

CSC 411: Lecture 06 - Decision Trees

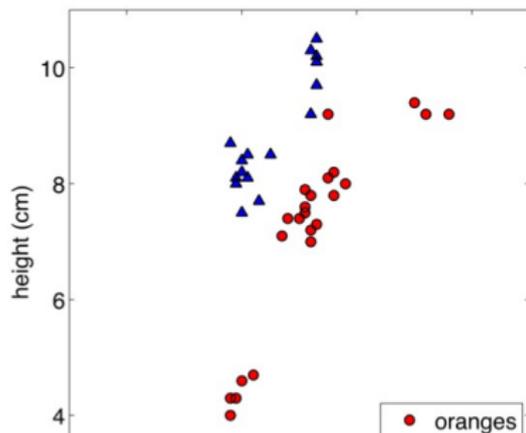
Ethan Fetaya, James Lucas and Emad Andrews

University of Toronto

- Decision Trees
 - ▶ entropy
 - ▶ information gain

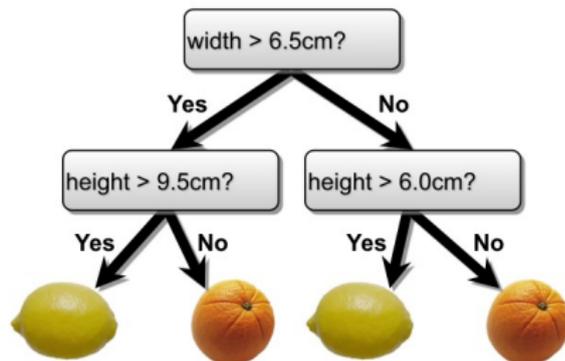
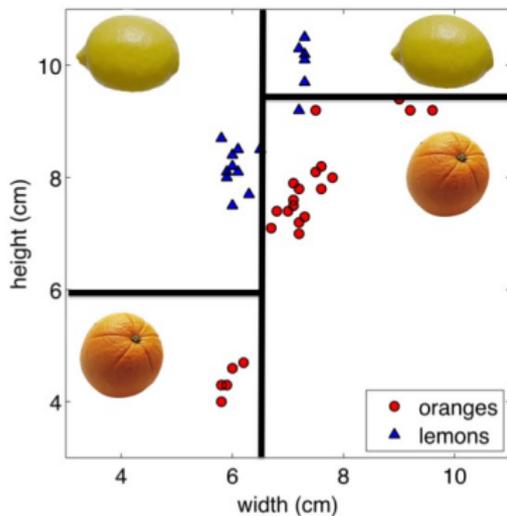
Another Classification Idea

- We learned about linear classification (e.g., logistic regression), and nearest neighbors. Any other idea?
- Pick an attribute, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign a class with majority vote
- Do other branches as well

