Decision Trees

MSE 2400 EaLiCaRA
Dr. Tom Way

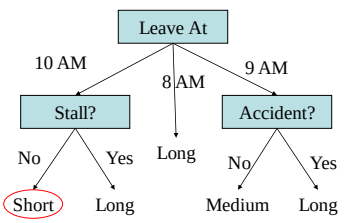
Decision Tree

- A **decision tree** is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- It is one way to display an **algorithm**.

What is a Decision Tree?

- An *inductive learning task*
  - Use particular facts to make more generalized conclusions
- A predictive model based on a branching series of Boolean tests
  - These smaller Boolean tests are less complex than a one-stage classifier
- Let's look at a sample decision tree…

Predicting Commute Time



If we leave at 10 AM and there are no cars stalled on the road, what will our commute time be?

Inductive Learning

- In this decision tree, we made a series of Boolean decisions and followed the corresponding branch
  - Did we leave at 10 AM?
  - Did a car stall on the road?
  - Is there an accident on the road?
- By answering each of these yes/no questions, we then came to a conclusion on how long our commute might take

Decision Trees as Rules

- We did not have to represent this tree graphically
- We could have represented as a set of rules. However, this may be much harder to read…

Decision Tree as a Rule Set

if hour equals 8am:
  commute time is long
else if hour equals 9am:
  if accident equals yes:
    commute time is long
  else:
    commute time is medium
else if hour equals 10am:
  if stall equals yes:
    commute time is long
  else:
    commute time is short

Notice that all attributes don't have to be used on each path of the decision.

Some attributes may not even appear in a tree.

How to Create a Decision Tree

- First, make a list of attributes that we can measure
  - These attributes (for now) must be **discrete**
- We then choose a *target attribute* that we want to predict
- Then create an *experience table* that lists what we have seen in the past

Sample Experience Table

Example	Attributes				Target
	Hour	Weather	Accident	Stall	Commute
D1	8 AM	Sunny	No	No	Long
D2	8 AM	Cloudy	No	Yes	Long
D3	10 AM	Sunny	No	No	Short
D4	9 AM	Rainy	Yes	No	Long
D5	9 AM	Sunny	Yes	Yes	Long
D6	10 AM	Sunny	No	No	Short
D7	10 AM	Cloudy	No	No	Short
D8	9 AM	Rainy	No	No	Medium
D9	9 AM	Sunny	Yes	No	Long
D10	10 AM	Cloudy	Yes	Yes	Long
D11	10 AM	Rainy	No	No	Short
D12	8 AM	Cloudy	Yes	No	Long
D13	9 AM	Sunny	No	No	Medium

Choosing Attributes

- The previous experience decision table showed 4 attributes: hour, weather, accident and stall
- But the decision tree only showed 3 attributes: hour, accident and stall
- Why is that?

Choosing Attributes (1)

- Methods for selecting attributes (which will be described later) show that weather is not a discriminating attribute
- We use the principle of *Occam's Razor*: Given a number of competing hypotheses, the simplest one is preferable

Occam's Razor

- 1852
- A "razor" is a maxim or rule of thumb
- "Entities should not be multiplied unnecessarily."

entia non sunt multiplicanda praeter necessitatem

Choosing Attributes (2)

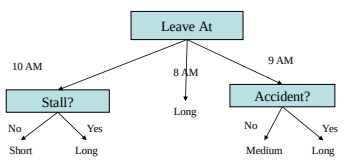
- The basic structure of creating a decision tree is the same for most decision tree algorithms
- The difference lies in how we select the attributes for the tree
- We will focus on the ID3 algorithm developed by Ross Quinlan in 1975

Decision Tree Algorithms

- The basic idea behind any decision tree algorithm is as follows:
  - Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node
  - Repeat this process for recursively for each child
  - Stop when:
    - All the instances have the same target attribute value
    - There are no more attributes
    - There are no more instances

Identifying the Best Attributes

Refer back to our original decision tree



How did we know to split on ***leave at*** then ***stall*** and ***accident*** and not ***weather***?

ID3 Heuristic

- To determine the best attribute, we look at the ID3 heuristic
- ID3 splits attributes based on their ***entropy***.
- Entropy is a measure of disinformation…

This isn't Entropy from Physics

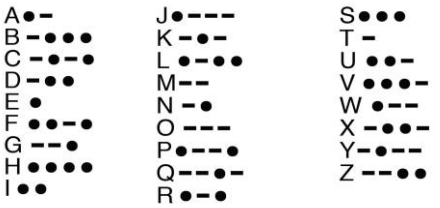
- In Physics…
- Entropy describes the concept that all things tend to move from an ordered state to a disordered state
- Hot is more organized than cold
- Tidy is more organized than neat
- Life is more organized than death
- Society is more organized than anarchy

Entropy in the real world (1)

- Entropy has to do with how rare or common an instance of information is
- It's natural for us to want to use fewer bits (send fewer messages) when reporting about common vs. rare events.
  - How often do we hear about a safe plane landing making the evening news?
  - How about a crash?
  - Why? The crash is rarer!

