ITCS 6156/8156 Fall 2024
Machine Learning

Decision Trees

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Tue & Thu, 4:00 PM – 5:15 PM, WWH 130

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

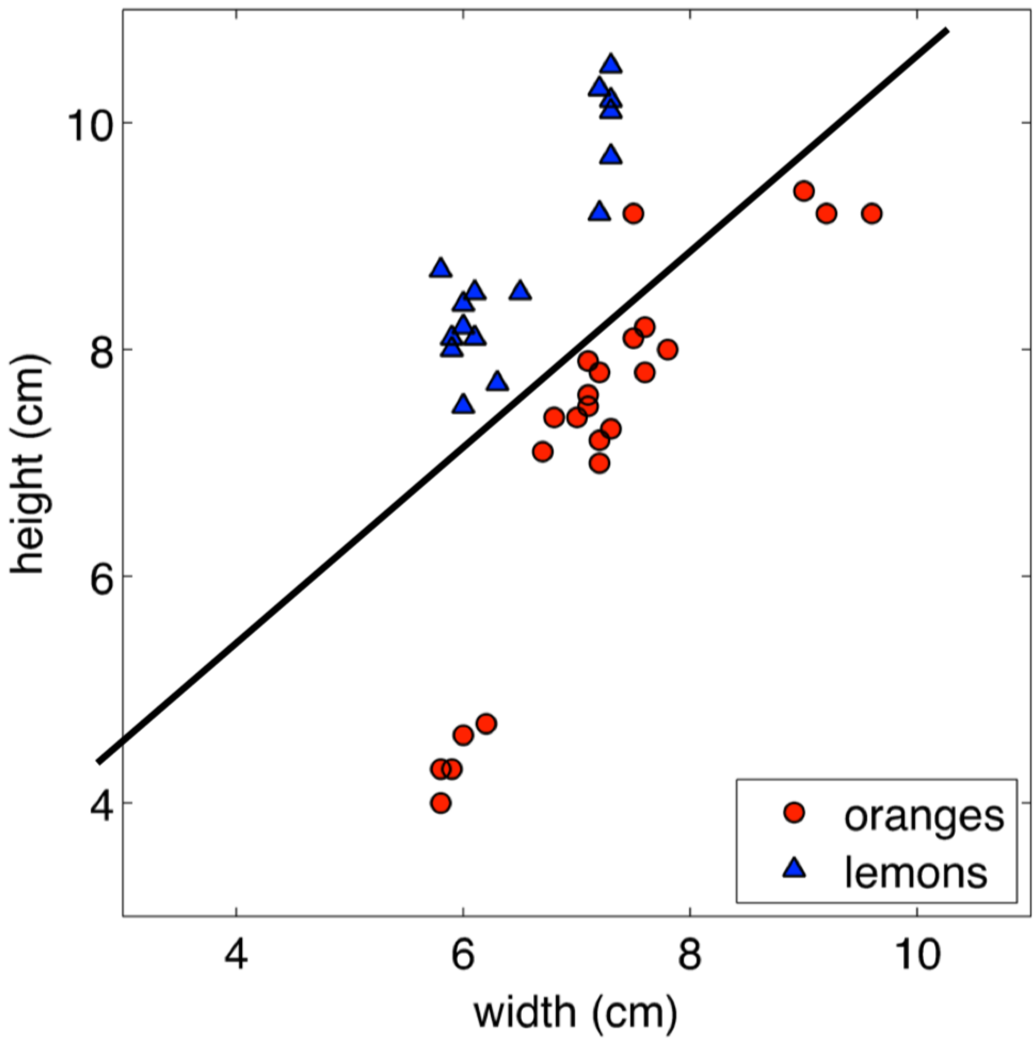
Some content in the slides is based on Dr. Raquel Urtasun’s lecture

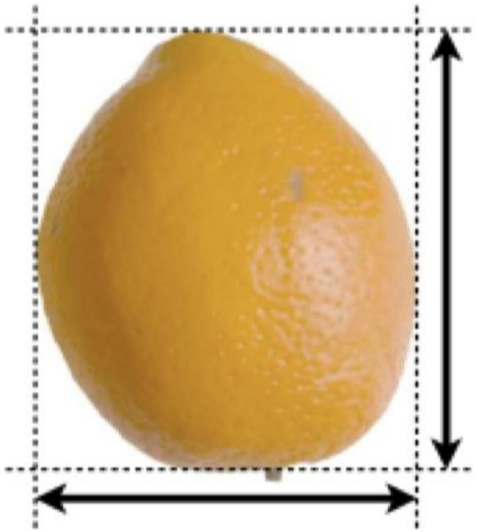
Another Classification Idea

- We learned about linear classification (e.g., logistic regression), and nearest neighbors. Any other idea?

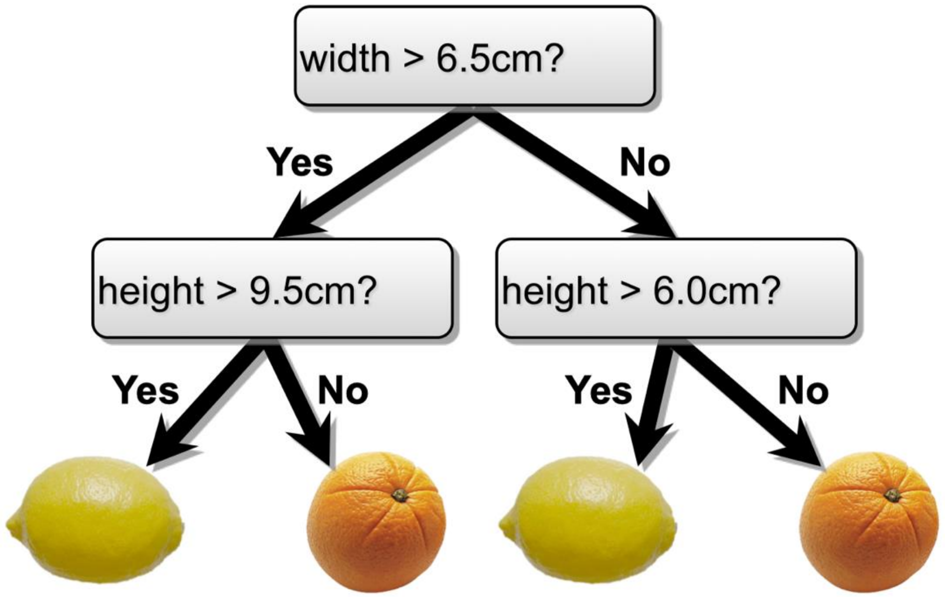
- Pick an attribute, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign a class with majority vote
- Do other branches as well