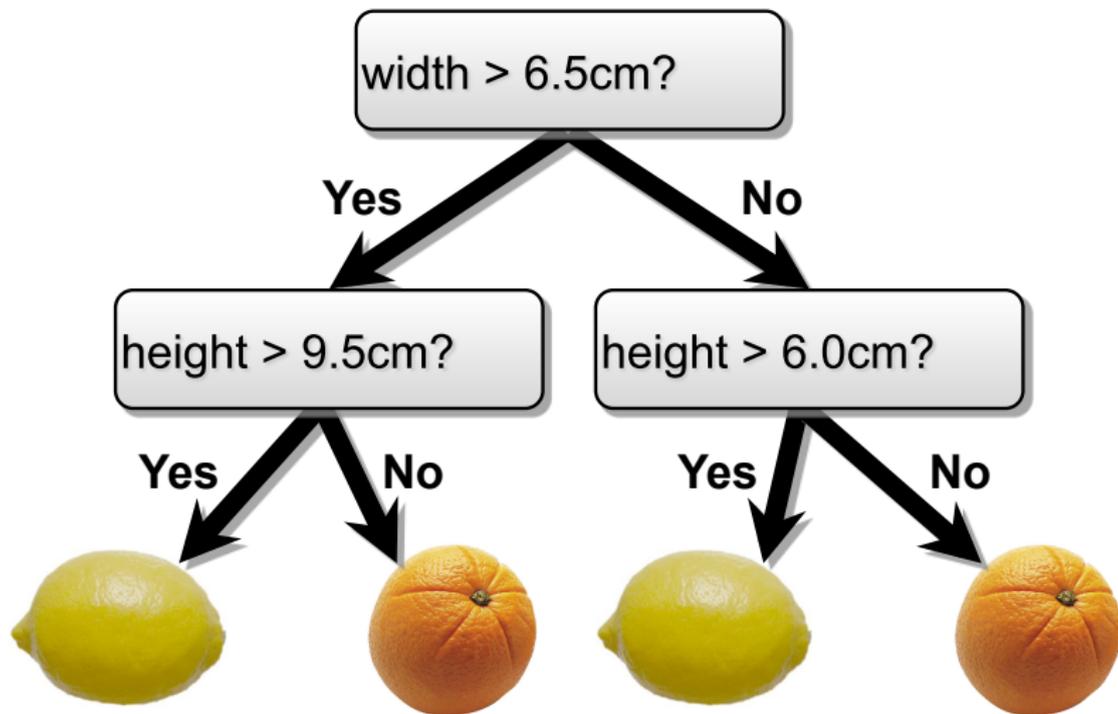


Another Classification Idea

- Gives axes aligned decision boundaries

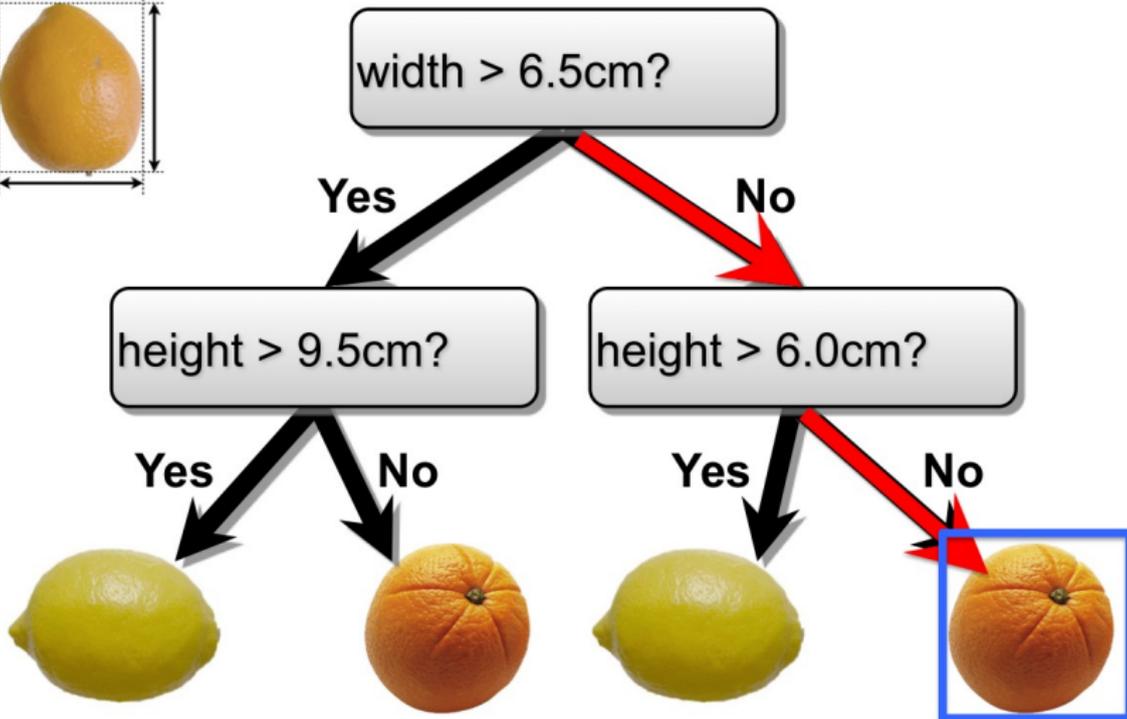
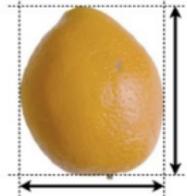


Decision Tree: Example



Decision Tree: Classification

Test example



Example with Discrete Inputs

- What if the attributes are discrete?