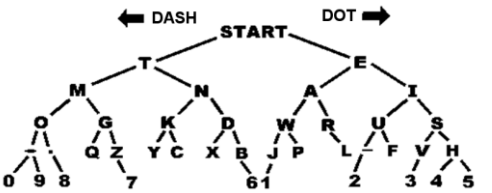
Entropy in the real world (2)

- Morse code – designed using entropy



Entropy in the real world (3)

- Morse code decision tree



How Entropy Works

- Entropy is minimized when all values of the target attribute are the same.
  - If we know that commute time will always be *short*, then entropy = 0
- Entropy is maximized when there is an equal chance of all values for the target attribute (i.e. the result is random)
  - If commute time = short in 3 instances, medium in 3 instances and long in 3 instances, entropy is maximized

Calculating Entropy

- Calculation of entropy
  - Entropy(S) = $\sum_{(i=1 \text{ to } l)}$-$|S_i|/|S|$ * $\log_2(|S_i|/|S|)$
    - S = set of examples
    - S_i = subset of S with value v_i under the target attribute
    - l = size of the range of the target attribute

ID3

- ID3 splits on attributes with the lowest entropy
- We calculate the entropy for all values of an attribute as the weighted sum of subset entropies as follows:
  - $\sum_{(i = 1 \text{ to } k)}$ $|S_i|/|S|$ Entropy(S_i), where k is the range of the attribute we are testing
- We can also measure information gain (which is inversely proportional to entropy) as follows:
  - Entropy(S) - $\sum_{(i = 1 \text{ to } k)}$ $|S_i|/|S|$ Entropy(S_i)

ID3

- Given our commute time sample set, we can calculate the entropy of each attribute at the root node

Attribute	Expected Entropy	Information Gain
Hour	0.6511	0.768449
Weather	1.28884	0.130719
Accident	0.92307	0.496479
Stall	1.17071	0.248842

Problems with ID3

- ID3 is not optimal
  - Uses *expected* entropy reduction, not actual reduction
- Must use discrete (or discretized) attributes
  - What if we left for work at 9:30 AM?
  - We could break down the attributes into smaller values…

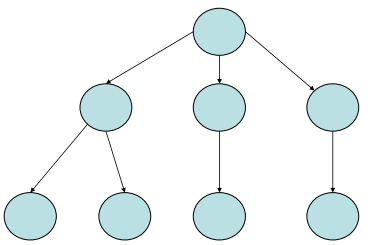
Problems with Decision Trees

- While decision trees classify quickly, the time for building a tree may be higher than another type of classifier
- Decision trees suffer from a problem of errors propagating throughout a tree
  - A very serious problem as the number of classes increases

Error Propagation

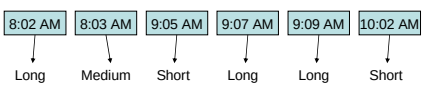
- Since decision trees work by a series of local decisions, what happens when one of these local decisions is wrong?
  - Every decision from that point on may be wrong
  - We may never return to the correct path of the tree

Error Propagation Example



Problems with ID3

- If we broke down leave time to the minute, we might get something like this:



Since entropy is very low for each branch, we have n branches with n leaves. This would not be helpful for predictive modeling.

Problems with ID3

- We can use a technique known as discretization
- We choose *cut points*, such as 9AM for splitting continuous attributes
- These cut points generally lie in a subset of *boundary points*, such that a boundary point is where two adjacent instances in a sorted list have different target value attributes

Problems with ID3

- Consider the attribute commute time

8:00 (L), 8:02 (L), 8:07 (M), 9:00 (S), 9:20 (S), 9:25 (S), 10:00 (S), 10:02 (M)

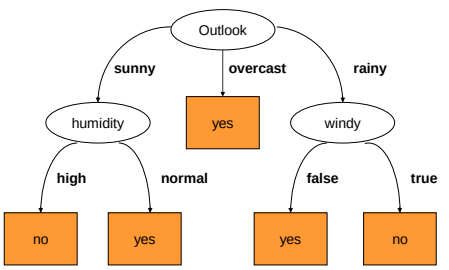
When we split on these attributes, we increase the entropy so we don't have a decision tree with the same number of cut points as leaves

A Decision Tree Example

Weather data example. Should we play tennis?