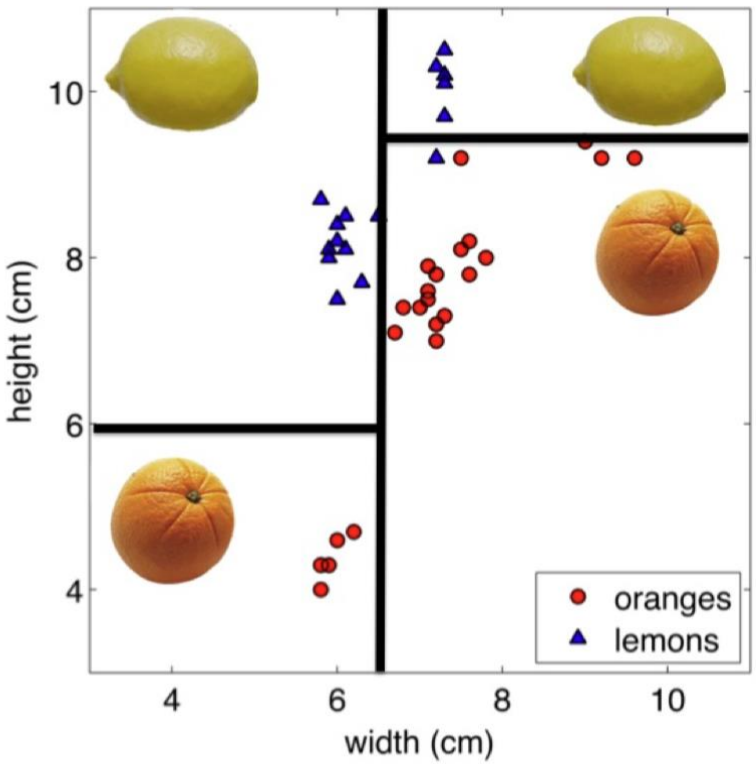
Another Classification Idea



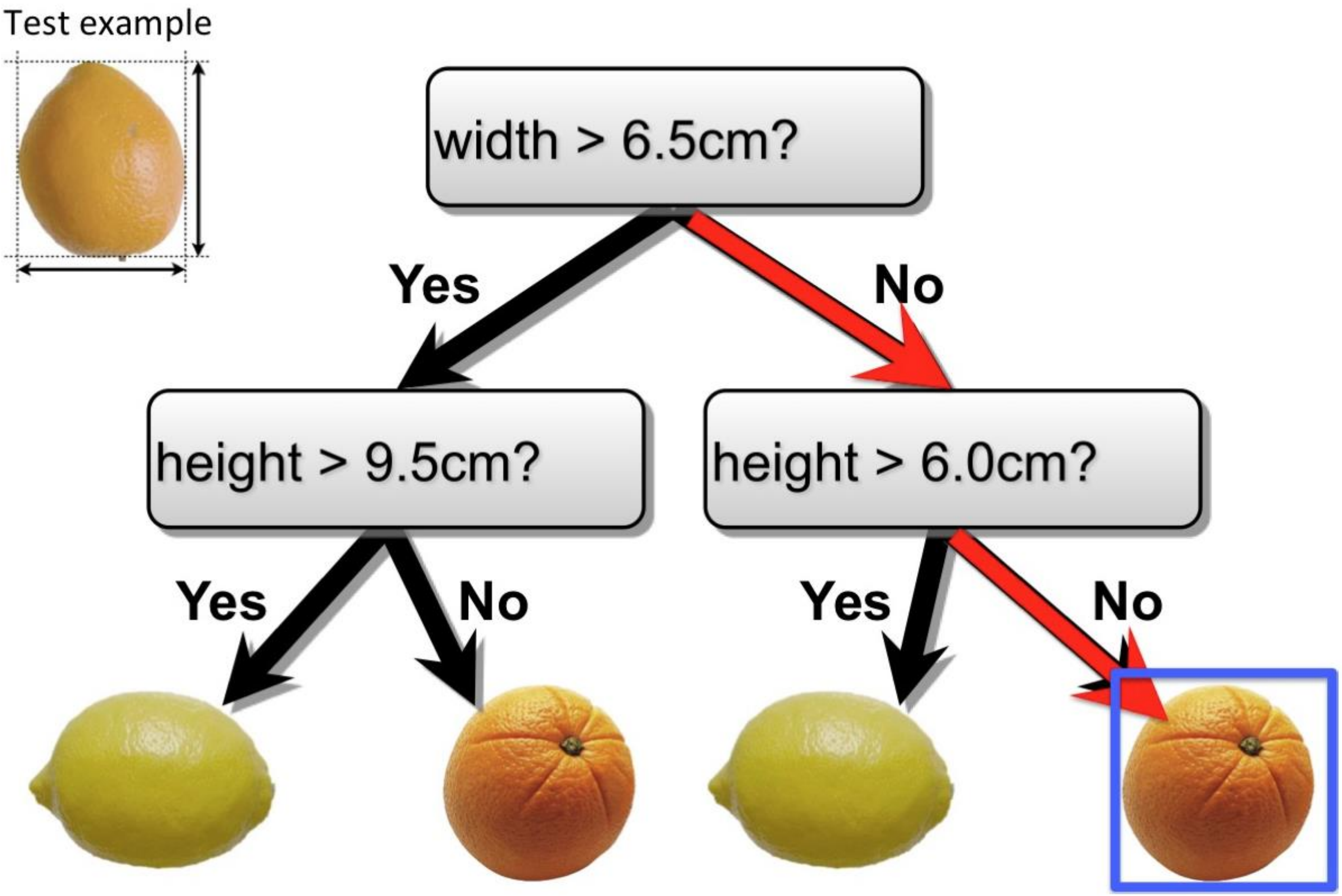


Another Classification Idea





Another Classification Idea



Example with Discrete Inputs

- What if the attributes are discrete?

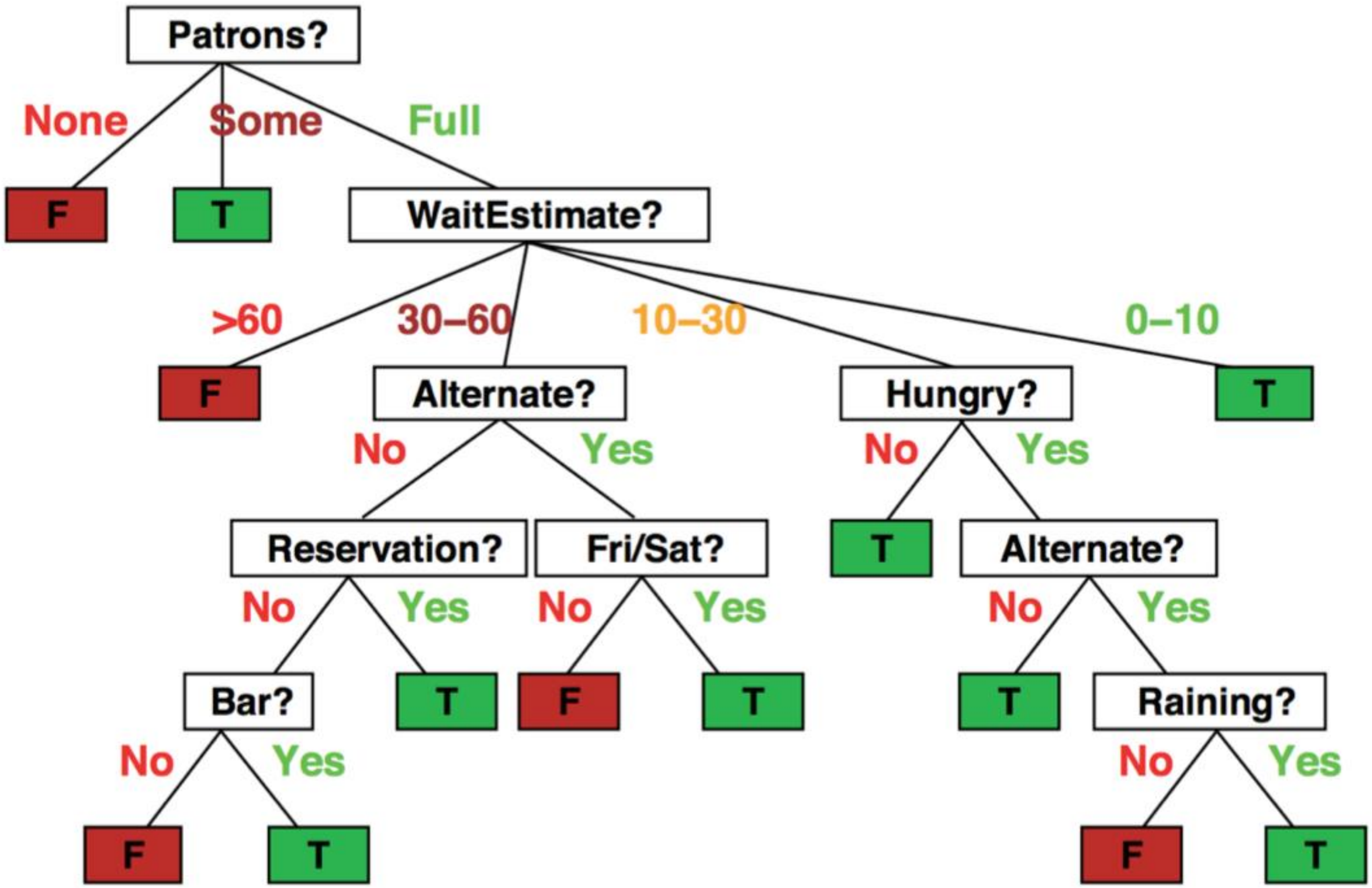
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	\$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	\$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	\$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	\$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30–60	$y_{12} =$ Yes

