AI and Machine Learning

Zhiyun Lin

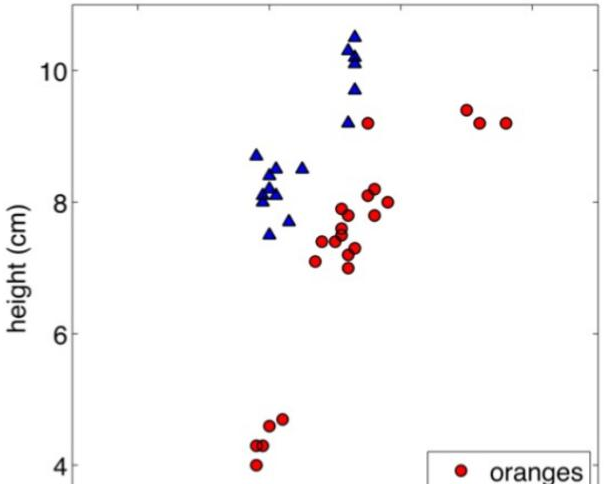
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Decision Trees

- Entropy
- Information Gain

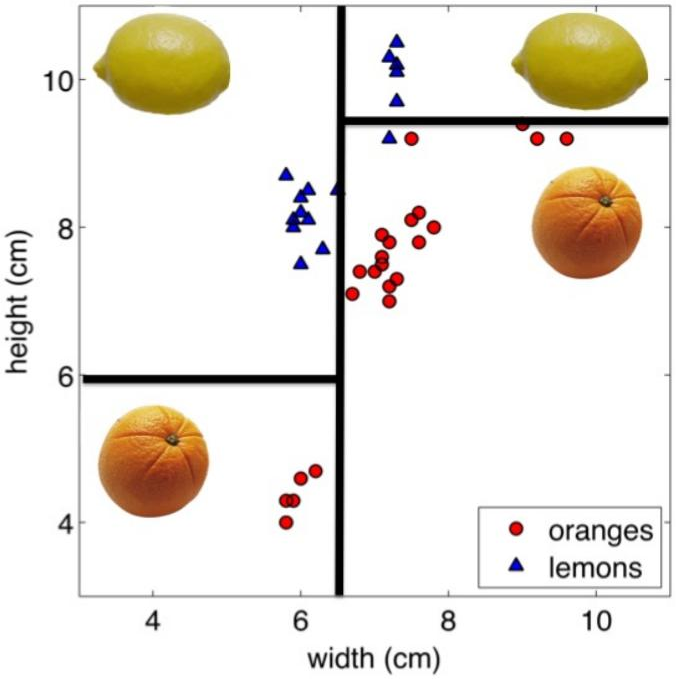
Another Classification Idea

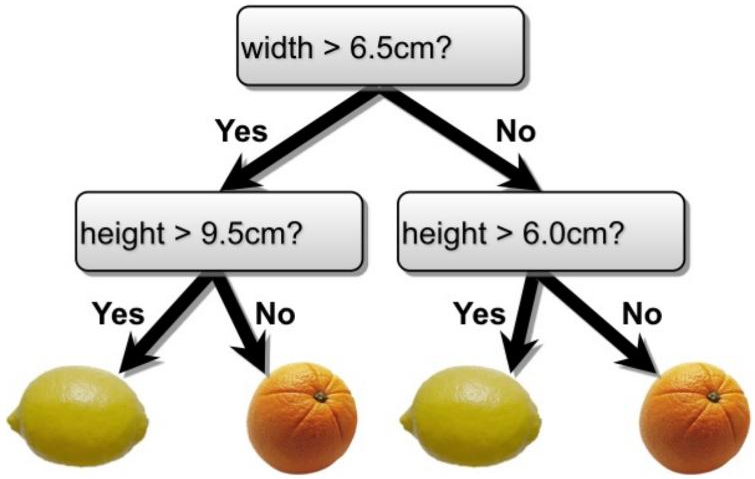
- We learned about linear classification (e.g., logistic regression), and nearest neighbors. Any other idea?
- **Pick an attribute**, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign a class with **majority vote**
- Do other branches as well



