Fundamentals of Computational Intelligence

Decision Trees

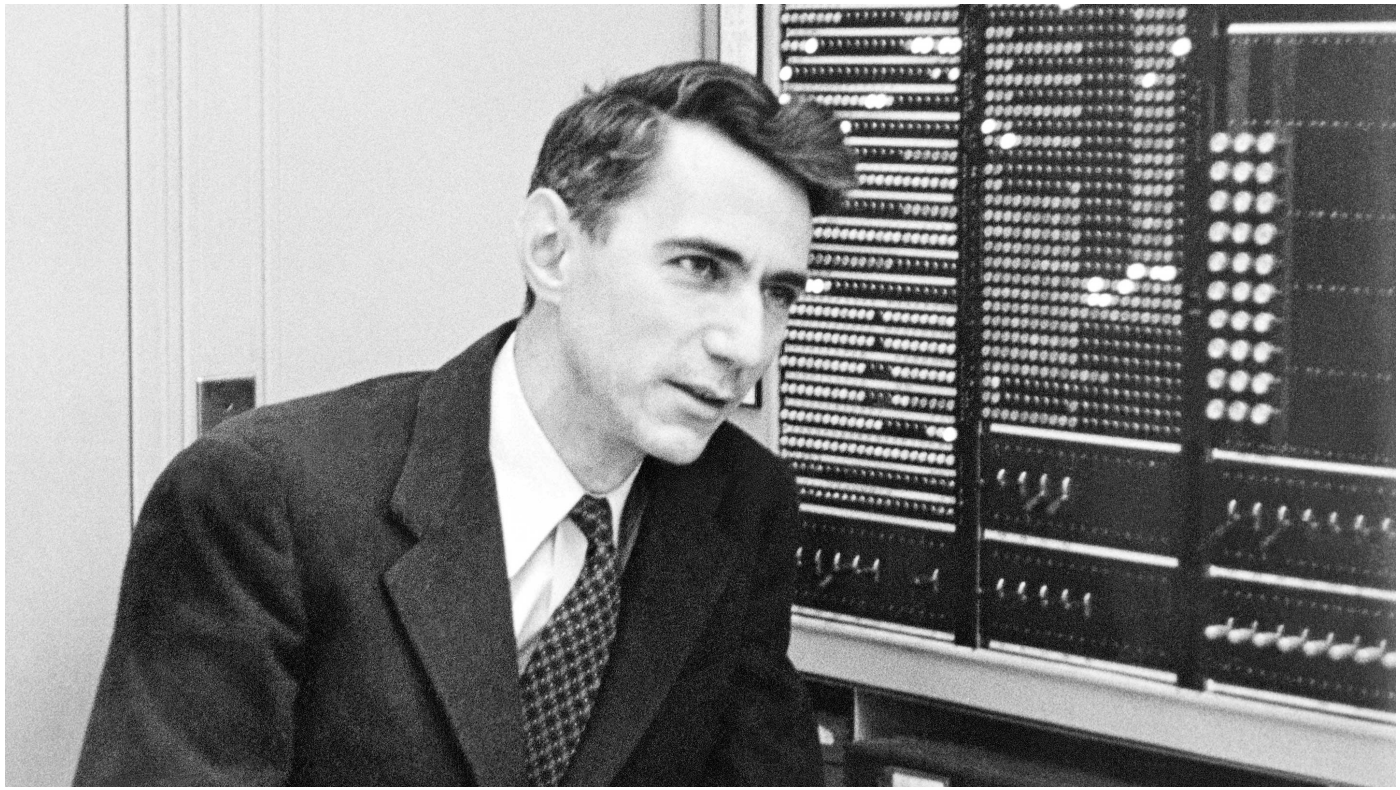
Mark Crowley

University of Waterloo
Department of Electrical and Computer Engineering

Get to Know a Scientist

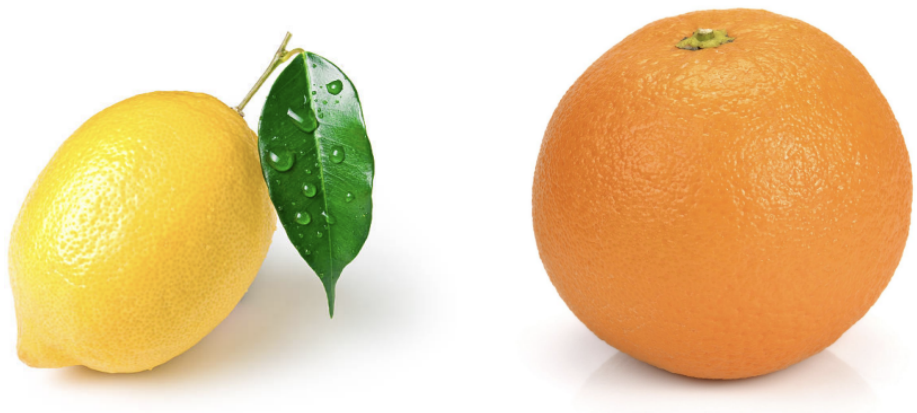
Who is this? 🤔 Answer in class.

Get to Know a Scientist



Who is this? 🤔 Answer in class.

Lemons or Oranges

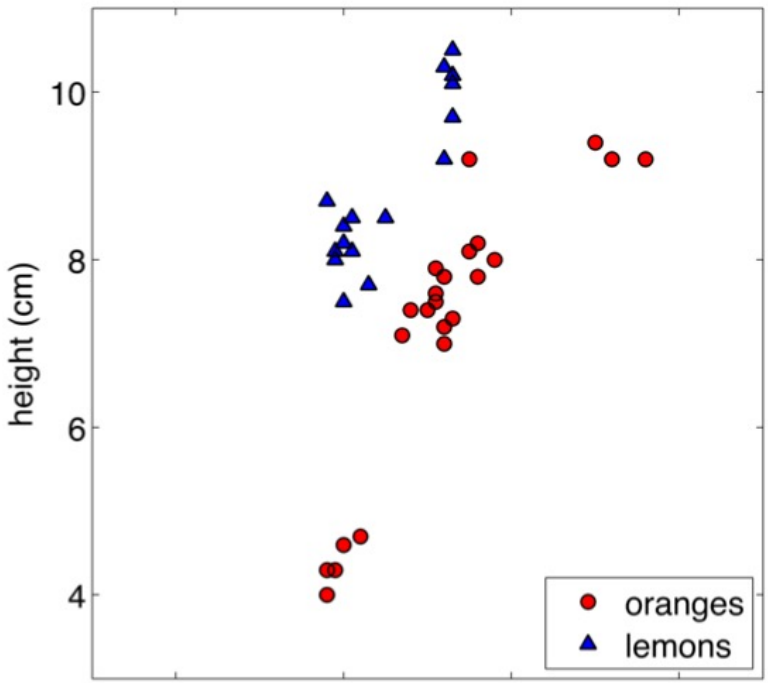


Scenario: You run a sorting facility for citrus fruits

- Features measured by sensors on conveyor belt: height & width
- For a new fruit, is it a lemon or an orange?

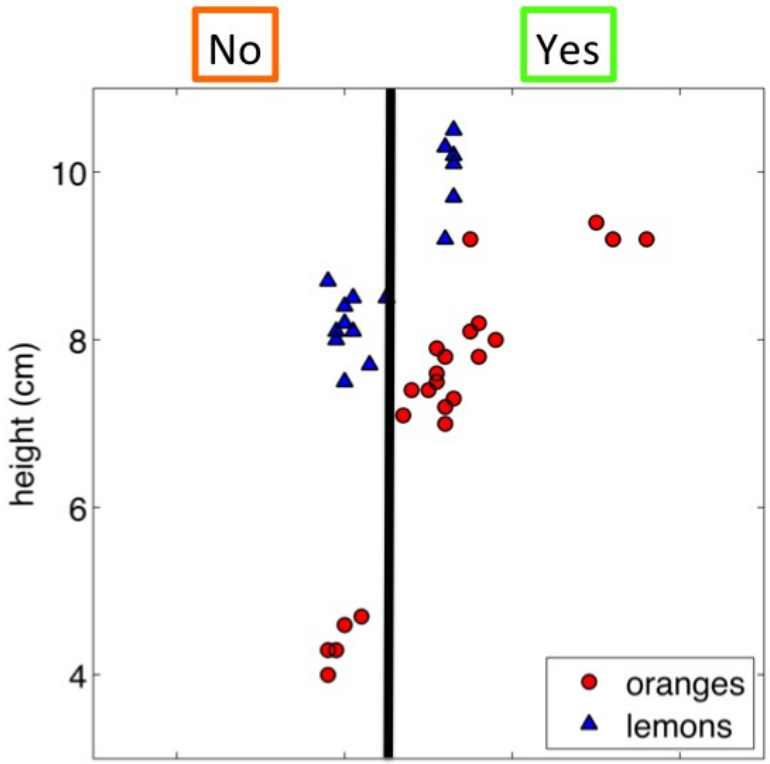
A New Classification Idea

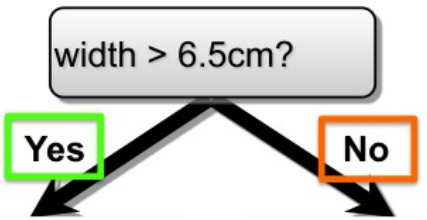
- Data on height and width can be visualized in a 2D scatter plot; How would you classify fruits? 🤔
- *Pick an attribute*, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign the prediction corresponding to most items in the leaf
- Do other branches as well