Attributes:

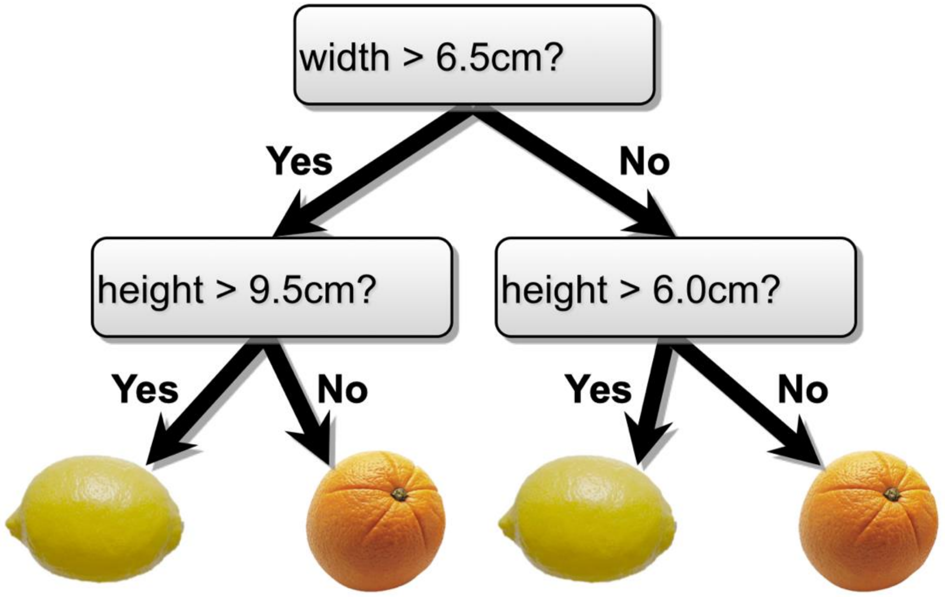
1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
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).

Example with Discrete Inputs

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



Decision Trees



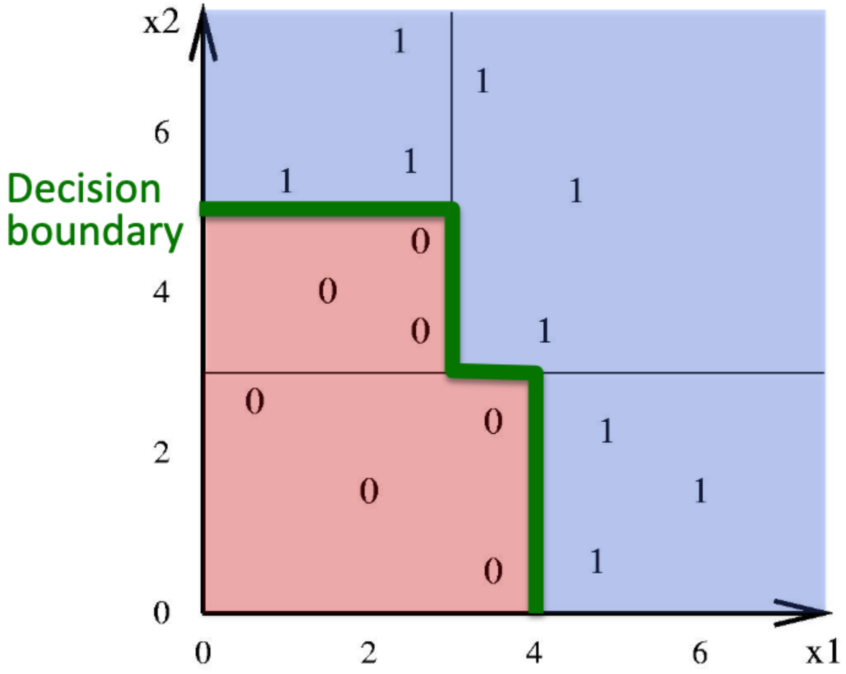
- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)

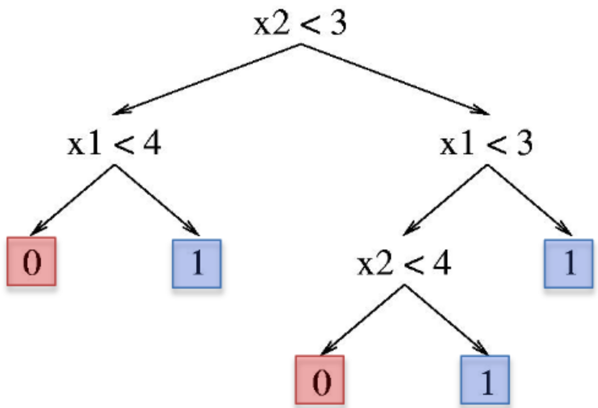
Decision Tree Algorithm

- 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 Boundary

- Decision trees divide the feature space into axis- parallel (hyper-) rectangles.
- Each rectangular region is labeled with one label (or a probability distribution over labels).





Classification and Regression

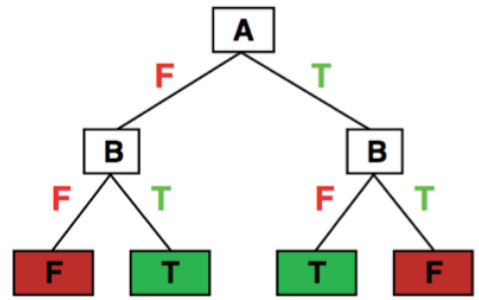
- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- **Classification tree**:
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)},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

Note: We will only talk about classification

Expressiveness

- **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

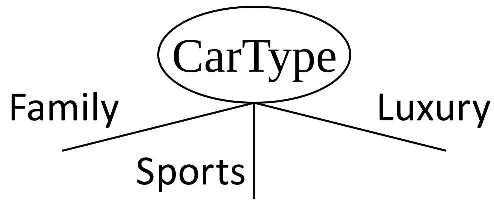
How to Specify Test Condition?