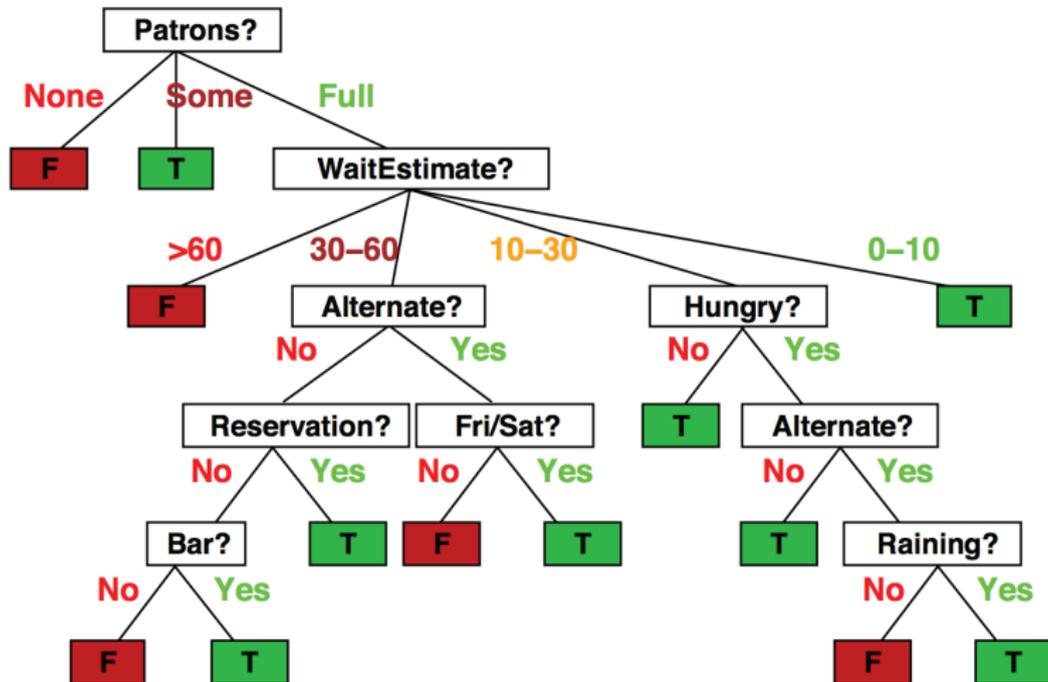
Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
x_1	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	$y_1 = \text{Yes}$
x_2	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	$y_2 = \text{No}$
x_3	No	Yes	No	No	Some	\$	No	No	Burger	0-10	$y_3 = \text{Yes}$
x_4	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10-30	$y_4 = \text{Yes}$
x_5	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 = \text{No}$
x_6	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	$y_6 = \text{Yes}$
x_7	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	$y_7 = \text{No}$
x_8	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	$y_8 = \text{Yes}$
x_9	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 = \text{No}$
x_{10}	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	$y_{10} = \text{No}$
x_{11}	No	No	No	No	None	\$	No	No	Thai	0-10	$y_{11} = \text{No}$
x_{12}	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	$y_{12} = \text{Yes}$

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range (\$, \$\$, \$\$\$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

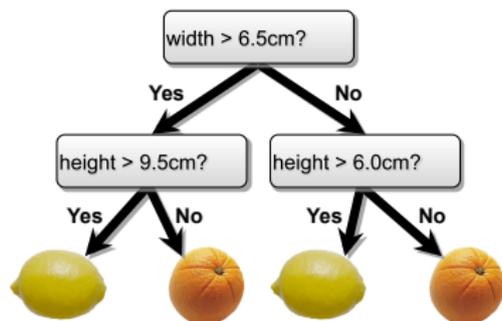
Attributes:

Decision Tree: Example with Discrete Inputs

- The tree to decide whether to wait (T) or not (F)



Decision Trees



- Internal nodes **test attributes**
- Branching is determined by **attribute value**
- Leaf nodes are **outputs** (class assignments)

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

Decision Tree: Classification and Regression

- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \dots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- **Classification tree:**
 - ▶ discrete output
 - ▶ leaf value y^m typically set to the most common value in $\{t^{(m_1)}, \dots, t^{(m_k)}\}$
- **Regression tree:**
 - ▶ continuous output
 - ▶ leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \dots, t^{(m_k)}\}$

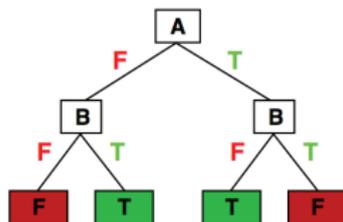
Note: We will only talk about classification

[Slide credit: S. Russell]

- **Discrete-input, discrete-output case:**

- ▶ Decision trees can express any function of the input attributes
- ▶ E.g., for Boolean functions, truth table row \rightarrow path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- **Continuous-input, continuous-output case:**

- ▶ Can approximate any function arbitrarily closely
- Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples

Need some kind of regularization to ensure more **compact** decision trees

[Slide credit: S. Russell]

How do we Learn a DecisionTree?

- How do we construct a useful decision tree?

