CSC 411: Lecture 06: Decision Trees

Raquel Urtasun & Rich Zemel

University of Toronto

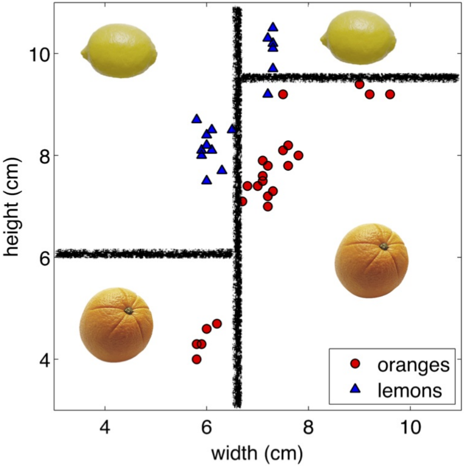
Sep 30, 2015

Today

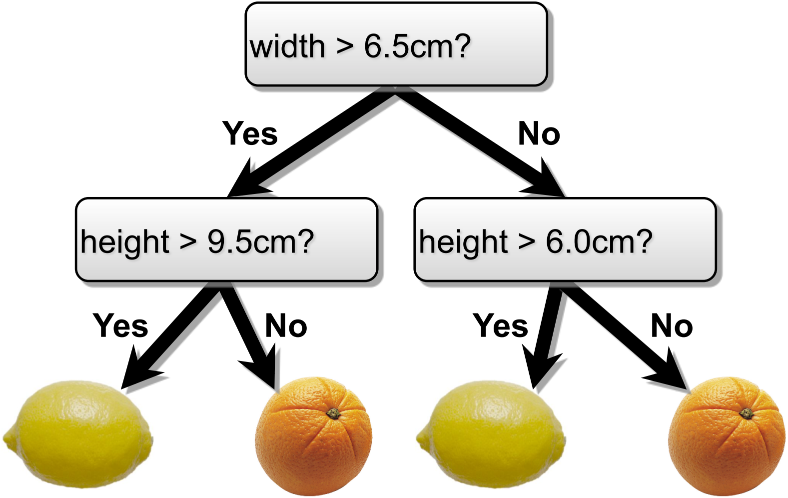
- Decision Trees
  - entropy
  - mutual information

Another Classification Idea

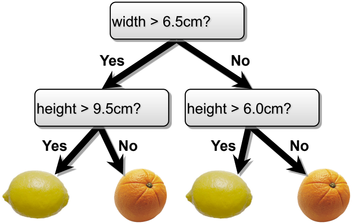
- We could view the decision boundary as being the composition of several simple boundaries.



Decision Tree: Example



Decision Trees



- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)
- In general, a decision tree can represent any binary function

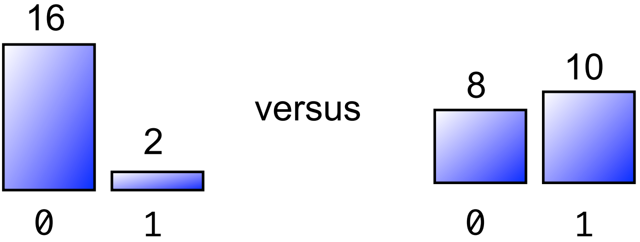
Decision Tree: Algorithm

- Choose an attribute on which to descend at each level.
- Condition on earlier (higher) choices.
- Generally, restrict only one dimension at a time.
- Declare an output value when you get to the bottom
- In the orange/lemon example, we only split each dimension once, but that is not required.
- How do you construct a useful decision tree?
- We use information theory to guide us

Two Binary Sequences

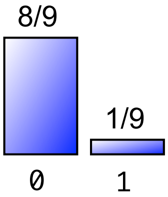
Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

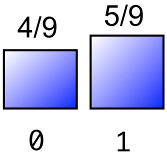
Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



Quantifying Uncertainty

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$





$$\frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} \approx \frac{1}{2}$$

$$\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99$$

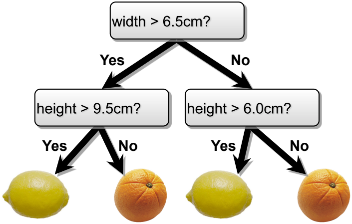
- How surprised are we by a new value in the sequence?
- How much information does it convey?