A New Classification Idea

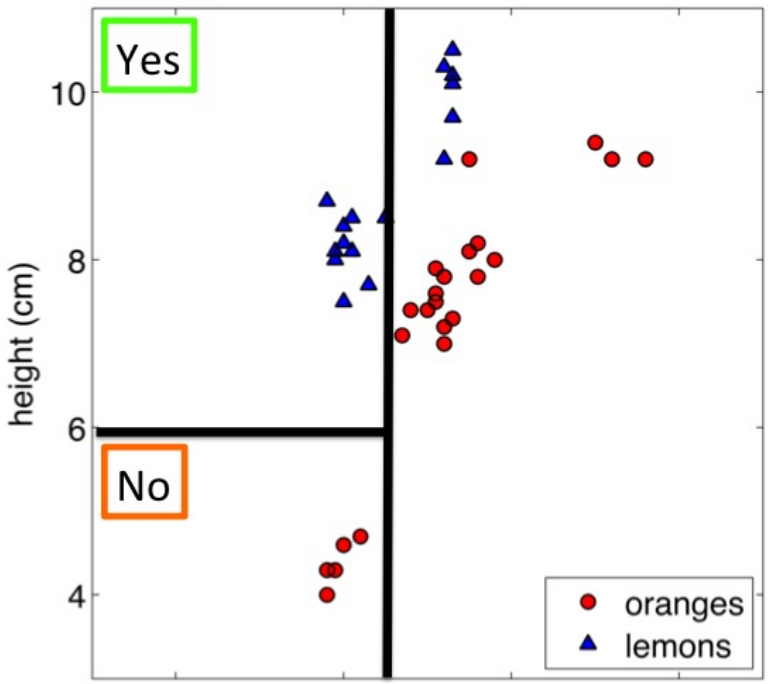
- Data on height and width can be visualized in a 2D scatter plot; How would you classify fruits? 🤔
- *Pick an attribute*, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign the prediction corresponding to most items in the leaf
- Do other branches as well

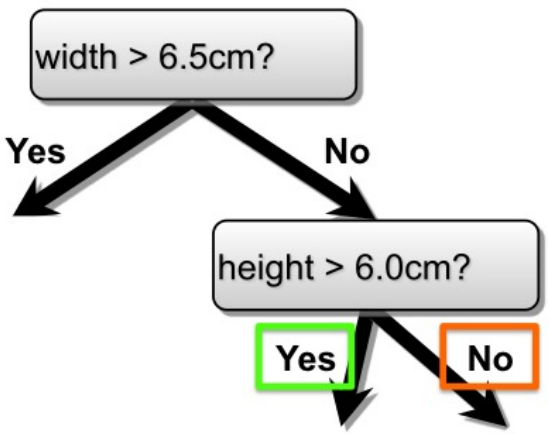




A New Classification Idea

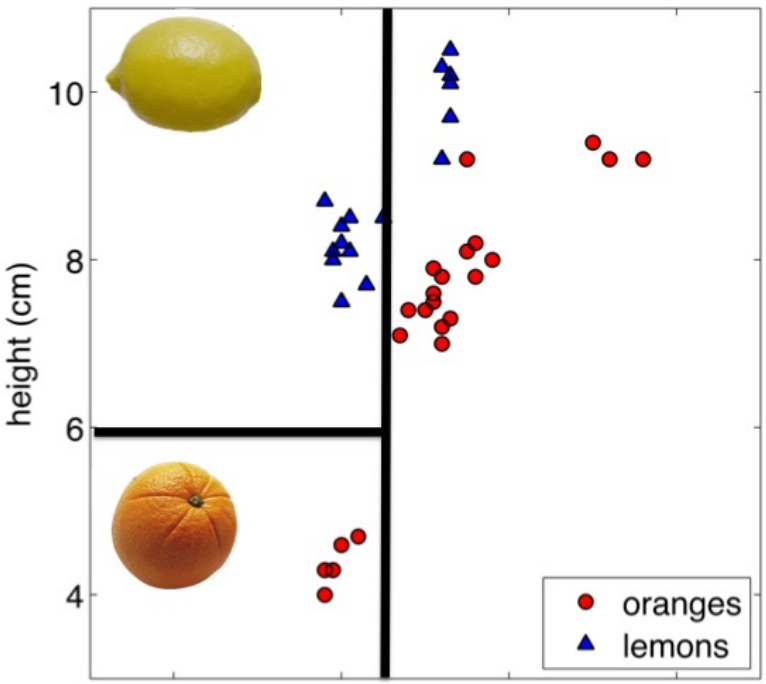
- Data on height and width can be visualized in a 2D scatter plot; How would you classify fruits? 🤔
- *Pick an attribute*, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign the prediction corresponding to most items in the leaf
- Do other branches as well

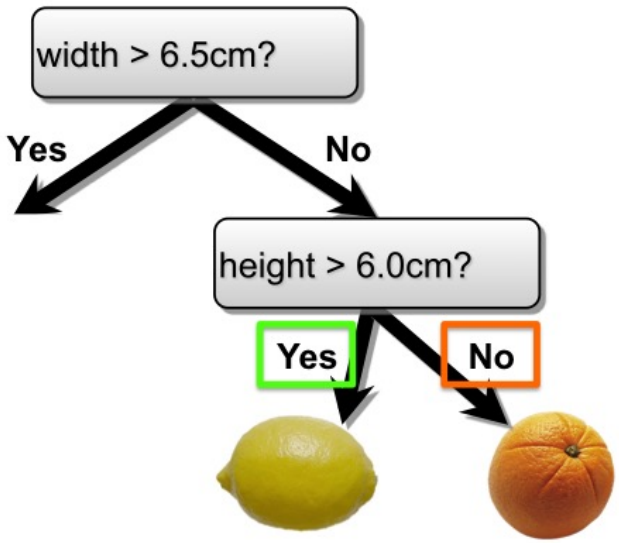




A New Classification Idea

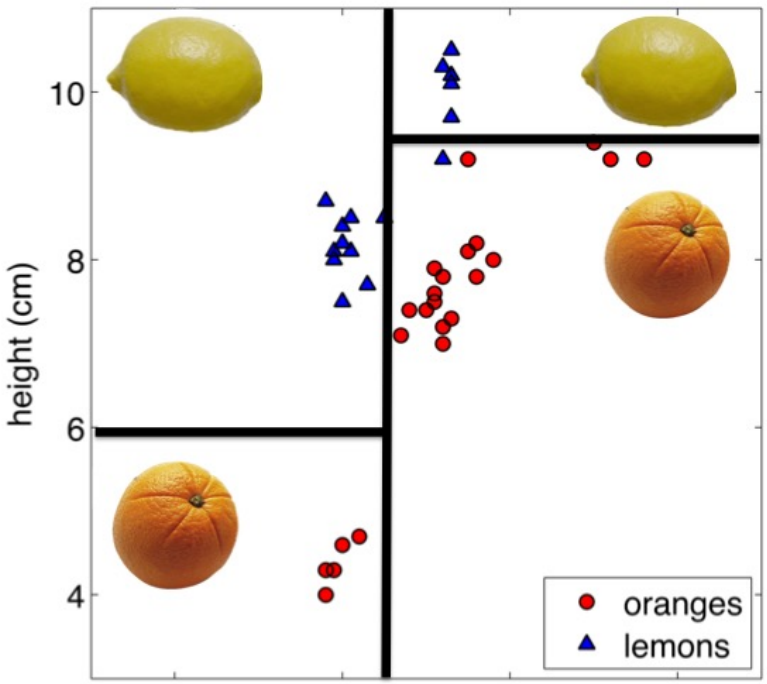
- Data on height and width can be visualized in a 2D scatter plot; How would you classify fruits? 🤔
- *Pick an attribute*, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign the prediction corresponding to most items in the leaf
- Do other branches as well

