Lec Wed

Wednesday, October 10, 2018 4:00 PM

Recap of Monday

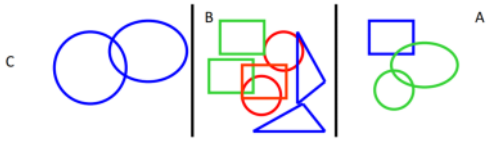
- Modeling
  - How to formulate your problem?
- Representation
  - What is the input/output and hypothesis space?

Decision Tree

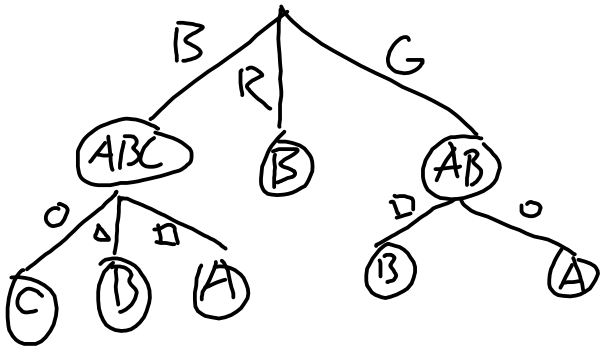
- a non-linear model
- classification: leaves are the labels
- regression: leaves are numerical

Ex:

- What features we can used?
- What is the label for a red triangle?



Our decision tree might look like:

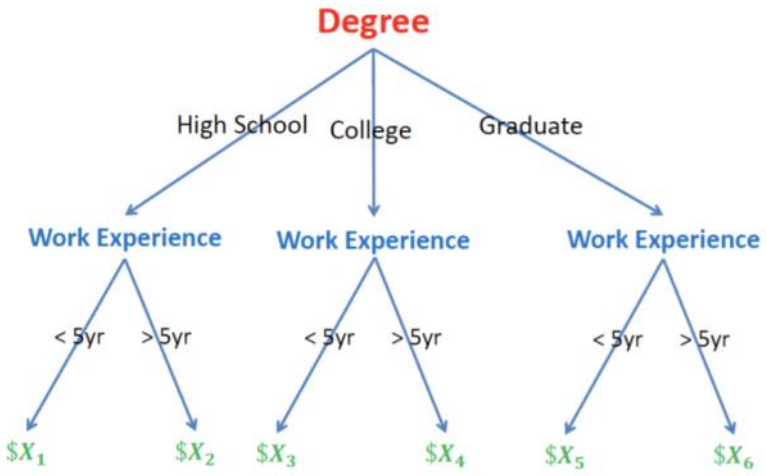


where we first look at color then shape in order to determine which group (A, B, or C) the object falls under

The representation

- at each level (or node) you ask a question and make a "Decision"
- edges represent label values

Ex: Company deciding your salary



note: if a feature is real-valued, we might use a threshold (like 5yr, above) to make it binary

Advantages of Decision Tree:

- can represent any boolean function
  - conversely, decision trees can be represented as boolean functions
- can be viewed as a way to compactly represent a lot of data
- the **evaluation** of Decision Tree Classifier is easy
  - anybody could look at one and follow it
  - programmatically, it's a bunch of conditionals

Challenges

- given data, there are many ways to represent it as a decision tree
- learning a good representation iss the challenge

Learning a decision tree
Ex: Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

23

How do we build a decision tree from this?
What do we need to learn?

- structure of tree
- thresholds

- values for the leaves

Decision Tree Algorithm: ID3(S, Attributes, Label)

- recursively build a decision tree top down
- base case: if all examples are labeled the same, return single node tree with that label
- otherwise:
    - create root node for tree
    - A = attribute in Attributes that best classifies S
    - for each possible value v of A
        - Add a new tree branch corresponding to A=v
        - Let Sv be the subset of examples in S with A=v
        - if Sv is empty:
            - add leaf node with the common value of Label in S
        - else:
            - below this branch add the subtree ID3(Sv, Attributes - {a}, Label)
    - return root

How do we decide which attribute is the best to use to split?

- goal: resulting decision tree as small as possible
    - but finding minimal decision tree consistent with data is NP-hard
- we'll use a heuristic called **information gain**
    - how to measure information gain?
        - idea: gaining information reduces uncertainty
        - uncertainty can be measured by entropy

Entropy

$$H[S] = -\sum_{v=1}^{K} P(S = a_v) \log_2 P(S = a_v)$$

- Measures the amount of uncertainty of a random variable with a specific probability distribution.
- The higher it is, the less confident we are in its outcome

Suppose we need to send a message. On average, how many bits do we need to send the message?

- minimize #bits/length of message
- Ex: suppose we have four possible tokens (a, b, c, d). What's the best way to encode them?
    - if the only token that shows up is a, we never need to send a message since it's just one (very low entropy)
    - if they're uniformly distributed, we would encode them as 00, 01, 10, 11 (pretty good entropy)
    - what if we only had 3 uniformly distributed letters? we could encode them as 10, 11, 0

Information Gain

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

See slides for entropy + information gain calculation example