Quantifying Uncertainty: Shannon Entropy

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- Shannon Entropy is an extremely powerful concept.
- It tells you how much you can compress your data!

Decision Tree: Algorithm



- Choose an attribute on which to descend at each level.
- Condition on earlier (higher) choices
- Generally, restrict only one dimension at a time.
- How do you construct a useful decision tree?
- We use information theory to guide us

Entropy of a Joint Distribution

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

$$\begin{aligned} H(X,Y) &= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y) \\ &= -\frac{24}{100}\log_2\frac{24}{100} - \frac{1}{100}\log_2\frac{1}{100} - \frac{25}{100}\log_2\frac{25}{100} - \frac{50}{100}\log_2\frac{50}{100} \\ &\approx 1.56\text{bits} \end{aligned}$$

Specific Conditional Entropy

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- What is the entropy of cloudiness, given that it is raining?

$$
\begin{aligned}
H(X|Y=y) &= \sum_{x \in X} p(x|y) \log_2 p(x|y) \\
&= -\frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25} \\
&\approx 0.24 \text{bits}
\end{aligned}
$$

(Non-Specific) Conditional Entropy

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- The expected conditional entropy:

$$H(X|Y) = \sum_{y \in Y} p(y) H(X|Y=y)$$

$$= -\sum_{y \in Y} \sum_{x \in X} p(x,y) \log_2 p(x|y)$$

(Non-Specific) Conditional Entropy

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$$
\begin{aligned}
H(X|Y) &= \sum_{y \in Y} p(y) H(X|Y=y) \\
&= \frac{1}{4} H(\text{clouds}|\text{is raining})) + \frac{3}{4} H(\text{clouds}|\text{not raining}) \\
&\approx 0.75 \text{ bits}
\end{aligned}
$$

Mutual Information

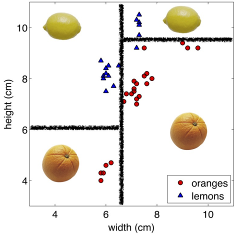
	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

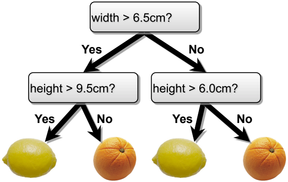
- How much information about cloudiness do we get by discovering whether it is raining?

$$
\begin{aligned}
IG(X|Y) &= H(X) - H(X|Y) \\
&\approx 0.25 \text{ bits}
\end{aligned}
$$

- Also called information gain in X due to Y
- For decision trees, X is the class/label and Y is an attribute

Constructing Decision Trees





- I made the fruit data partitioning just by eyeballing it.
- We can use the mutual information to automate the process.
- At each level, one must choose:
  1. Which variable to split.
  2. Possibly where to split it.
- Choose them based on how much information we would gain from the decision!

Decision Tree Algorithm

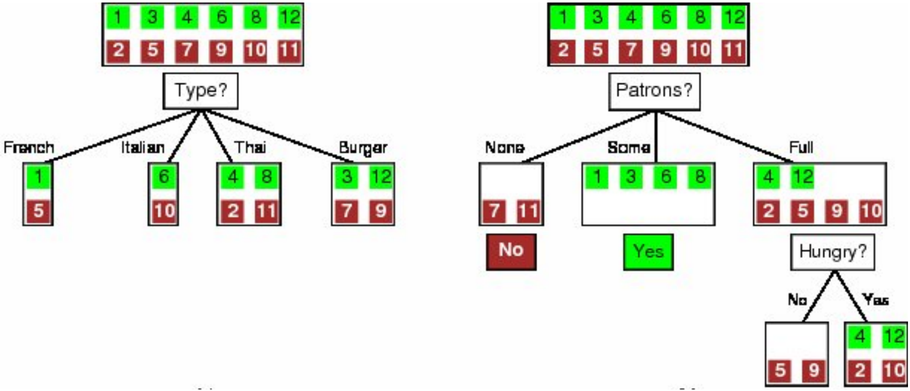
- Simple, greedy, recursive approach, builds up tree node-by-node