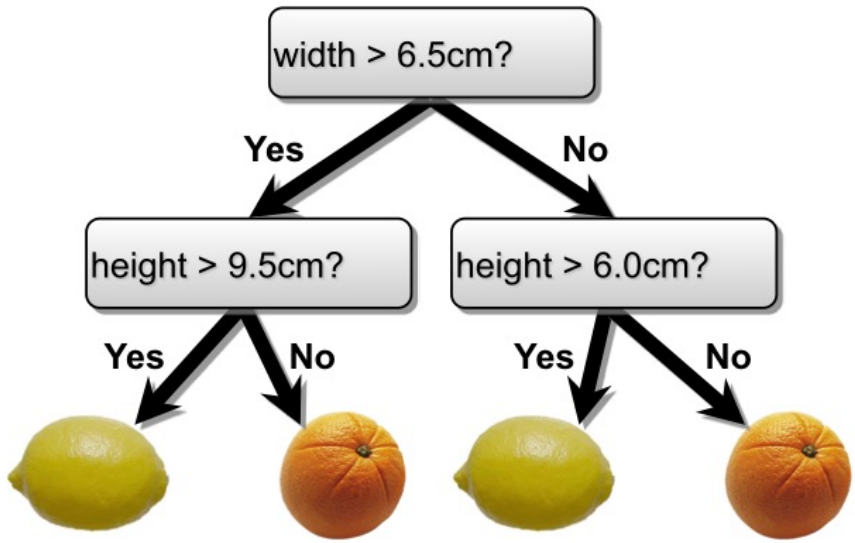




A New Classification Idea

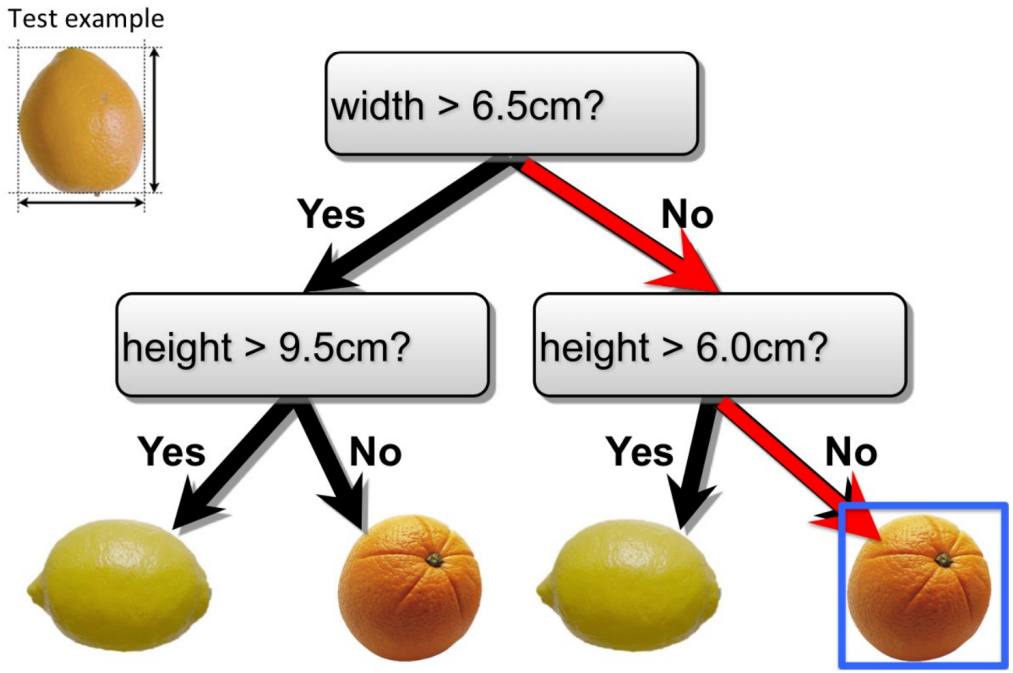
- Data on height and width can be visualized in a 2D scatter plot; How would you classify fruits? 🤔
- *Pick an attribute*, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign the prediction corresponding to most items in the leaf
- Do other branches as well





Decision Trees - Classifying a Test Instance

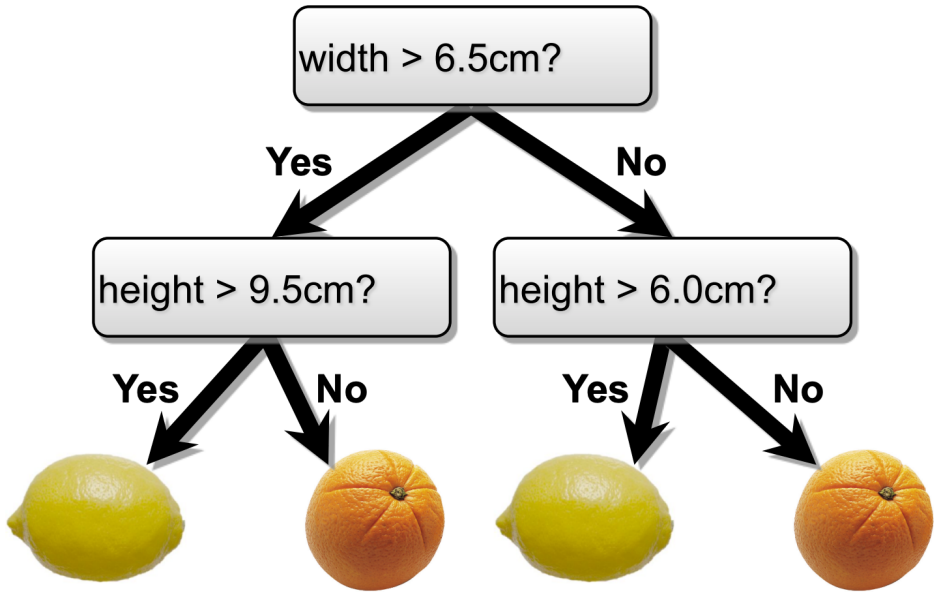
Make predictions by splitting on features according to a tree structure.



How do we classify an example using a DT? 🤔

- Start at root node.
- Test input feature and follow corresponding edge.
- Once we reach a leaf node, output class label at that node.

Characterizing Decision Trees

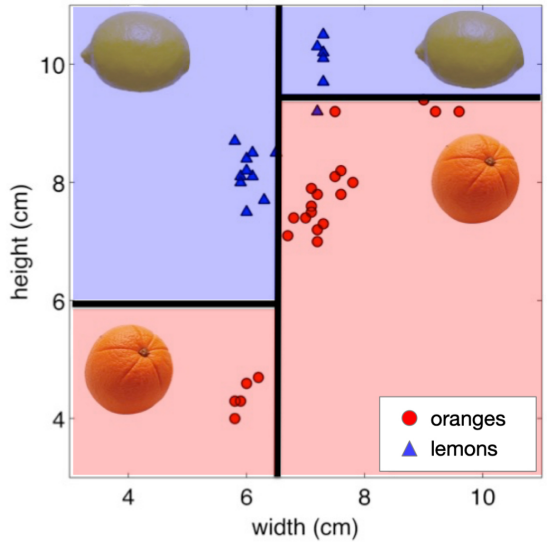


- *Internal nodes* test a *feature/attribute*.
- *Branching* is determined by the *feature/attribute value*.
- *Leaf nodes* are *outputs* (predictions).

Question: What are the params/hyperparams of this model? 🤔 Answer in class.

Characterizing Decision Trees - Classification and Regression

- Each path from root to a leaf defines a region R_m of input space
  - Let $\{(x_m^{(1)}, t_m^{(1)}), \cdots, (x_m^{(k)}, t_m^{(k)})\}$ be the training examples that fall into R_m
- Declare an output value when you get to the bottom (different in classif. vs regr.)



Regression tree:

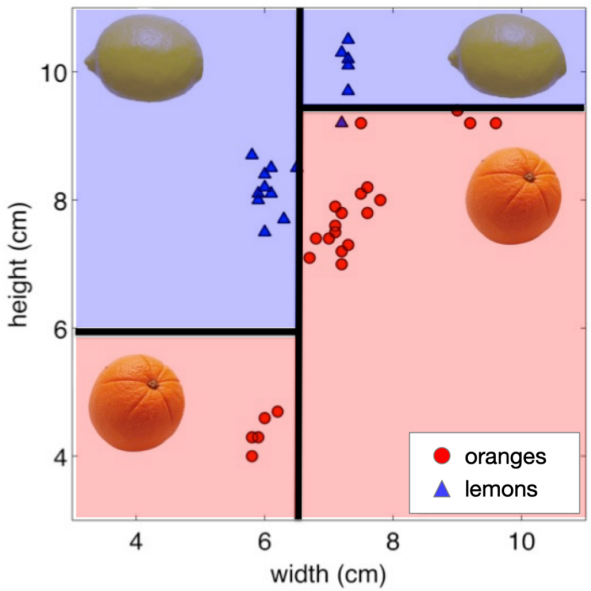
- continuous output y_m
- example? 🤔 Answer in class.
- leaf value y_m typically set to *mean*$\{(t_m^{(1)}, \cdots, t_m^{(k)})\}$

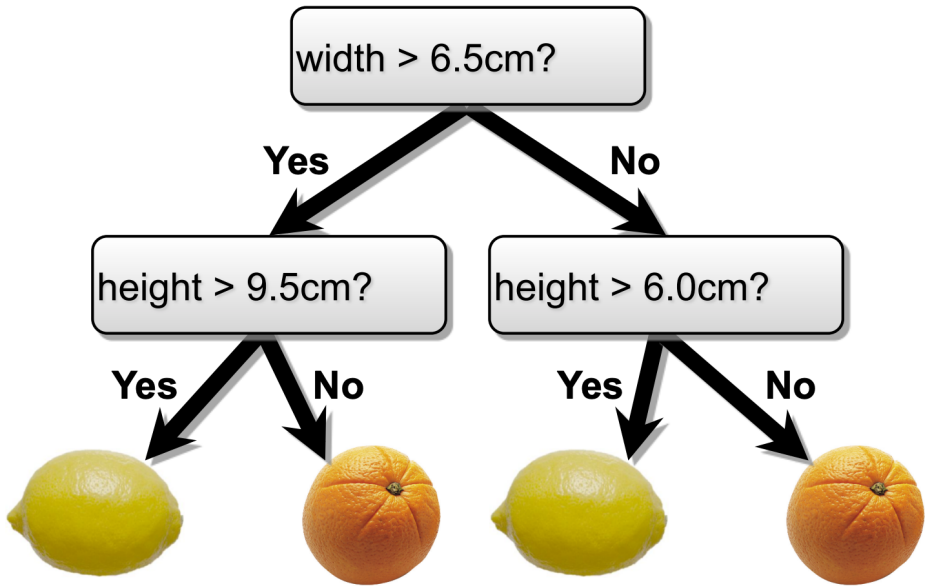
Classification tree:

- discrete output y_m
- example? 🤔 Answer in class.
- leaf value y_m typically set to *mode*$\{(t_m^{(1)}, \cdots, t_m^{(k)})\}$

Decision Trees - Continuous Features

- Split *continuous features* by checking whether that feature is greater than or less than some threshold.