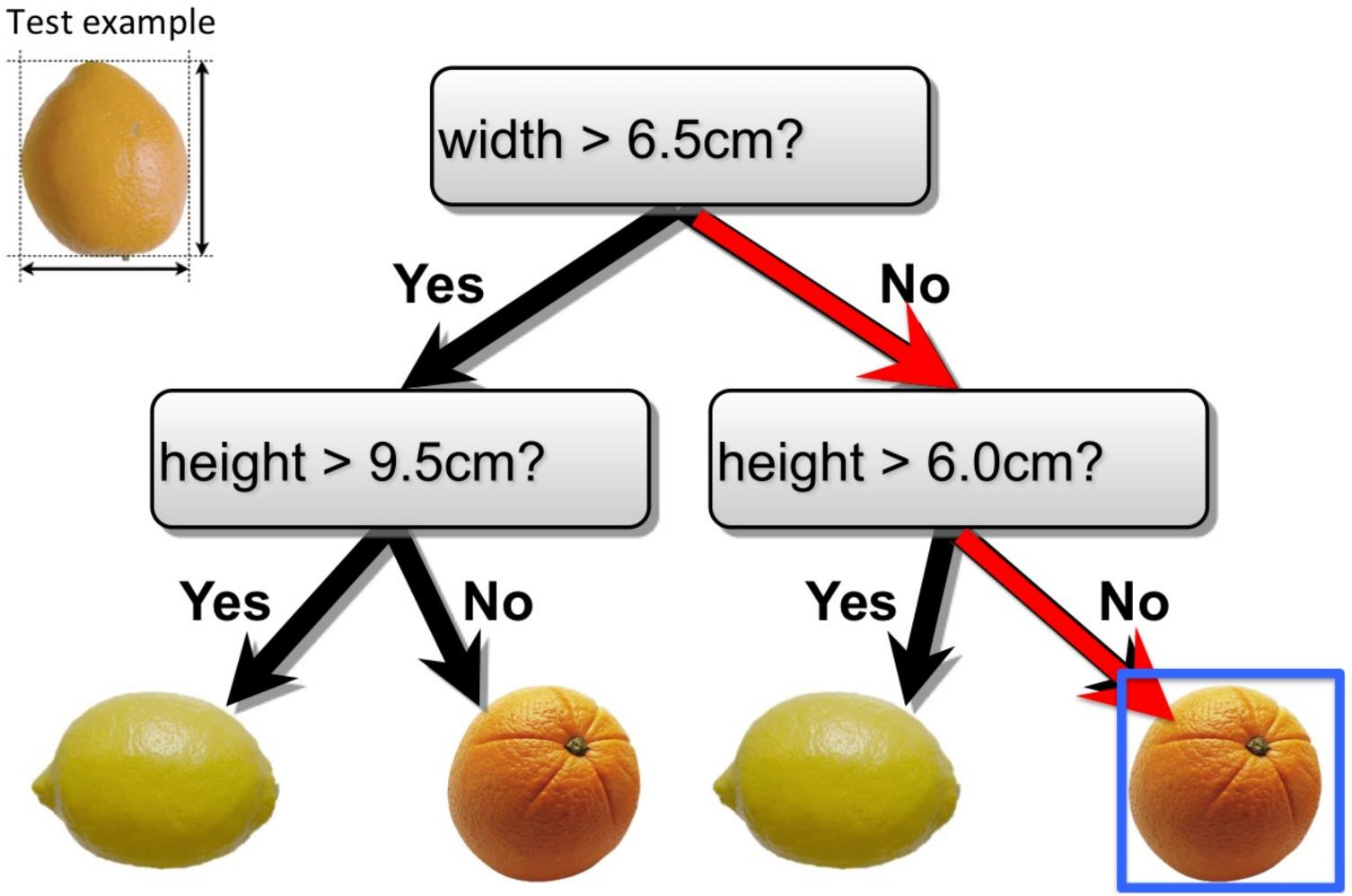
Another Classification Idea

- Gives **axes aligned decision boundaries**





Decision Tree: Classification



Example with Discrete Inputs

- What if the attributes are discrete?