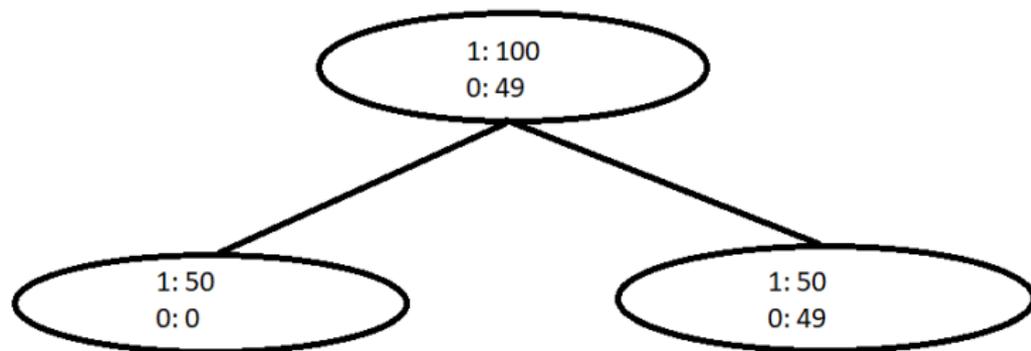
Learning Decision Trees

Learning the simplest (smallest) decision tree is an NP complete problem [if you are interested, check: Hyafil & Rivest'76]

- Resort to a **greedy heuristic**:
 - ▶ Start from an empty decision tree
 - ▶ Split on next best attribute
 - ▶ Recurse
- What is **best** attribute?
- How about accuracy?

Choosing a Good Split

- Why isn't accuracy a good measure?



- Is this split good? Zero accuracy gain.
- We will use intuition from [information theory](#)

Idea: Use counts at leaves to define probability distributions, so we can measure uncertainty

Choosing a Good Split

- Which attribute is better to split on, X_1 or X_2 ?
 - ▶ Deterministic: good (all are true or false; just one class in the leaf)
 - ▶ Uniform distribution: bad (all classes in leaf equally probable)
 - ▶ What about distributions in between?

Note: Let's take a slight detour and remember concepts from information theory

[Slide credit: D. Sontag]

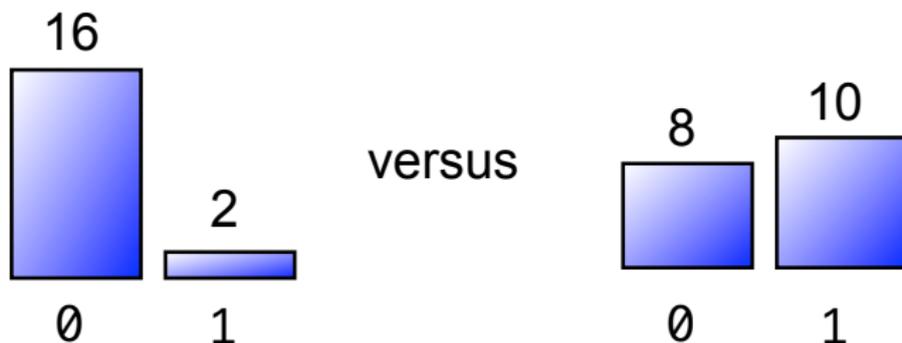
We Flip Two Different Coins

Sequence 1:

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:

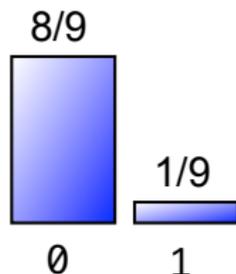
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



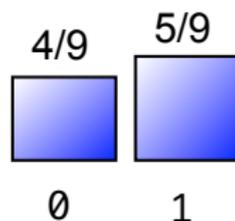
Quantifying Uncertainty

Entropy H :