- *Decision boundary* is made up of *axis-aligned planes*.
  - Why does this happen? 🤔 Answer in class.
  - How does this compare to k-NNs? 🤔 Answer in class.

Decision Trees - Discrete Features

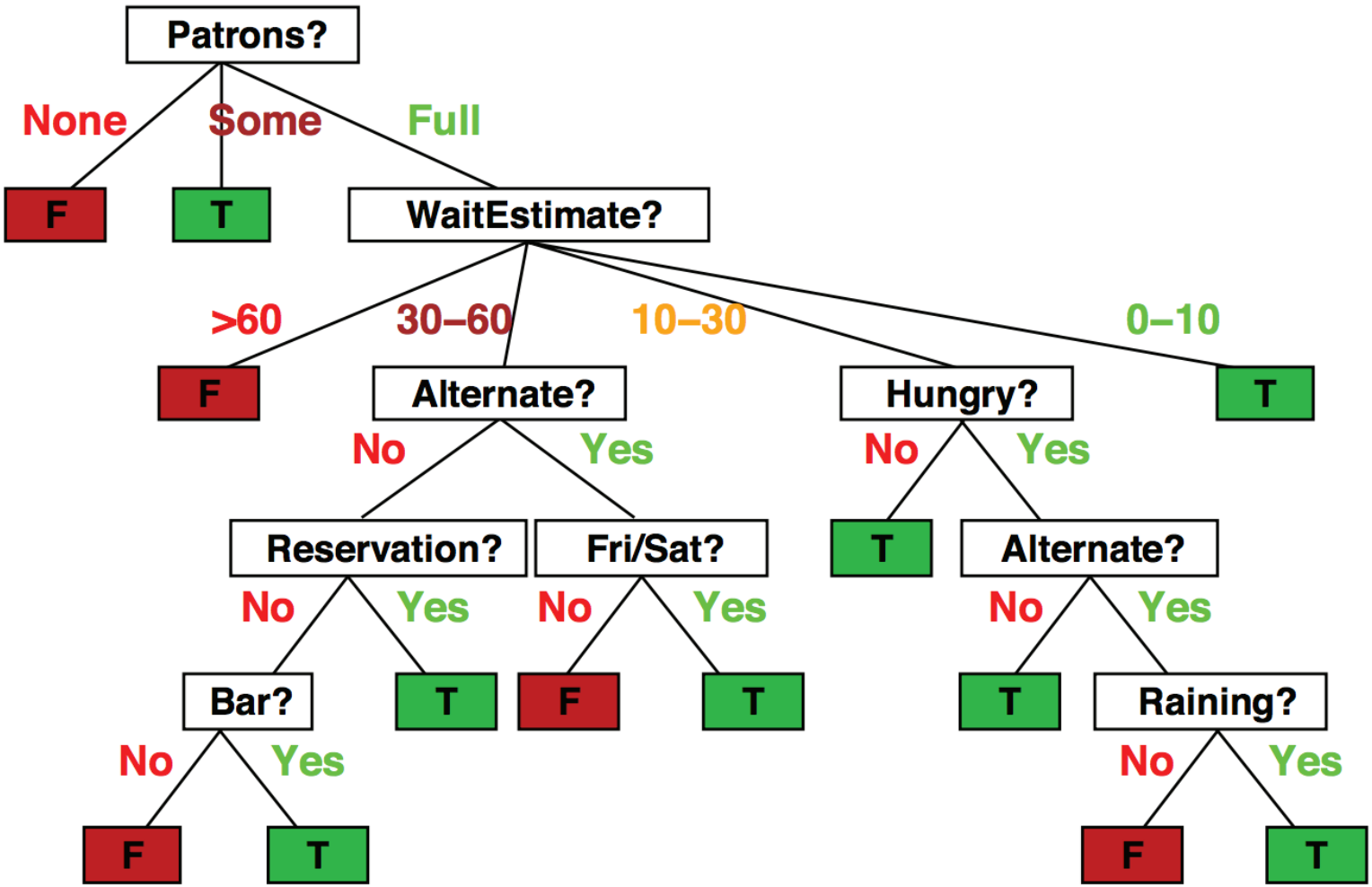
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*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).

Decision Trees - Discrete Features

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



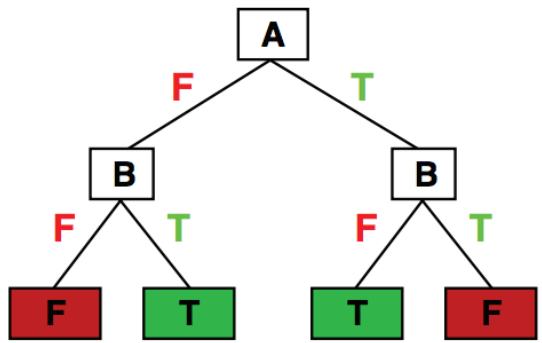
Expressiveness

Decision trees can express any function of the input attributes, i.e., can *approximate any function arbitrarily closely*, i.e., are *universal function approximators*.* Why? 🤔

- Answer in class.*

- E.g., for Boolean functions, truth table row → path to leaf:

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



- Any problems with this? 🤔 Answer in class.

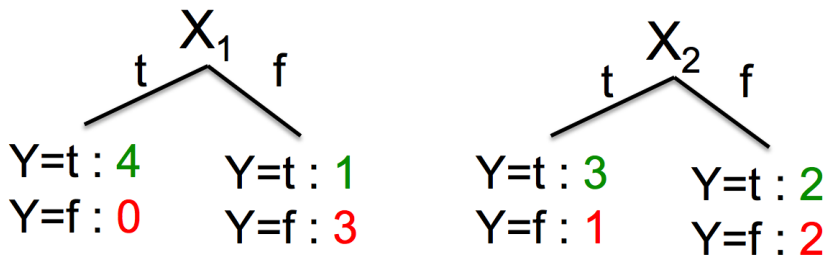
** Can classify any training set perfectly, it probably won't generalize, i.e., it overfits!*

How do we Learn a Decision Tree?

- Finding the *smallest / optimal* decision tree that correctly classifies a training set is NP complete!
  - Is too computationally expensive; may not be worth it...
  - If you are interested, check: Laurent and Rivest, 1976
- So, if not optimally, how do we construct a decision tree? 🤔 Answer in class.
  - Start with the *whole training set* and an *empty decision tree*.
  - Split on next *best* attribute.
  - *Recurse* on subpartitions.
- What is the *best* attribute?

Choosing a Good Attribute

- Which attribute is better to split on, X_1 or X_2? 🤔 Answer in class.



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

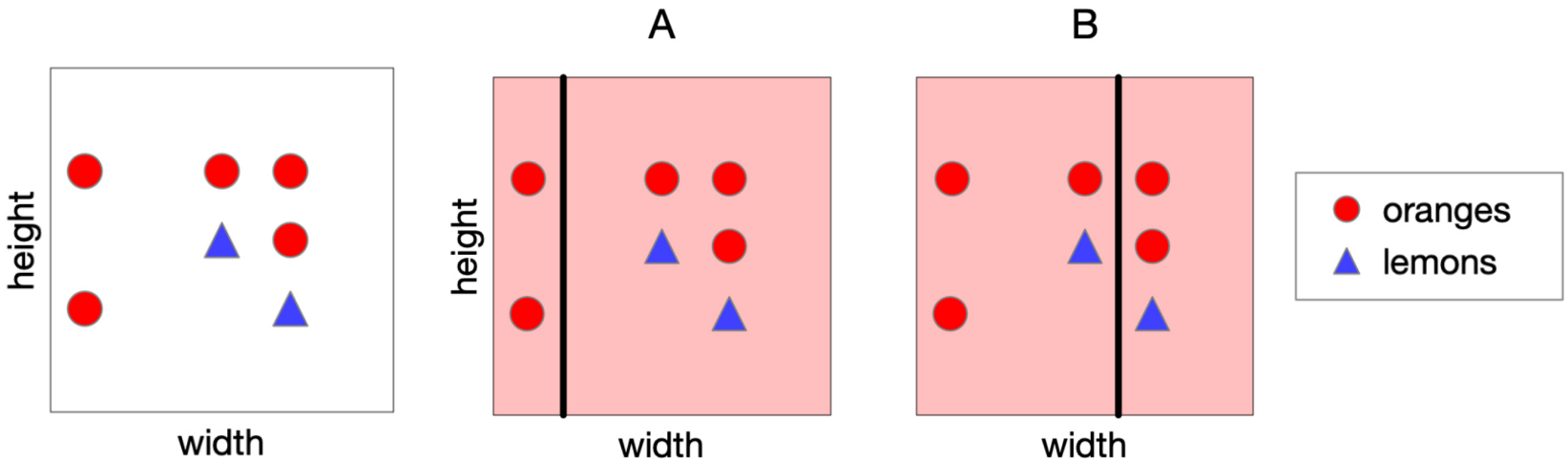
- *Deterministic decision*: good (most are true or false; just one class in the leaf)
- *Uniform distribution*: bad (most classes in leaf equally probable)

Intuition: at each (splitting) step, choose the *most informative feature* right now; Why is it greedy? 🤔 Answer in class.