- **Depends on attribute types**
  - Nominal
  - Ordinal
  - Continuous

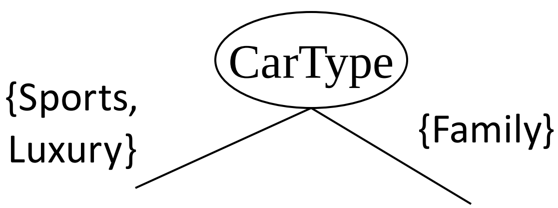
- **Depends on number of ways to split**
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

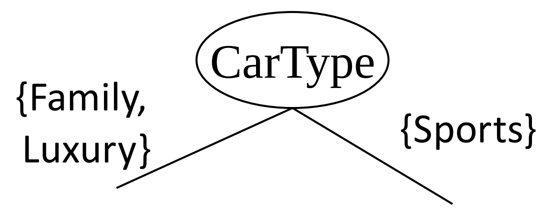
- **Multi-way split:** Use as many partitions as distinct values



- **Binary split:** Divides values into two subsets
  Need to find optimal partitioning

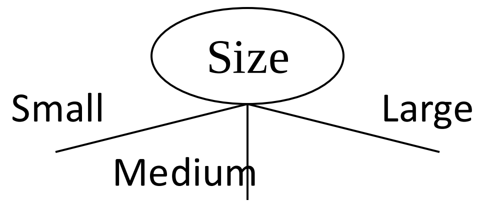


OR

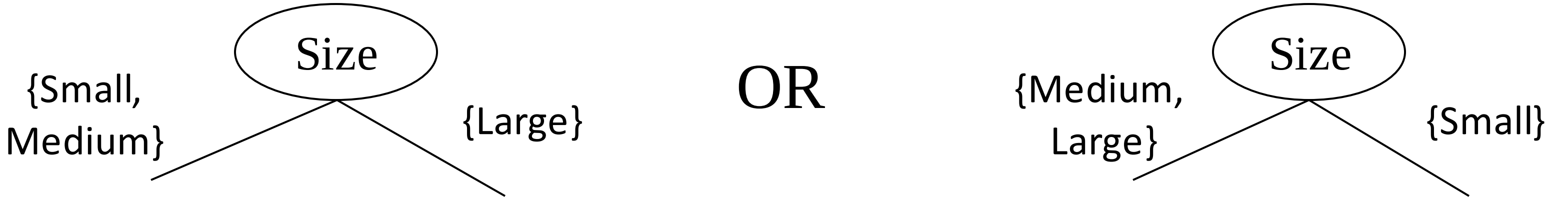


Splitting Based on Ordinal Attributes

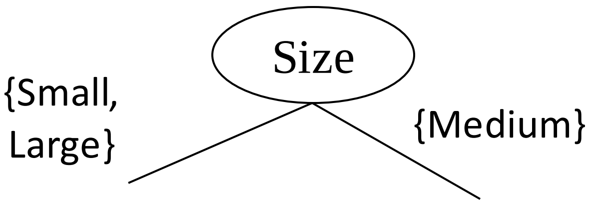
- **Multi-way split:** Use as many partitions as distinct values.



- **Binary split:** Divides values into two subsets
  Need to find optimal partitioning



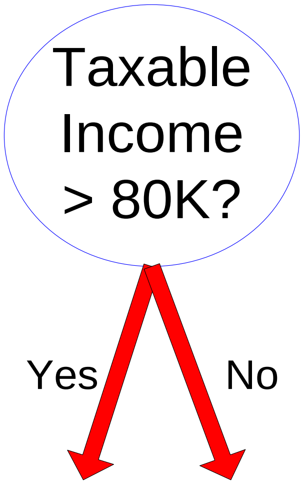
- What about this split?



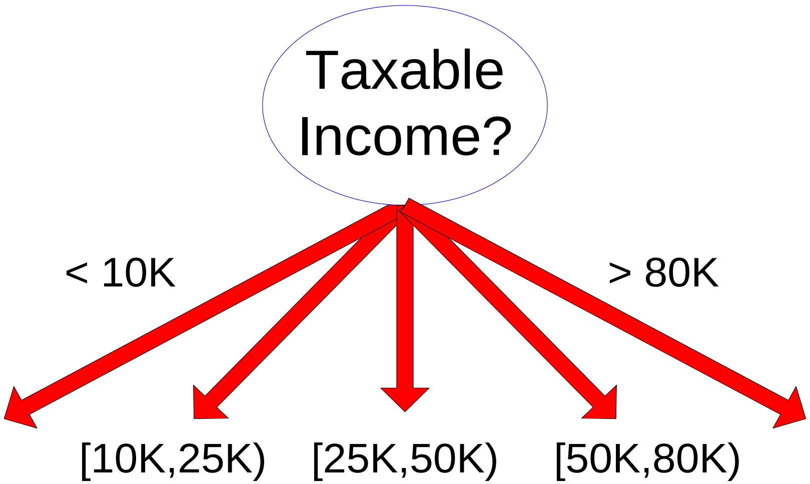
Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute
  - **Binary Decision:** $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more computation intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

Learn a Decision Tree

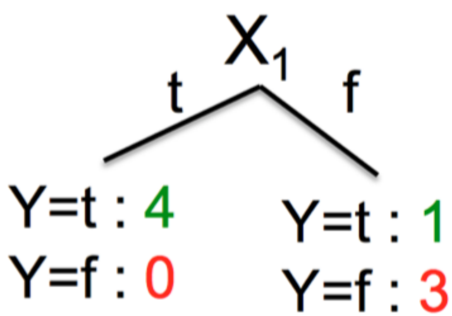
- The best tree?
  - Occam's Razor: The smallest decision tree that correctly classifies all of the training examples is best.
  - Finding the the smallest (simplest) decision tree is an NP-hard problem [if you are interested, check: Hyafil & Rivest'76].

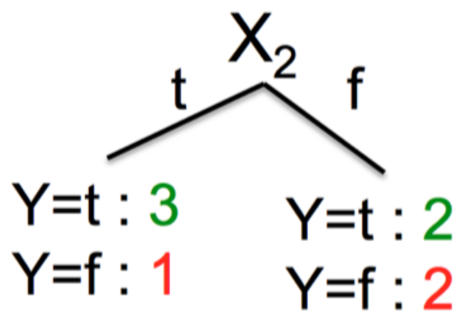
- How do we construct a useful decision tree?
- Resort to a greedy heuristic:
  - Start from an empty decision tree
  - Split on next best attribute
  - Recurse
- What is best attribute?

- We use information theory to guide us

Choosing a Good Attribute

- Which attribute is better to split on, X_1 or X_2?