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



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

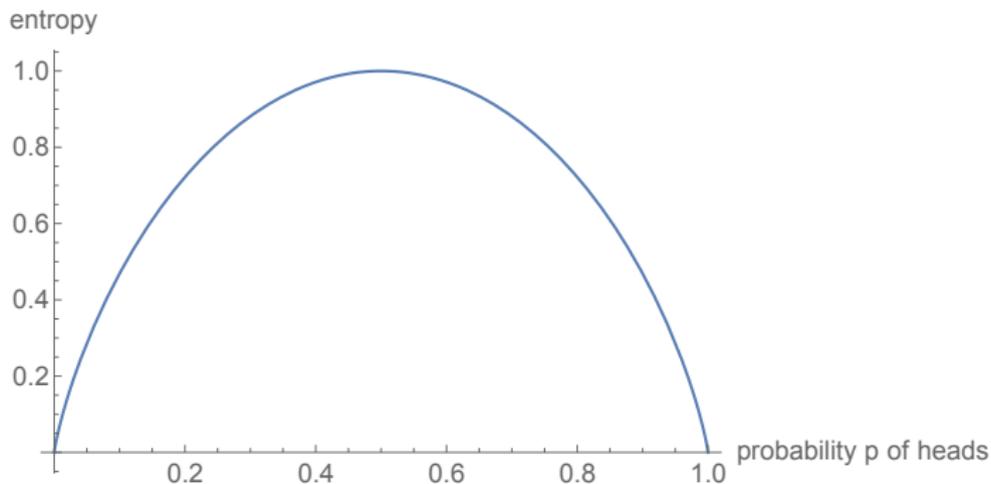


$$-\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99$$

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

Quantifying Uncertainty

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



- **“High Entropy”**:
 - ▶ Variable has a uniform like distribution
 - ▶ Flat histogram
 - ▶ Values sampled from it are less predictable
- **“Low Entropy”**
 - ▶ Distribution of variable has many peaks and valleys
 - ▶ Histogram has many lows and highs
 - ▶ Values sampled from it are more predictable

[Slide credit: Vibhav Gogate]

Entropy of a Joint Distribution

- Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	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

- Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

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

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

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

- We used: $p(y|x) = \frac{p(x,y)}{p(x)}$, and $p(x) = \sum_y p(x,y)$ (sum in a row)

Conditional Entropy

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

- The expected conditional entropy:

$$\begin{aligned}H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x)\end{aligned}$$

Conditional Entropy

- Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	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?

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

Conditional Entropy

- Some useful properties:
 - ▶ H is always non-negative
 - ▶ Chain rule: $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
 - ▶ If X and Y independent, then X doesn't tell us anything about Y :
 $H(Y|X) = H(Y)$
 - ▶ But Y tells us everything about Y : $H(Y|Y) = 0$
 - ▶ By knowing X , we can only decrease uncertainty about Y :
 $H(Y|X) \leq H(Y)$

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

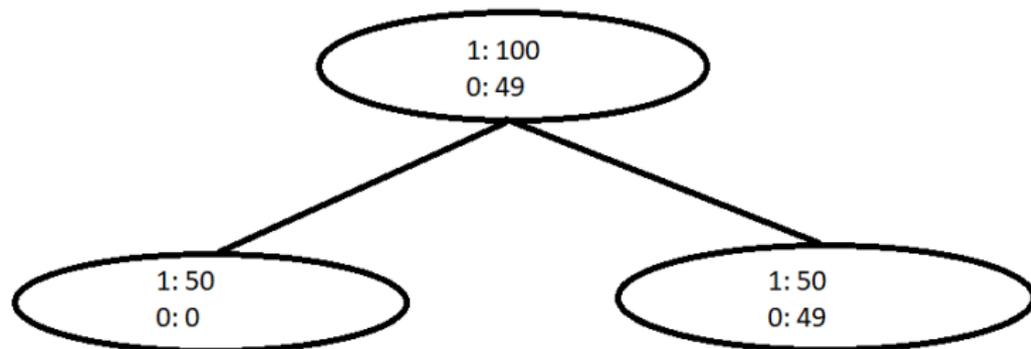
- How much information about cloudiness do we get by discovering whether it is raining?

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

- Also called **information gain** in Y due to X
- If X is completely uninformative about Y : $IG(Y|X) = 0$
- If X is completely informative about Y : $IG(Y|X) = H(Y)$
- How can we use this to construct our decision tree?

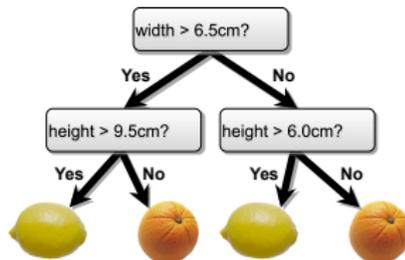
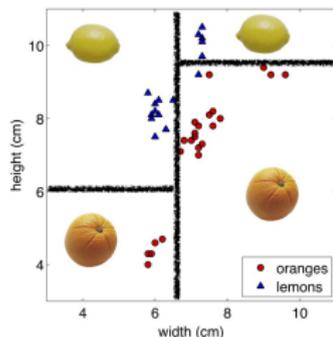
Revisiting Our Original Example

- What is the information gain of this split?



