shape=square)^(Size=large))

Inducing Decision Trees from Data

- Suppose we have a set of training data and want to construct a decision tree consistent with that data
- One trivial way: Construct a tree that essentially just reproduces the training data, with one path to a leaf for each example
  - no hope of generalizing
- Better way: ID3 algorithm
  - tries to construct more compact trees
  - uses information-theoretic ideas to create tree recursively

Inducing a decision tree: example

- Suppose our tree is to determine whether it's a good day to play tennis based on attributes representing weather conditions
- Input attributes

Attribute	Possible Values
Outlook	Sunny, Overcast, Rain
Temperature	Hot, Mild, Cool
Humidity	High, Normal
Wind	Strong, Weak

- Target attribute is PlayTennis, with values Yes or No

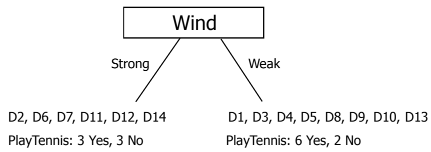
Training Data

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

Essential Idea

- Main question: Which attribute test should be placed at the root?
- In this example, 4 possibilities
- Once we have an answer to this question, apply the same idea recursively to the resulting subtrees
- Base case: all data in a subtree give rise to the same value for the target attribute
  - In this case, make that subtree a leaf with the appropriate label

- For example, suppose we decided that Wind should be used as the root
- Resulting split of the data looks like this:



- Is this a good test to split on? Or would one of the other three attributes be better?

Digression: Information & Entropy

- Suppose we want to encode and transmit a long sequence of symbols from the set {a, c, e, g} drawn randomly according to the following probability distribution D:

Symbol	a	c	e	g
Probability	1/8	1/8	1/4	1/2

- Since there are 4 symbols, one possibility is to use 2 bits per symbol
- In fact, it's possible to use 1.75 bits per symbol, on average
- Can you see how?

- Here's one way:

Symbol	Encoding
a	000
c	001
e	01
g	1

- Average number of bits per symbol $= \frac{1}{8} * 3 + \frac{1}{8} * 3 + \frac{1}{4} * 2 + \frac{1}{2} * 1$ $= 1.75$
- Information theory: Optimal length code assigns $\log_2 1/p = -\log_2 p$ bits to a message having probability p

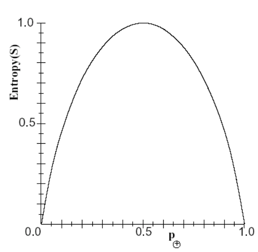
Entropy

- Given a distribution D over a finite set, where $<p_1, p_2, \ldots, p_n>$ are the corresponding probabilities, define the entropy of D by

$$H(D) = -\sum_i p_i \log_2 p_i$$

- For example, the entropy of the distribution we just examined, $<\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}>$, is 1.75 (bits)
- Also called information
- In general, entropy is higher the closer the distribution is to being uniform

- Suppose there are just 2 values, so the distribution has the form <p, 1-p>
- Here's what the entropy looks like as a function of p:



Back to decision trees - almost

- Think of the input attribute vector as revealing some information about the value of the target attribute
- The input attributes are tested sequentially, so we'd like each test to reveal the maximal amount of information possible about the target attribute value

This encourages shallower trees, we hope

- To formalize this, we need the notion of conditional entropy

- Return to our symbol encoding example:

Symbol	a	c	e	g
Probability	1/8	1/8	1/4	1/2

- Suppose we're given the identity of the next symbol received in 2 stages:
  - we're first told that the symbol is a vowel or consonant
  - then we learn its actual identity
- We'll analyze this 2 different ways

- First consider the second stage – conveying the identity of the symbol given prior knowledge that it's a vowel or consonant
- For this we use the conditional distribution of D given that the symbol is a vowel

Symbol	a	e
Probability	1/3	2/3

and the conditional distribution of D given that the symbol is a consonant

Symbol	c	g
Probability	1/5	4/5

- We can compute the entropy of each of these conditional distributions:

$H(D|Vowel) = - 1/3 \log_2 1/3 – 2/3 \log_2 2/3$

$= 0.918$

$H(D|Consonant)$

$= - 1/5 \log_2 1/5 – 4/5 \log_2 4/5$

$= 0.722$

- We then compute the expected value of this as 3/8 * 0.918 + 5/8 * 0.722 = 0.796