Slide credit: D. Sonntag

Choosing a good split

- Consider the following data. Let's split on width. What is the best split? Vote! 🤔



- Answer in class.
- In general, *the faster we can assign a class label* or reach a leaf node, *the better*.
- Can we *quantify* this?

Choosing a good split

- How can we *quantify uncertainty* in prediction for a given leaf node?
  - If a leaf contains *examples with the same class*: good, low uncertainty
  - If a leaf contains *examples from every class*: bad, high uncertainty
- Idea: Use *counts at leaves* to define probability distributions; use a *probabilistic notion of uncertainty* to decide splits.
- A brief detour through *information theory*...

Entropy - Quantifying Uncertainty

- You may have encountered the term *entropy* quantifying ...
  - In *chemical* and *physical* systems, it is the state of chaos.
  - In *statistics*, it is a *property of a random variable*.
    - The entropy of a discrete random variable is a number that quantifies the *uncertainty inherent* in its *possible outcomes*.
- The mathematical definition of entropy (in a few slides) may seem arbitrary, but it can be motivated axiomatically.
  - Interested? Ash, 2012; Cover, 1999
- To explain entropy, consider flipping two different biased coins...

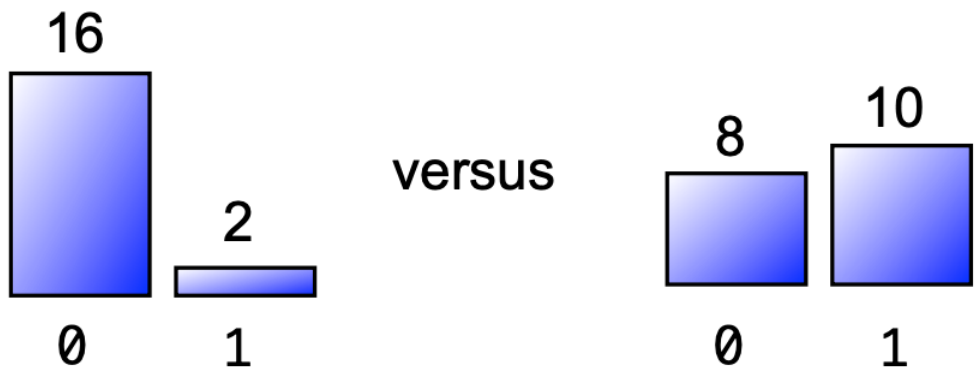
Entropy - Flipping Two Biased Coins

- Each coin is a binary random variable; outcomes Heads (0) or Tails (1)

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 ... ?

- Which is more "biased"/"loaded"? 🤔 Answer in class.

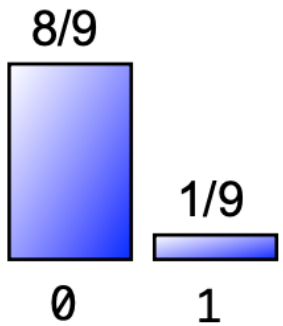


Quantifying Uncertainty

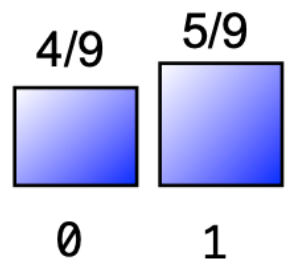
- The entropy of a *loaded coin* with probability p of heads is given by

$$\mathrm{H}(X) := -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x),$$

$$p \overset{\text{binary}}{\Longrightarrow} \mathrm{H}(X) = -p \log_2(p) - (1-p) \log_2(1-p)$$



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

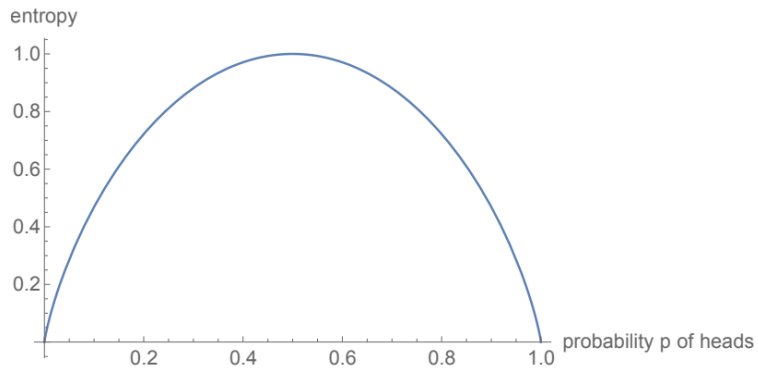


$$-\tfrac{4}{9} \log_2 \tfrac{4}{9} - \tfrac{5}{9} \log_2 \tfrac{5}{9} \approx 0.99$$

- Notice: the coin whose outcomes are *more certain* has a *lower entropy*.
- In the extreme case $p = 0$ or $p = 1$, we would be certain of the outcome before observing a new flip.
  - What is H in these cases? 🤔
  - Answer in class.

Quantifying Uncertainty

- Can also think of *entropy* as the *expected information content* of a *random draw* from a probability distribution



Fair coin: *most difficult to predict* the outcome of the next flip. So the result of next flip gives *1 full bit of information*.

- Claude Shannon showed: you cannot store the outcome of a random draw using fewer expected bits than the entropy without losing information.
- What are the units of entropy? 🤔 Answer in class.

Quantifying Uncertainty

- The entropy of a *loaded coin* with probability p of heads is given by

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

- *High Entropy*
  - Distribution: uniform over many outcomes
  - Histogram: flat
  - Sampled values: less predictable
- *Low Entropy*
  - Distribution: concentrated on a few outcomes
  - Histogram: concentrated in a few areas
  - Sampled values: more predictable

Entropy of Lemons vs Oranges

- Suppose we observe partial information X about a random variable Y
  - X: *is the width* > 6.5*cm?*
  - Y: *lemon vs orange?*
- What is the expected amount of information that will be conveyed about Y by observing X?
  - Equivalently, What is the *expected reduction in our uncertainty* about Y after observing X?

Joint Entropy

- The *joint entropy* measures *how much uncertainty there is* in the two random variables X and Y taken together.
- Example: $X = \{\text{Raining}, \neg\ \text{Raining}\}$, $Y = \{\text{Cloudy}, \neg\ \text{Cloudy}\}$

	Cloudy	¬ Cloudy
Raining	24/100	1/100
¬ Raining	25/100	50/100

This means:

- if Raining, it is Cloudy, *almost for sure*
- if ¬ Raining, it is slightly more likely to be ¬ Cloudy, *but not sure*

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

From Bayes Rule: $p(y|x) = \frac{p(x,y)}{p(x)}$ *and* $p(x) = \sum_y p(x,y)$

Joint Entropy

Practice at home; Which system has lower/higher entropy? 🤔

	Cloudy	¬ Cloudy
Raining	95/100	1/100
¬ Raining	3/100	1/100

	Cloudy	¬ Cloudy
Raining	25/100	25/100
¬ Raining	25/100	25/100

Show that the max value of $\mathrm{H}(X, Y)$ is 2. 🤔

Conditional Entropy - $\mathrm{H}(Y|X = \text{Raining})$

- The *conditional entropy* is a measure of *how much uncertainty remains* about the random variable Y when we know the value of X.
- Example: $X = \{\text{Raining}, \neg\ \text{Raining}\}$, $Y = \{\text{Cloudy}, \neg\ \text{Cloudy}\}$

	Cloudy	$\neg$ Cloudy
Raining	24/100	1/100
$\neg$ Raining	25/100	50/100

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

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

Conditional Entropy - $\mathrm{H}(Y|X = \neg\text{Raining})$

- Example: $X = \{\text{Raining}, \neg\ \text{Raining}\}$, $Y = \{\text{Cloudy}, \neg\ \text{Cloudy}\}$

	Cloudy	$\neg$ Cloudy
Raining	24/100	1/100
$\neg$ Raining	25/100	50/100

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

$$\begin{aligned}\mathrm{H}(Y|X = \neg\text{Raining}) &= \sum_{y\in\mathcal{Y}} p(y|x)\log_2 p(y|x) \\ &= -\frac{25}{75}\log_2\frac{25}{75} - \frac{50}{75}\log_2\frac{50}{75} \\ &\approx 0.92 \text{ bits}\end{aligned}$$

Conditional Entropy

- Example: $X = \{\text{Raining}, \neg\ \text{Raining}\}$, $Y = \{\text{Cloudy}, \neg\ \text{Cloudy}\}$

	Cloudy	¬ Cloudy
Raining	24/100	1/100
¬ Raining	25/100	50/100

What is the entropy of cloudiness Y...