X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

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

Choosing a Good Attribute

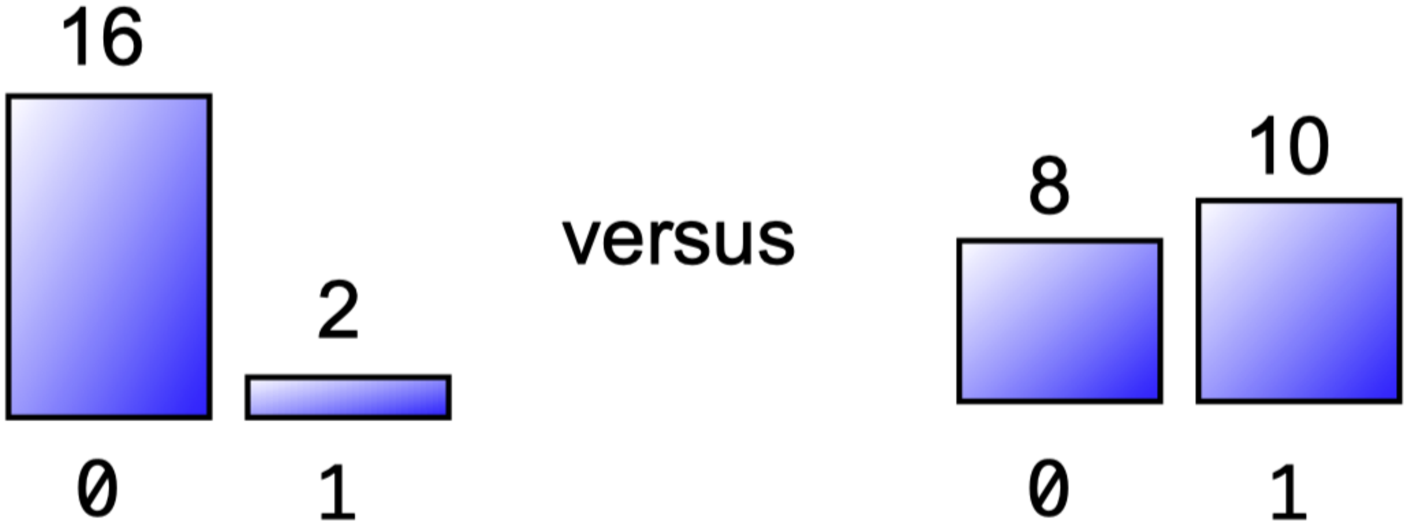
- Which attribute is better to split on, X_1 or X_2?
  - Deterministic: good (all are true or false; just one class in the leaf)
  - Uniform distribution: bad (all classes in leaf equally probable)
  - What about distributons in between?

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

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 ... ?

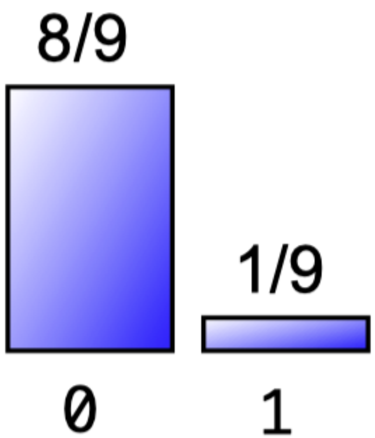
Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

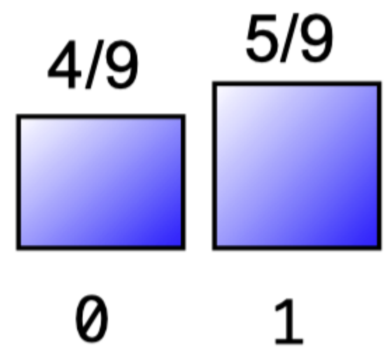


Quantifying Uncertainty

Entropy H:

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



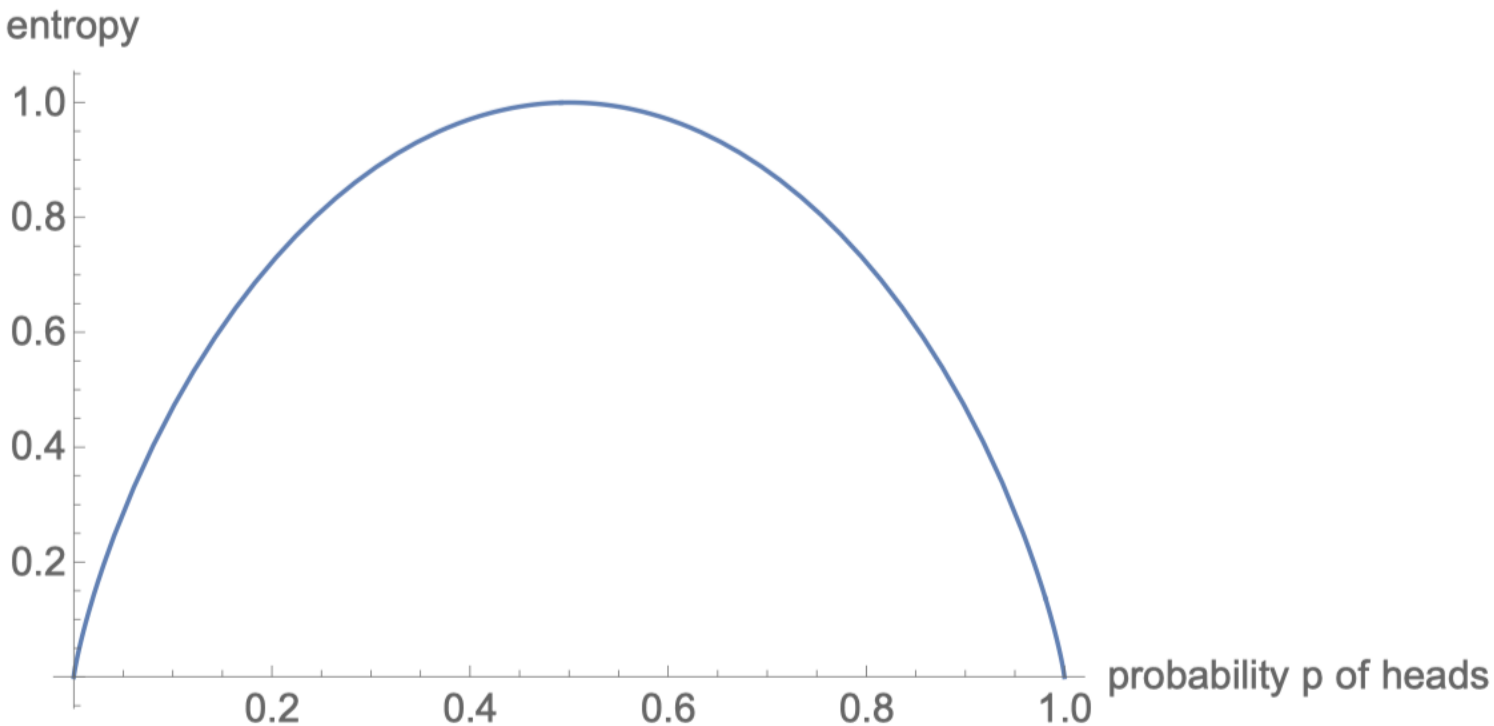


$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

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

Quantifying Uncertainty

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



Entropy

- **“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

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: $H(Y|X) \leq H(Y)$

Information Gain

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

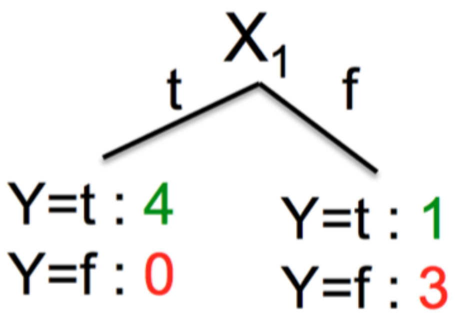
- How much information about cloudiness do we get by discovering whether it is raining?

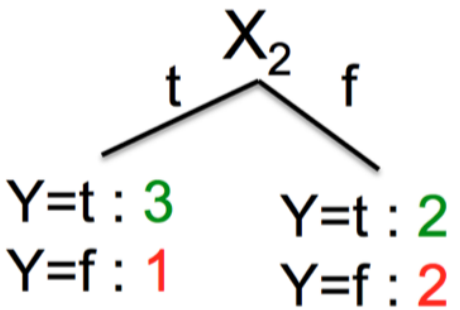
$$
\begin{aligned}
IG(Y|X) &= H(Y) - H(Y|X) \\
&\approx 0.25 \text{ bits}
\end{aligned}
$$

- 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?