- H(D|Vowel) = 0.918 represents the expected number of bits to convey the actual identity of the symbol given that it's a vowel
- H(D|Consonant) = 0.722 represents the expected number of bits to convey the actual identity of the symbol given that it's a consonant
- Then the weighted average 0.796 is the expected number of bits to convey the actual identity of the symbol given whichever is true about it – that it's a vowel or that it's a consonant

Information Gain

- Thus while it requires an average of 1.75 bits to convey the identity of each symbol, once it's known whether it's a vowel or a consonant, it only requires 0.796 bits, on average, to convey its actual identity
- The difference 1.75 – 0.796 = 0.954 is the number of bits of information that are gained, on average, by knowing whether the symbol is a vowel or a consonant
  - called *information gain*

- The way we computed this corresponds to the way we'll apply this to identify good split nodes in decision trees
- But it's instructive to see another way: Consider the first stage – specifying whether vowel or consonant
- The probabilities look like this:

	Vowel	Consonant
Probability	3/8	5/8

- The entropy of this is

  $- 3/8 * \log_2 3/8 – 5/8 * \log_2 5/8 = 0.954$

Now back to decision trees for real

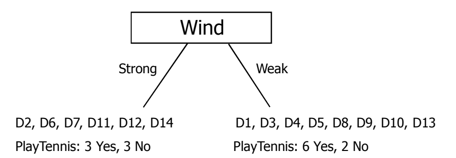
- We'll illustrate using our PlayTennis data
- The key idea will be to select as the test for the root of each subtree the one that gives maximum information gain for predicting the target attribute value
- Since we don't know the actual probabilities involved, we instead use the obvious frequency estimates from the training data
- Here's our training data again:

Training Data

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 test at the root?

- We can place at the root of the tree a test for the values of one of the 4 possible attributes Outlook, Temperature, Humidity, or Wind
- Need to consider each in turn
- But first let's compute the entropy of the overall distribution of the target PlayTennis values: There are 5 No's and 9 Yes's, so the entropy is
  $- 5/14 * \log_2 5/14 – 9/14 * \log_2 9/14$
  $= 0.940$



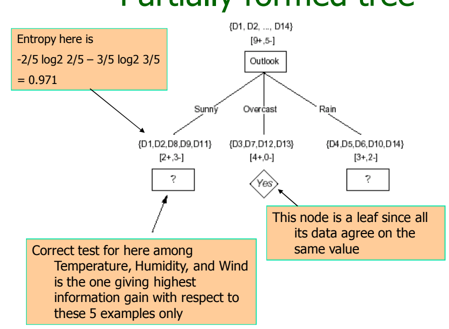
H(PlayTennis|Wind=Strong) = $- 3/6 * \log_2 3/6 - 3/6 * \log_2 3/6 = 1$
H(PlayTennis|Wind=Weak) = $- 6/8 * \log_2 6/8 - 2/8 * \log_2 2/8 = 0.811$
So the expected value is $6/14 * 1 + 8/14 * 0.811 = 0.892$
Therefore, the information gain after the Wind test is applied is
$0.940 – 0.892 = 0.048$

- Doing this for all 4 possible attribute tests yields

Attribute tested at root	Information Gain
Outlook	0.246
Temperature	0.029
Humidity	0.151
Wind	0.048

- Therefore the root should test for the value of Outlook

Partially formed tree



Extensions

- Continuous input attributes
  - Sort data on any such attribute and try to identify a high information gain threshold, forming binary split
- Continuous target attribute
  - Called a regression tree – won't deal with it here
- Avoiding overfitting   More on this later
  - Use separate validation set
  - Use tree post-pruning based on statistical tests

Extensions (continued)

- Inconsistent training data (same attribute vector classified more than one way)
  - Store more information in each leaf
- Missing values of some attributes in training data
  - Won't deal with this here
- Missing values of some attributes in a new attribute vector to be classified (or missing branches in the induced tree)
  - Send the new vector down multiple branches corresponding to all values of that attribute, then let all leaves reached contribute to result