- Root entropy: $H(Y) = -\frac{49}{149} \log_2\left(\frac{49}{149}\right) - \frac{100}{149} \log_2\left(\frac{100}{149}\right) \approx 0.91$
- Leaf entropy: $H(Y|left) = 0$, $H(Y|right) \approx 1$.
- $IG(split) \approx 0.91 - \left(\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 1\right) \approx 0.24 > 0$

Constructing Decision Trees



- I made the fruit data partitioning just by eyeballing it.
- We can use the **information gain** 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! (choose attribute that gives the highest gain)

Decision Tree Construction 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

Back to Our Example

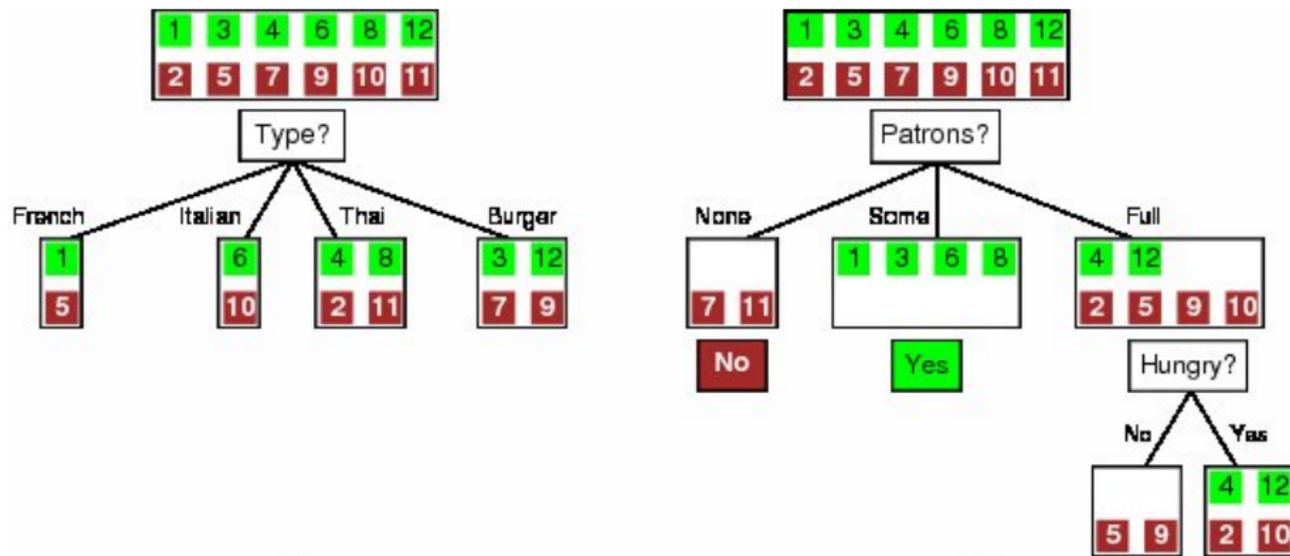
Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
x_1	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	$y_1 = \text{Yes}$
x_2	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	$y_2 = \text{No}$
x_3	No	Yes	No	No	Some	\$	No	No	Burger	0-10	$y_3 = \text{Yes}$
x_4	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10-30	$y_4 = \text{Yes}$
x_5	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 = \text{No}$
x_6	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	$y_6 = \text{Yes}$
x_7	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	$y_7 = \text{No}$
x_8	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	$y_8 = \text{Yes}$
x_9	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 = \text{No}$
x_{10}	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	$y_{10} = \text{No}$
x_{11}	No	No	No	No	None	\$	No	No	Thai	0-10	$y_{11} = \text{No}$
x_{12}	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	$y_{12} = \text{Yes}$

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range (\$, \$\$, \$\$\$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Attributes:

[from: Russell & Norvig]