1. pick an attribute to split at a non-terminal node
2. split examples into groups based on attribute value
3. for each group:
   - if no examples – return majority from parent
   - else if all examples in same class – return class
   - else loop to step 1

Ex.	Attributes										Target
	Alt	Ba^r	*Fri*	Hu^n	*Pat*	*Price*	*Rain*	*Res*	Ty^{pe}	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
$X_1{}^1$	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
$X_1{}^2$	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Russell & Norvig example

Attribute Selection

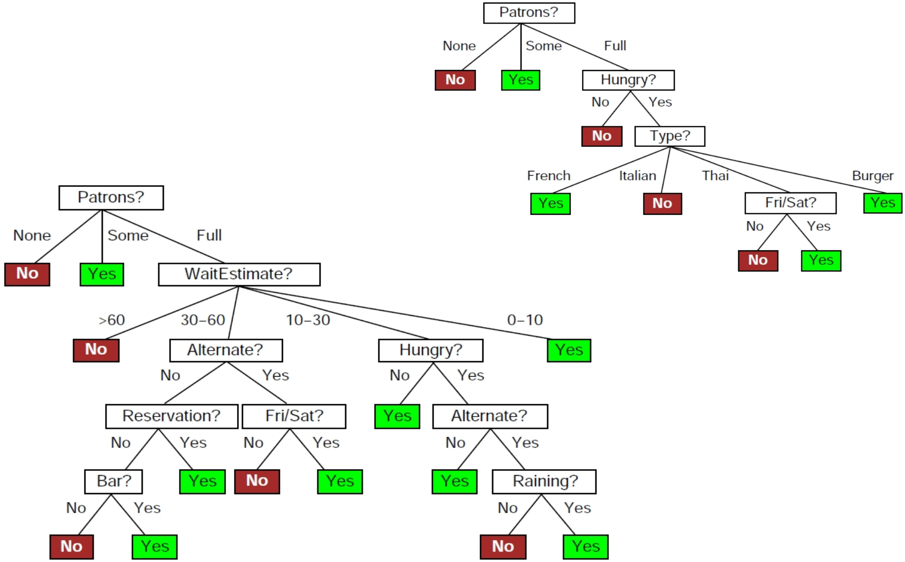


$$IG(Y) = H(X) - H(X|Y)$$

$$IG(type) = 1 - \left[\frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4})\right] = 0$$

$$IG(Patrons) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})\right] \approx 0.541$$

Which Tree is Better?



What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
- Occam's Razor: find the simplest hypothesis (smallest tree) that fits the observations
- Inductive bias: small trees with informative nodes near the root

Decision Tree Miscellany

- Problems:
  - You have exponentially less data at lower levels.
  - Too big of a tree can overfit the data.
  - Greedy algorithms don't necessarily yield the global optimum.
- In practice, one often regularizes the construction process to try to get small but highly-informative trees.
- Decision trees can also be used for regression on real-valued outputs, but it requires a different formalism.

Comparison to k-NN

K-Nearest Neighbors

- Decision boundaries: piece-wise
- Test complexity: non-parametric, few parameters besides (all?) training examples

Decision Trees

- Decision boundaries: axis-aligned, tree structured
- Test complexity: attributes and splits

Applications of Decision Trees

- Can express any Boolean function, but most useful when function depends critically on few attributes
- Bad on: parity, majority functions; also not well-suited to continuous attributes
- Practical Applications:
  - Flight simulator: 20 state variables; 90K examples based on expert pilot's actions; auto-pilot tree
  - Yahoo Ranking Challenge
  - Random Forests