- given that it is Raining: $\mathrm{H}(Y|X = \text{Raining}) \approx 0.24$ bits
- given that it is ¬ Raining: $\mathrm{H}(Y|X = \neg\text{Raining}) \approx 0.92$ bits

What does this mean? 🤔 Answer in class.

low (high) bits = low (high) uncertainty = low (high) info gain

What is the overall *entropy of Cloudiness*, *given the knowledge* of *whether or not it is Raining*, *despite not knowing in advance* what it will be? 🤔
Answer in class.

Expected Entropy

$$\begin{aligned}
\mathrm{H}(Y|X) &= \mathbb{E}[\mathrm{H}[Y|X]] \\
&= \sum_{x \in \mathcal{X}} p(x)\mathrm{H}(Y|X) \\
&= p(\text{Raining}) * \mathrm{H}(Y|X = \text{Raining}) \\
&\quad + p(\neg\text{Raining}) * \mathrm{H}(Y|X = \neg\text{Raining}) \\
&= \frac{(24+1)}{100} * 0.24 + \frac{(25+50)}{100} * .92 \\
&\approx 0.75\text{bits}
\end{aligned}$$

Practice at home:

$$\begin{aligned}
\mathrm{H}(Y|X) &= \mathbb{E}[\mathrm{H}[Y|X]] \\
&= \sum_{x \in \mathcal{X}} p(x)\mathrm{H}(Y|X) \\
&= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log_2 p(y|x)
\end{aligned}$$

Information Gain

Some useful properties:

- Entropy, H, is *always non-negative*.
- *Chain rule*: $\mathrm{H}(X, Y) = \mathrm{H}(X|Y) + \mathrm{H}(Y) = \mathrm{H}(Y|X) + \mathrm{H}(X)$.
- If X and Y *independent*, then X does not affect our uncertainty about Y : $\mathrm{H}(Y|X) = \mathrm{H}(Y)$.
- By knowing X, we *can only decrease uncertainty* about Y: $\mathrm{H}(Y|X) \leq \mathrm{H}(Y)$.
- By knowing Y, *we are perfectly certain* of Y certain (duh!): $\mathrm{H}(Y|Y) = 0$.

Information Gain

- How much *more certain* will I be of cloudiness (Y) *if I were to know whether* or *not* it's Raining? 🤔
- Answer in class.
- *Information Gain*: $\mathrm{IG}(Y|X)$ in Y due to X:*

$$\mathrm{IG}(Y|X) := \mathrm{H}(Y) - \mathrm{H}(Y|X)$$

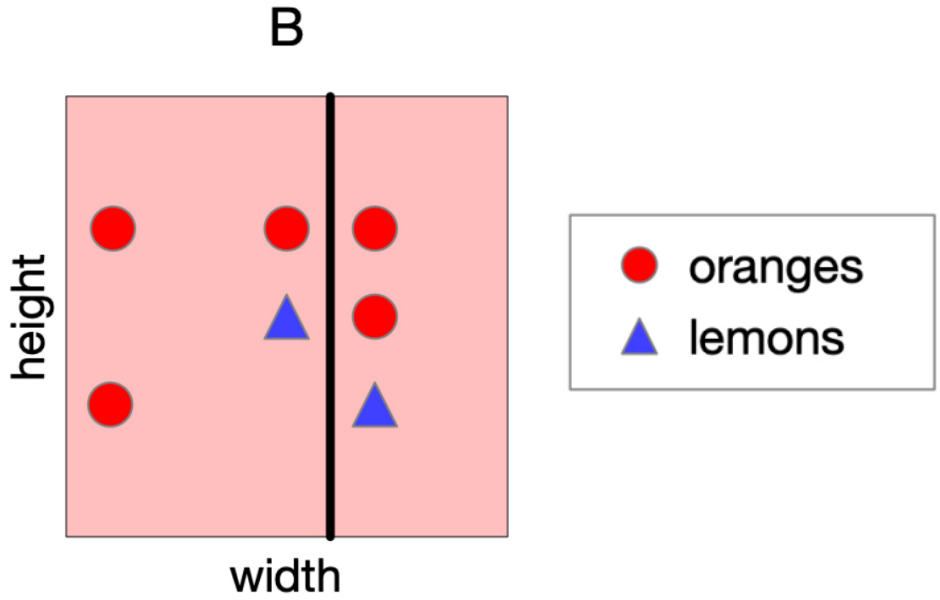
  - Remember the latter is strictly less than or equal to the former, i.e., $\mathrm{H}(Y|X) \leq \mathrm{H}(Y)$, cannot lose information!
  - If X is *completely uninformative* about Y: $\mathrm{IG}(Y|X) = 0$
  - If X is *completely informative* about Y: $\mathrm{IG}(Y|X) = \mathrm{H}(Y)$ because $\mathrm{H}(Y|X) = \mathrm{H}(Y|Y) = 0$

** Also known as the mutual information of Y and X, i.e., the amount of info obtained about one random variable through observing the other.*

Back to Trees 🌲

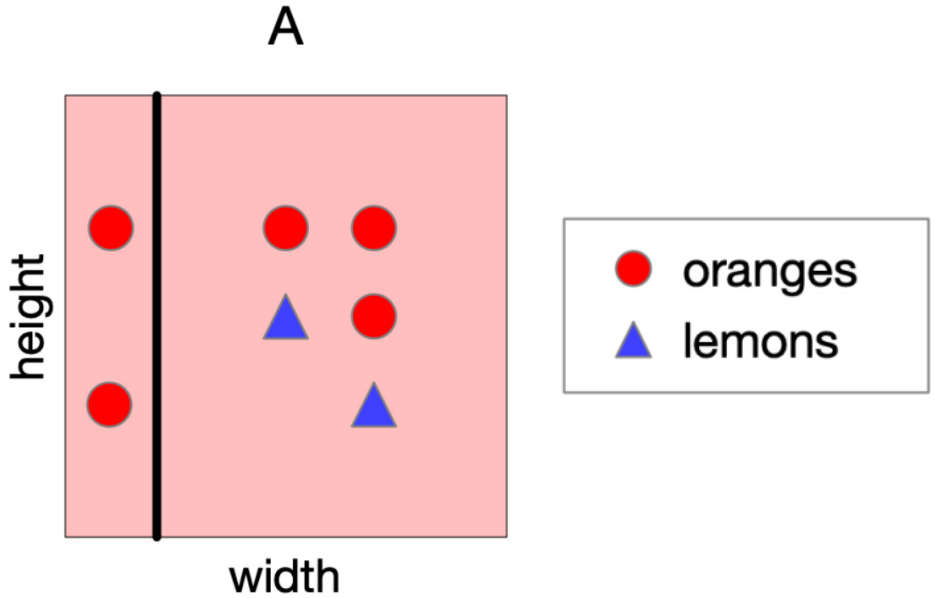
- When learning a tree, we would like to *split* on attributes that *reduce the uncertainty* in the target variable (Y) *the most*.
- In other words, we seek X_i *that maximally informs us about* Y.
- *Information Gain (in* Y *due to* X_i*)* tells us how much information we shall gain about the target Y if we are given knowledge of (i.e., if we split on) X_i (literally in its name!).

Information Gain from Split B



- Targets in the datasets: 2 lemons and 5 oranges $\implies$ $Y : (\frac{2}{7}, \frac{5}{7})$
- *Before-split* (on width) *entropy*: $\mathrm{H}(Y) = -\frac{2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7} \approx 0.863$
- *After-split* (on width) *conditional entropy*:
  - left: 1 lemons and 3 oranges $\implies$ $Y : (\frac{1}{4}, \frac{3}{4})$ $\implies$ $\mathrm{H}(Y|\text{left}) \approx 0.81$
  - right: 1 lemons and 2 oranges $\implies$ $Y : (\frac{1}{3}, \frac{2}{3})$ $\implies$ $\mathrm{H}(Y|\text{right}) \approx 0.92$
  - together: $\mathrm{H}(Y|\text{split}) = \frac{4}{7}\mathrm{H}(Y|\text{left}) + \frac{3}{7}\mathrm{H}(Y|\text{right}) \approx 0.857$
- What is the information gain of split B?
  $\mathrm{IG}(\text{split}) = \mathrm{H}(Y) - \mathrm{H}(Y|\text{split}) \approx 0.006$

Information Gain from Split A



- Targets in the datasets: 2 lemons and 5 oranges $\implies$ $Y : (\frac{2}{7}, \frac{5}{7})$
- *Before-split* (on width) *entropy*: $\mathrm{H}(Y) = -\frac{2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7} \approx 0.863$
- *After-split* (on width) *conditional entropy*:
  - left: $\mathrm{H}(Y|\text{left}) \approx$?? 🤔
  - right: $\mathrm{H}(Y|\text{right}) \approx$?? 🤔
  - together: $\mathrm{H}(Y|\text{split}) = \frac{2}{7}\mathrm{H}(Y|\text{left}) + \frac{5}{7}\mathrm{H}(Y|\text{right}) \approx$?? 🤔
- What is the information gain of split A?
  $\mathrm{IG}(\text{split}) = \mathrm{H}(Y) - \mathrm{H}(Y|\text{split}) \approx 0.17$!! What's the take-away? 🤔

Decision Trees Construction

- At each level, one must choose:
  1. Which feature to split.
  2. Possibly where to split it.
- Choose them based on how much information we would gain from the decision! (choose feature that gives the highest gain)

Decision Trees Construction

- *Simple*, *greedy*, *recursive* approach, *builds up tree node-by-node*
  1. pick a feature to split at a non-terminal node (i.e., *feature selection*)
  2. split examples into groups based on feature value (i.e., *threshold selection*)
  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
- Terminates when all leaves contain only examples in the same class or are empty.