Back to the Previous Sample

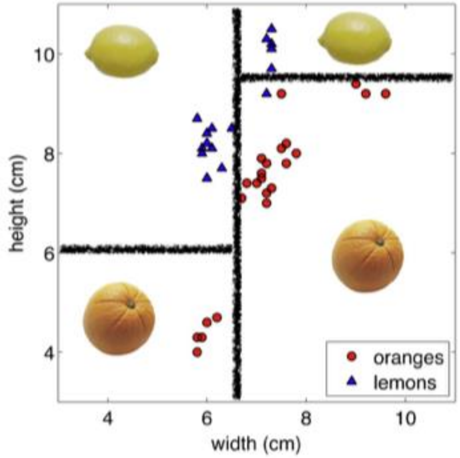
- Which attribute is better to split on, X_1 or X_2?

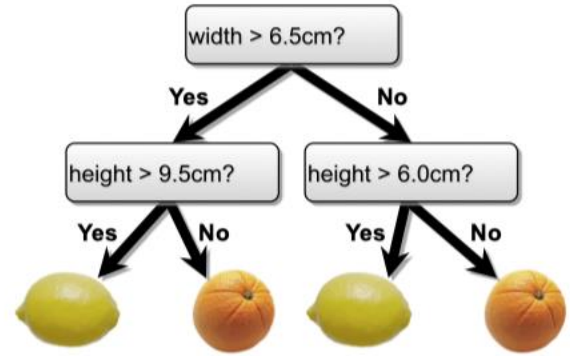




X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

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

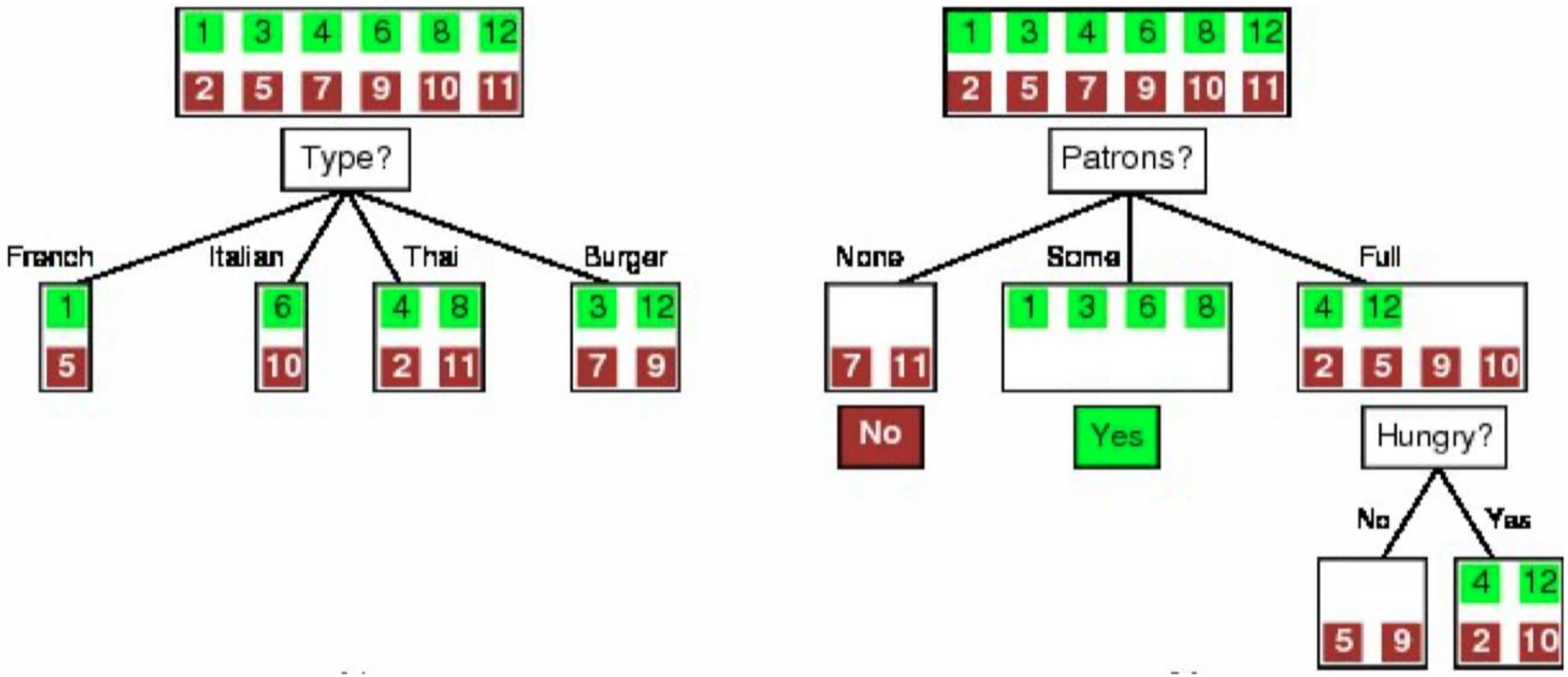
Back to Our Example

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

Attributes:

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).

Attribute Selection



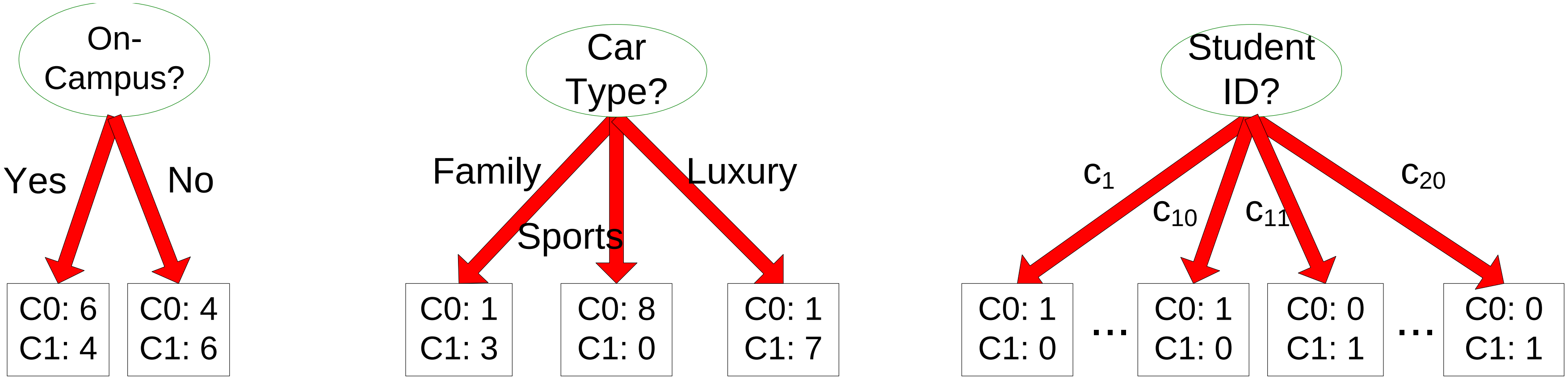
$$IG(Y) = H(Y) - H(Y|X)$$

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

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

How to determine the Best Split: Impurity

Before Splitting: 10 records of class 0,
10 records of class 1



How to determine the Best Split: Impurity

- **Greedy approach:**
  - Nodes with homogeneous class distribution are preferred
- **Need a measure of node impurity:**