Attribute Selection

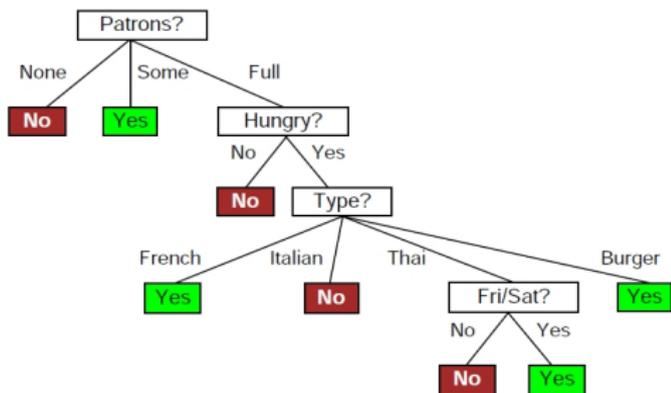
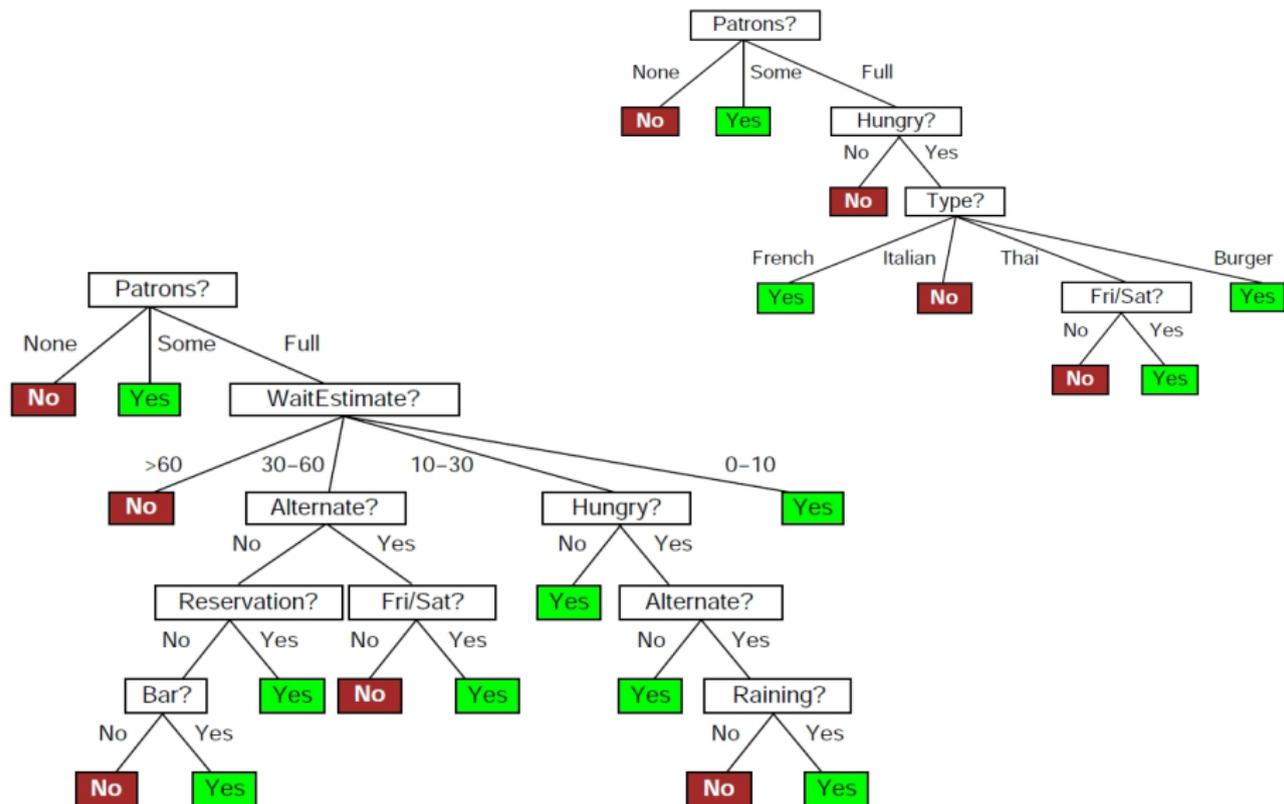


$$IG(Y) = H(Y) - H(Y|X)$$

$$IG(\text{type}) = 1 - \left[\frac{2}{12} H(Y|Fr.) + \frac{2}{12} H(Y|It.) + \frac{4}{12} H(Y|Thai) + \frac{4}{12} H(Y|Bur.) \right] = 0$$

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

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
 - ▶ Pruning.
 - ▶ Minimum gain for split.
 - ▶ Minimum node size.
- Decision trees can also be used for regression on real-valued outputs, but it requires a different formalism

K-Nearest Neighbors

- Decision boundaries: piece-wise linear
- 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: Xbox!