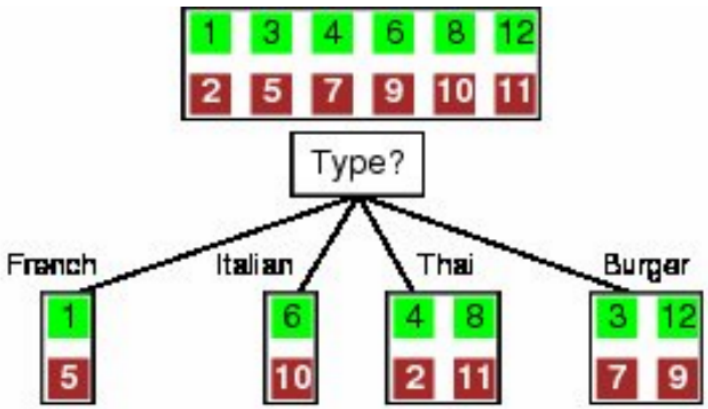
Back to the example

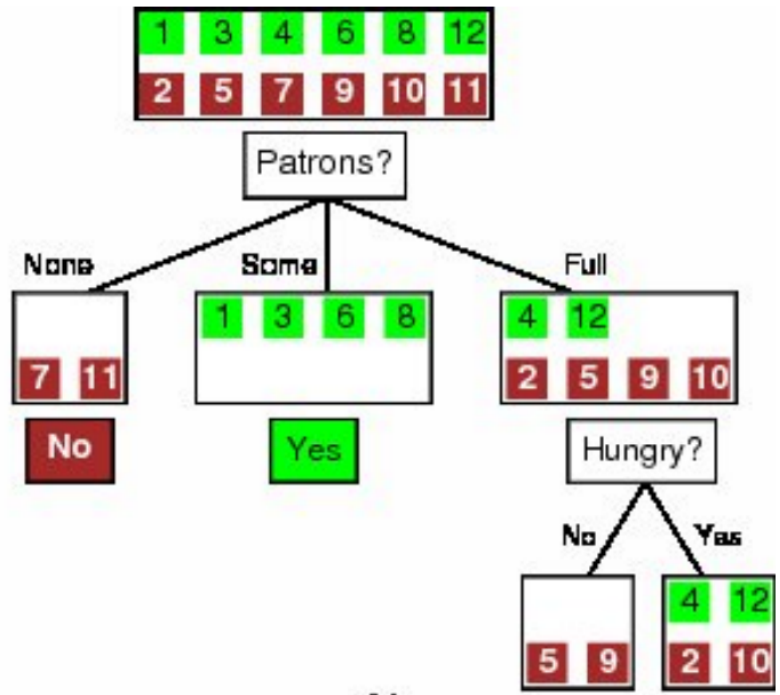
- Which *discrete features* should we split on first? "Type" or "Patrons"?
- Is it immediately obvious to anyone? 🤔

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*	$y_{12} = Yes$

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).

Feature Selection



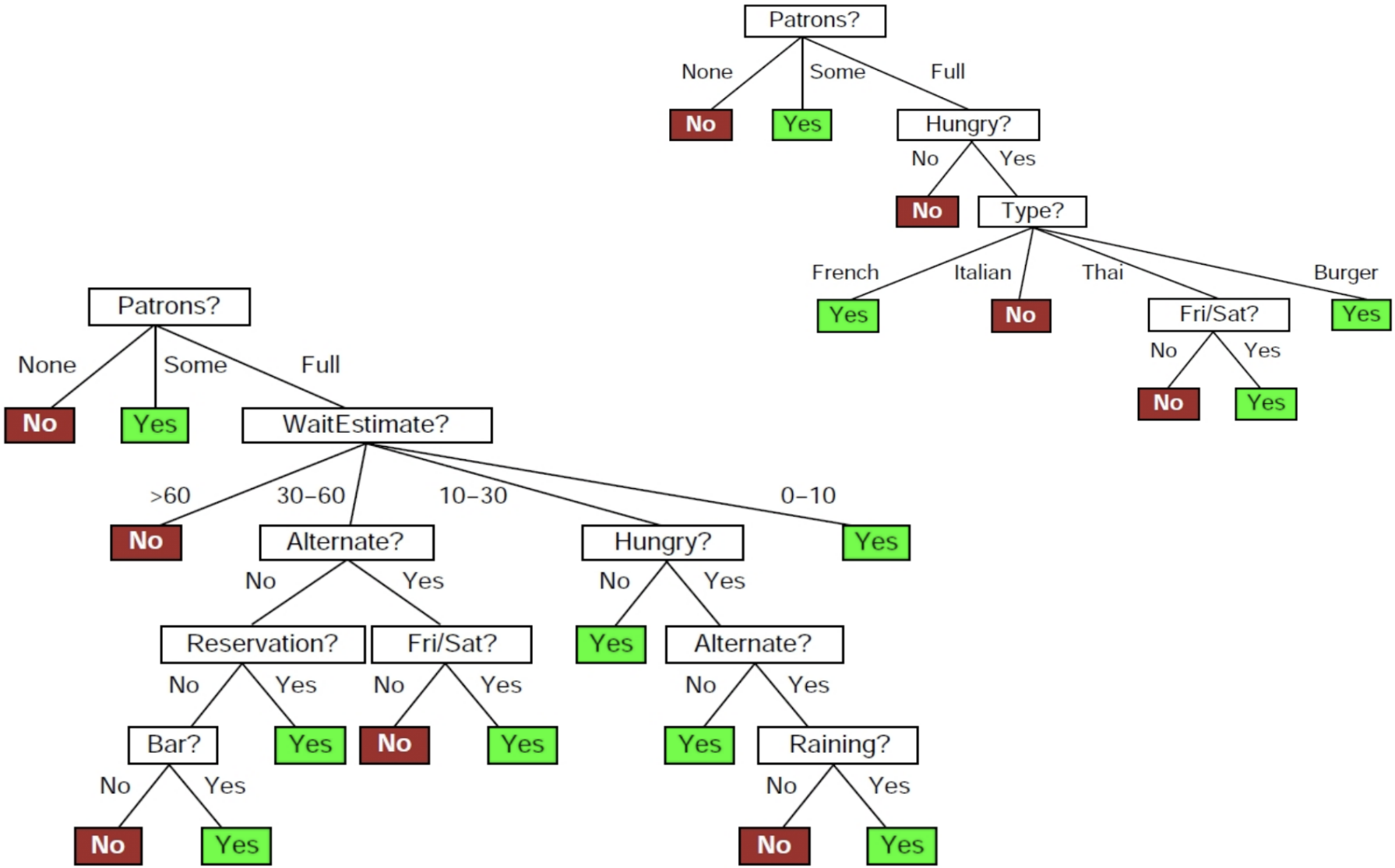


$$\mathrm{IG}(Y) = \mathrm{H}(Y) - \mathrm{H}(Y|X)$$

$$\mathrm{IG}(\mathsf{Type}) = 1 - \left[\frac{2}{12}\mathrm{H}(Y|\mathsf{Fr.}) + \frac{2}{12}\mathrm{H}(Y|\mathsf{It.}) + \frac{4}{12}\mathrm{H}(Y|\mathsf{Thai}) + \frac{4}{12}\mathrm{H}(Y|\mathsf{Bu.})\right]$$

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

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*:
  - Human interpretability
  - Avoid over-fitting training examples
  - Computational efficiency (avoid redundant, spurious attributes)
- *"Occam's Razor"*: find the simplest hypothesis that fits the observations
  - Useful principle, but hard to formalize (how to define simplicity?)
  - See Domingos, 1999
- We desire *small trees* with *informative nodes near the root*

Decision Tree Miscellany

- Problems? 🤔
  - Too big of a tree can *overfit* the data (extreme: one leaf per training sample)
  - You have *exponentially less data at lower levels* $\implies$ perhaps *prune* leaves w/ too little data
  - *Greedy algorithms* dont necessarily yield the *global optimum*, i.e., optimal choice at each step $\not\iff$ a globally optimal tree!
- Handling continuous attributes
  - Split based on a threshold, chosen to maximize information gain
- Note: decision trees can also be used for *regression on real-valued outputs* $\implies$ choose splits to *minimize squared error*, rather than maximize information gain.