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most useful information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Measure of Impurity: GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

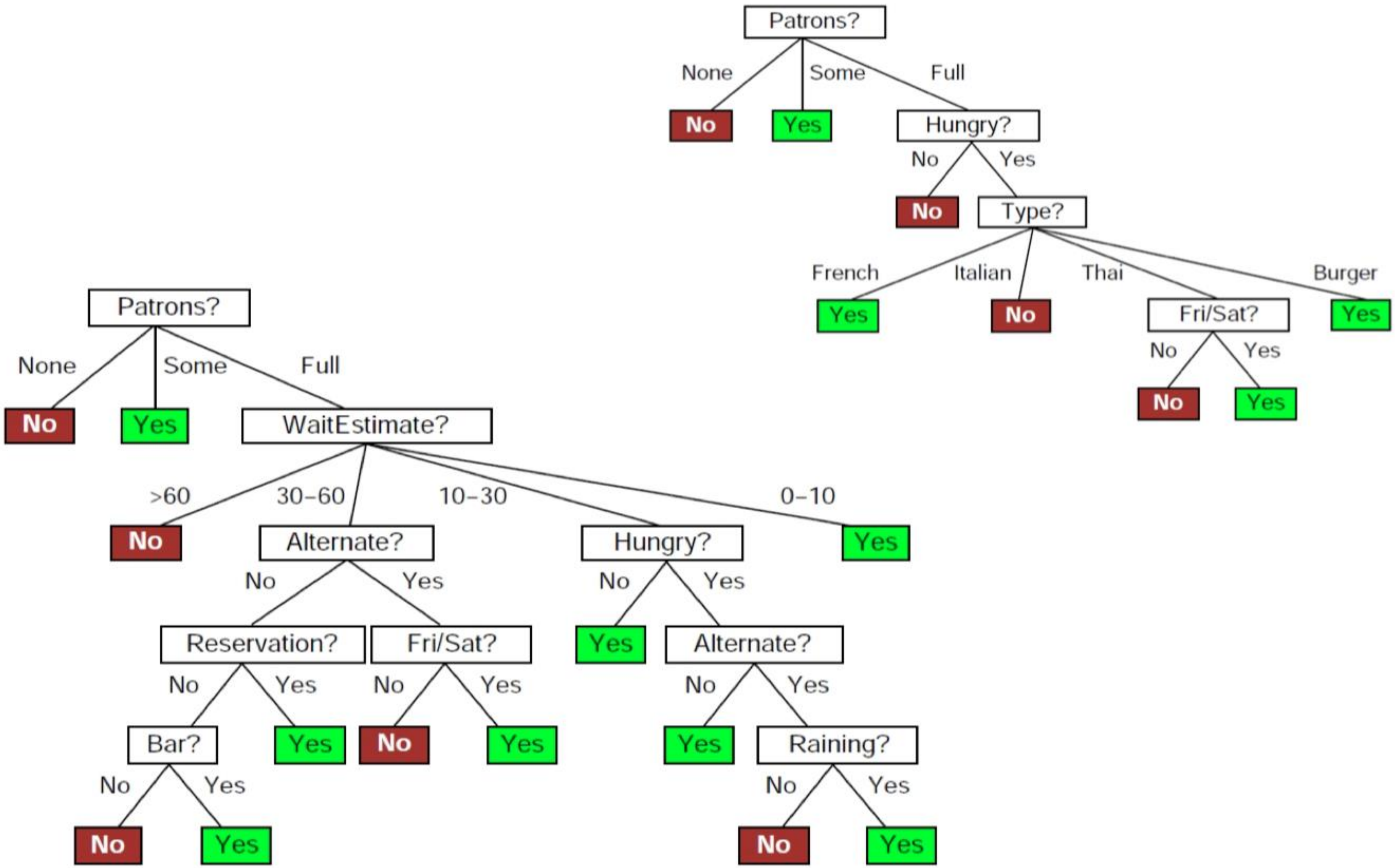
Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

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 linear
- Test complexity: non-parametric, few parameters besides (all?) training examples

Decision Trees

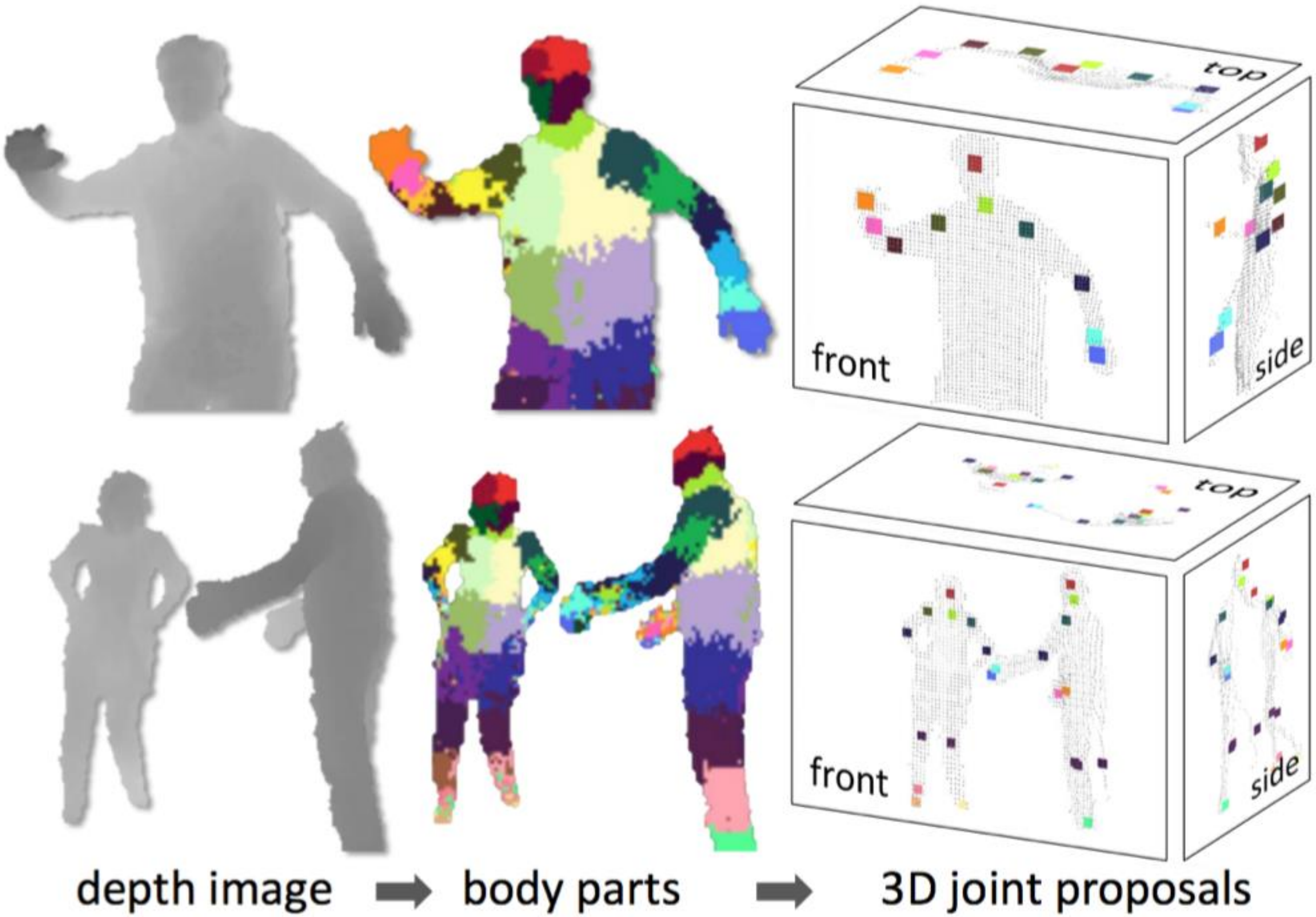
- Decision boundaries: axis-aligned, tree structured
- Test complexity: attributes and splits

Applications of Decision Trees: XBox!