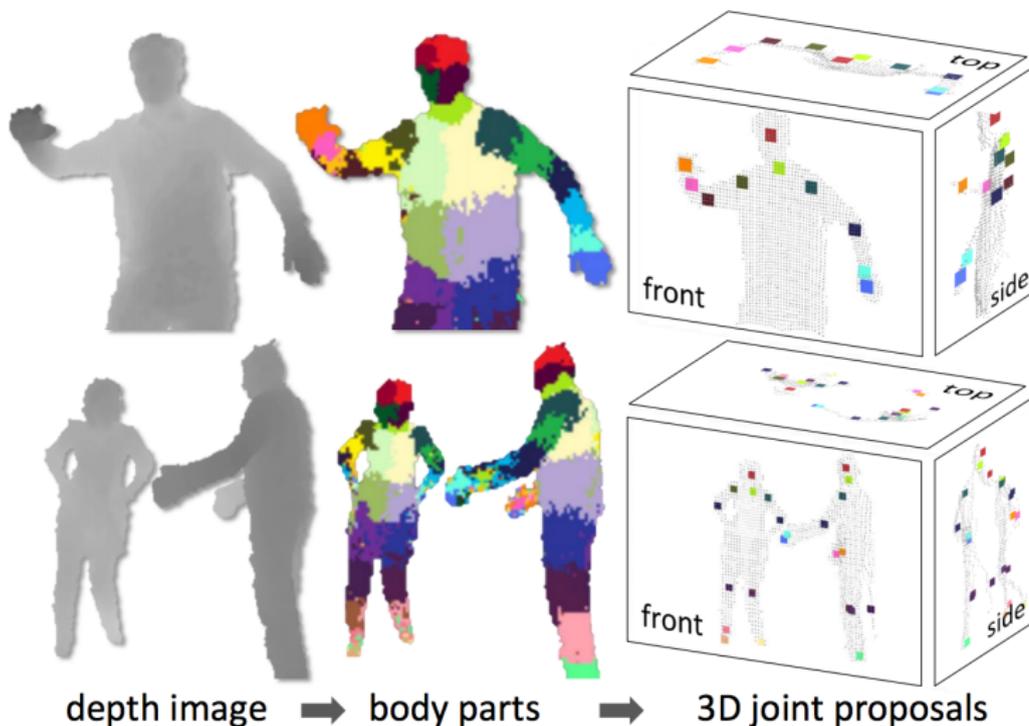
- Decision trees are in Xbox



[J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image. CVPR'11]

Applications of Decision Trees: XBox!

- Decision trees are in XBox: Classifying body parts



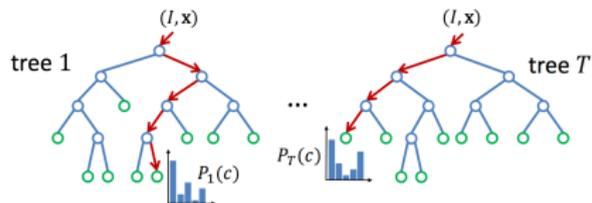
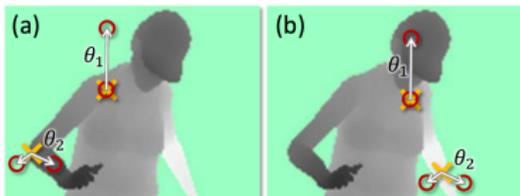
Applications of Decision Trees: XBox!

- Trained on million(s) of examples

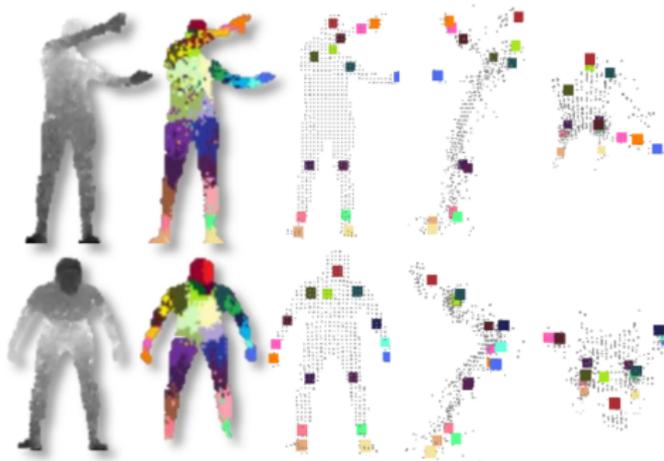


Applications of Decision Trees: XBox!

- Trained on million(s) of examples



- Results:



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:
 - ▶ Tubular data
 - ▶ Flight simulator: 20 state variables; 90K examples based on expert pilot's actions; auto-pilot tree
 - ▶ Yahoo Ranking Challenge
 - ▶ Random Forests: Microsoft Kinect Pose Estimation