ID code	Outlook	Temperature	Humidity	Windy	Play
a	Sunny	Hot	High	False	No
b	Sunny	Hot	High	True	No
c	Overcast	Hot	High	False	Yes
d	Rainy	Mild	High	False	Yes
e	Rainy	Cool	Normal	False	Yes
f	Rainy	Cool	Normal	True	No
g	Overcast	Cool	Normal	True	Yes
h	Sunny	Mild	High	False	No
i	Sunny	Cool	Normal	False	Yes
j	Rainy	Mild	Normal	False	Yes
k	Sunny	Mild	Normal	True	Yes
l	Overcast	Mild	High	True	Yes
m	Overcast	Hot	Normal	False	Yes
n	Rainy	Mild	High	True	No

Decision tree for weather data



The Process of Constructing a Decision Tree

- Select an attribute to place at the root of the decision tree and make one branch for every possible value.
- Repeat the process recursively for each branch.

Which Attribute Should Be Placed at a Certain Node

- One common approach is based on the information gained by placing a certain attribute at this node.

Information Gained by Knowing the Result of a Decision

- In the weather data example, there are 9 instances of which the decision to play is "yes" and there are 5 instances of which the decision to play is "no'. Then, the information gained by knowing the result of the decision is

$$\frac{9}{14}\times\left(-\log\frac{9}{14}\right)+\left(\frac{5}{14}\right)\times\left(-\log\frac{5}{14}\right)=0.940\text{ bits.}$$

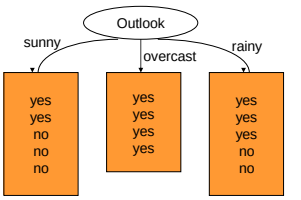
The General Form for Calculating the Information Gain

- Entropy of a decision =

$$-P_1 \times \log P_1 - P_2 \times \log P_2 - \cdots - P_n \times \log P_n$$

P_1, P_2, …, P_n are the probabilities of the n possible outcomes.

Information Further Required If "Outlook" Is Placed at the Root



Information further required =

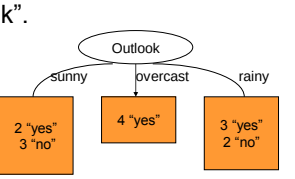
$$\left(\frac{5}{14}\right) \times 0.971 + \left(\frac{4}{14}\right) \times 0 + \left(\frac{5}{14}\right) \times 0.971 = 0.693 bits.$$

Information Gained by Placing Each of the 4 Attributes

- Gain(outlook) = 0.940 bits – 0.693 bits = 0.247 bits.
- Gain(temperature) = 0.029 bits.
- Gain(humidity) = 0.152 bits.
- Gain(windy) = 0.048 bits.

The Strategy for Selecting an Attribute to Place at a Node

- Select the attribute that gives us the largest information gain.
- In this example, it is the attribute "Outlook".



The Recursive Procedure for Constructing a Decision Tree

- The operation discussed above is applied to each branch recursively to construct the decision tree.
- For example, for the branch "Outlook = Sunny", we evaluate the information gained by applying each of the remaining 3 attributes.
  - Gain(Outlook=sunny;Temperature) = 0.971 – 0.4 = 0.571
  - Gain(Outlook=sunny;Humidity) = 0.971 – 0 = 0.971
  - Gain(Outlook=sunny;Windy) = 0.971 – 0.951 = 0.02

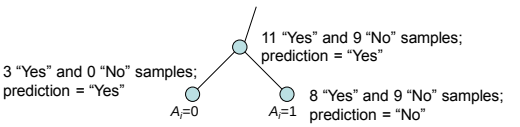
Recursive Procedure (cont'd)

- Similarly, we also evaluate the information gained by applying each of the remaining 3 attributes for the branch "Outlook = rainy".
  - Gain(Outlook=rainy;Temperature) = 0.971 – 0.951 = 0.02
  - Gain(Outlook=rainy;Humidity) = 0.971 – 0.951 = 0.02
  - Gain(Outlook=rainy;Windy) =0.971 – 0 = 0.971

The Over-fitting Issue

- Over-fitting is caused by creating decision rules that work accurately on the training set based on insufficient quantity of samples.
- As a result, these decision rules may not work well in more general cases.
- Also called the “Training Effect”

Example of the Over-fitting Problem in Decision Tree Construction



$$\text{Entropy at the subroot} = -\left(\frac{11}{20}\log_2\frac{11}{20}+\frac{9}{20}\log_2\frac{9}{20}\right)$$
$$= 0.993 \text{ bits}$$

$$\text{Average entropy at the children} = -\frac{17}{20}\left(\frac{8}{17}\log_2\frac{8}{17}+\frac{9}{17}\log_2\frac{9}{17}\right)$$
$$= 0.848 \text{ bits}$$

Summary

- Decision trees can be used to help predict the future based on past experience…
- That is… an example of machine learning
- Trees are easy to understand
- Decision trees work more efficiently with discrete attributes
- Trees may suffer from error propagation