- Decision trees are in XBox

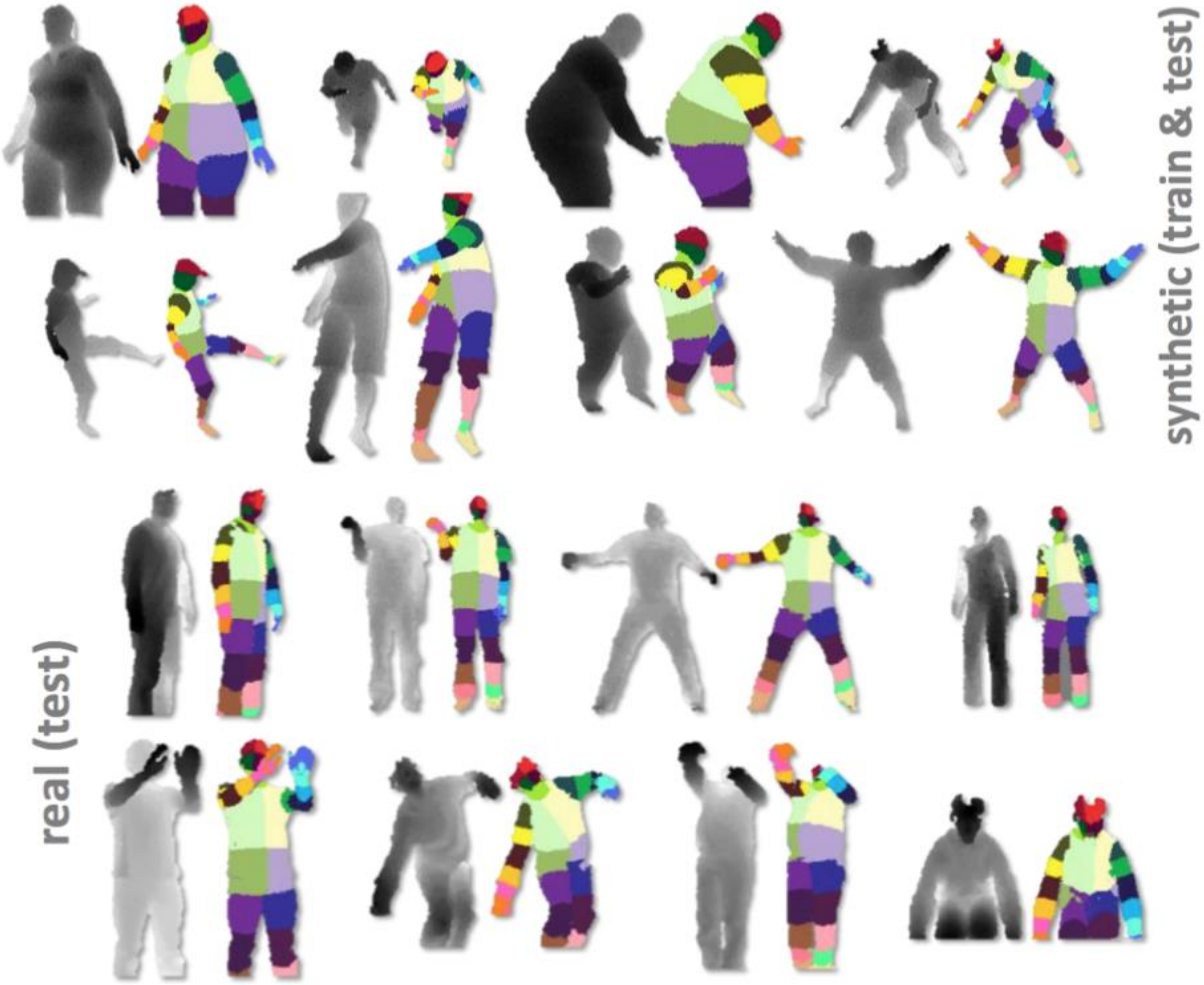
Applications of Decision Trees: XBox!

- Decision trees are in XBox: Classifying body parts



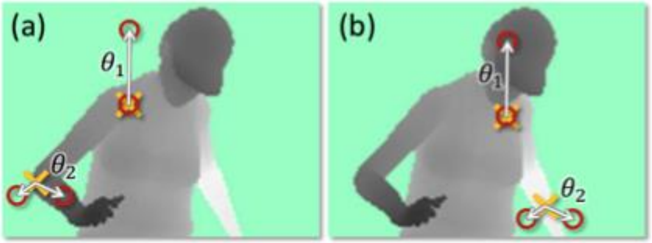
Applications of Decision Trees: XBox!

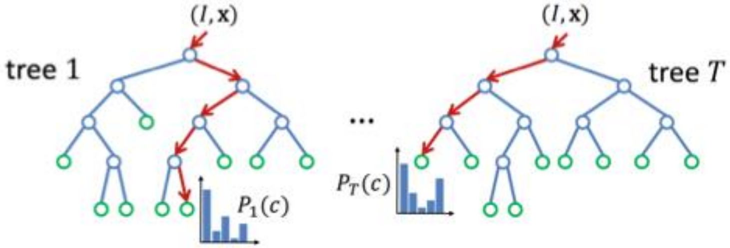
- Trained on million(s) of examples



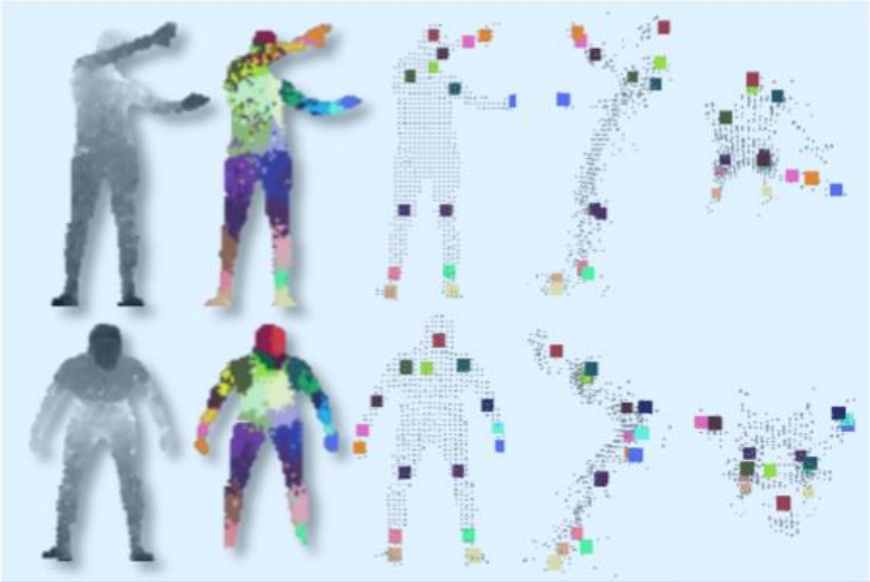
Applications of Decision Trees: XBox!

- 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:
  - 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

Questions?

UNIVERSITY OF NORTH CAROLINA CHARLOTTE