k-NN versus Decision Tree

- DT $>$ k-NN:
  - Simple to deal with *discrete features*, *missing values*, and *poorly scaled data*
  - *Fast at test time* (k-NN needs to iterate through entire data set)
  - More *interpretable* (easily explain the decision making process)
- k-NN $>$ DT:
  - *Few hyperparameters* (just k)
  - Can incorporate *interesting distance measures* (e.g. shape contexts)

K-Nearest Neighbors

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

Decision Trees

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

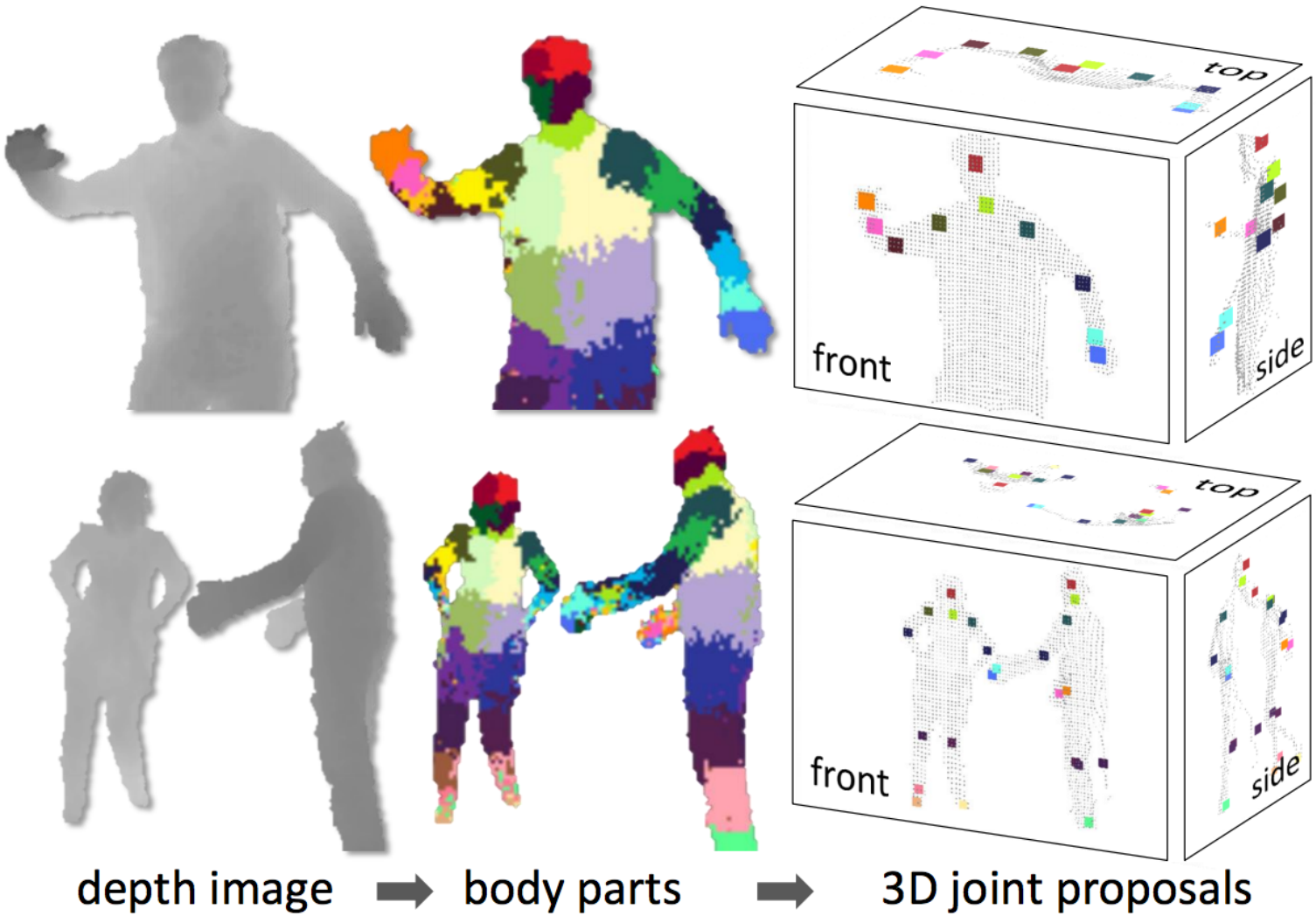
Applications of Decision Trees: XBox!

- 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]

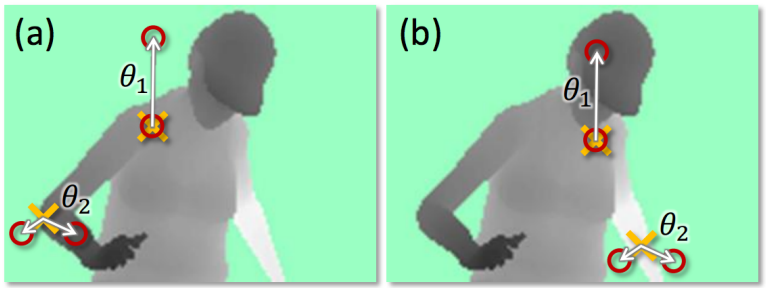
Applications of Decision Trees: XBox!

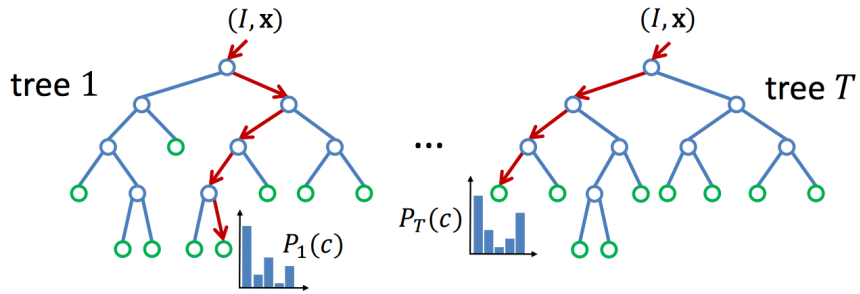
- Decision trees are in XBox: Classifying body parts



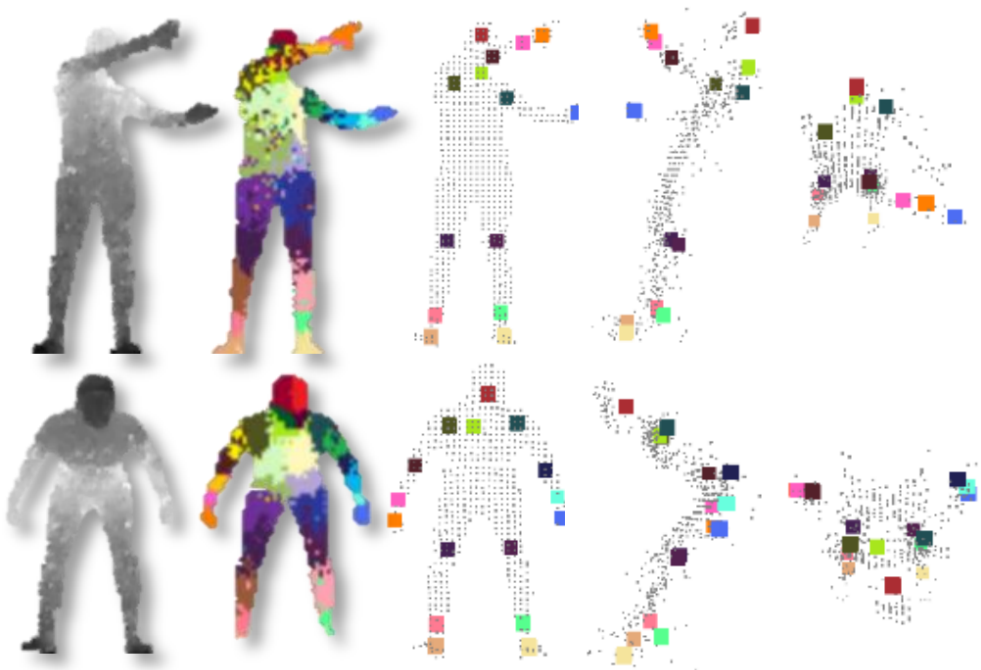
Applications of Decision Trees: XBox!

- Trained on million(s) of examples





- Results:



Ensembles

- We have seen a few classification algorithms (k-NN, DT, etc.)
- We can combine multiple classifiers into an *ensemble*, which is a set of predictors whose *individual decisions are combined in some way* to classify new examples
  - E.g., (possibly weighted) majority vote
- For this to be nontrivial, the classifiers must differ somehow, e.g., trained with...
  - different algorithm
  - different choice of hyperparameters
  - different sampling of training data
  - different weighting of training data
- Next lecture, we will study some specific ensembling techniques.

References I

Ash, R. B. (2012). *Information theory*. Courier Corporation.

Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons.

Domingos, P. (1999).The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, *3*, 409–425.

Laurent, H., & Rivest, R. L. (1976).Constructing optimal binary decision trees is np-complete. *Information processing letters*, *5*(1), 15–17.