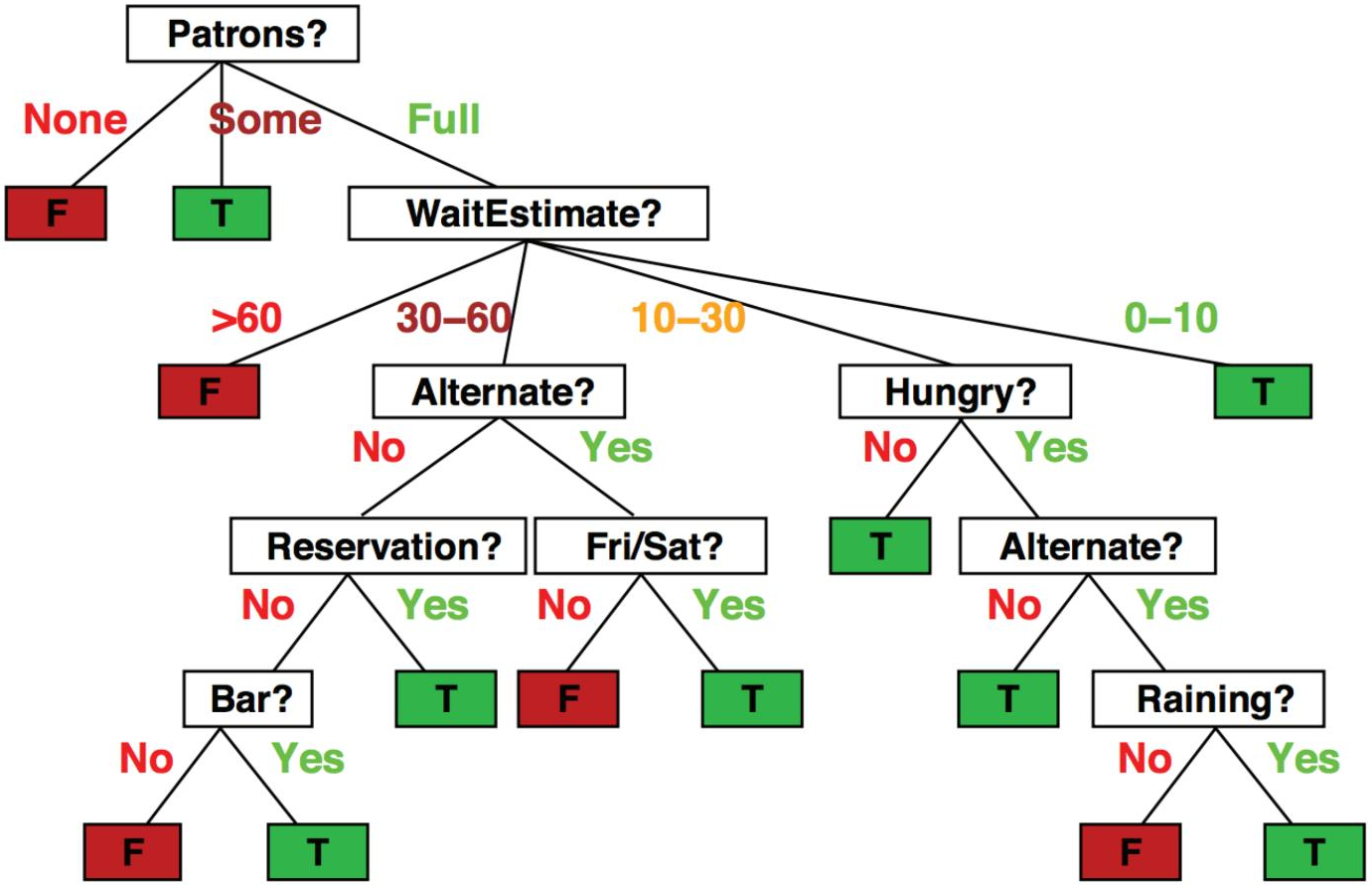
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$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

Decision Tree: Example with Discrete Inputs

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



Example with Discrete Inputs

- What if the attributes are discrete?

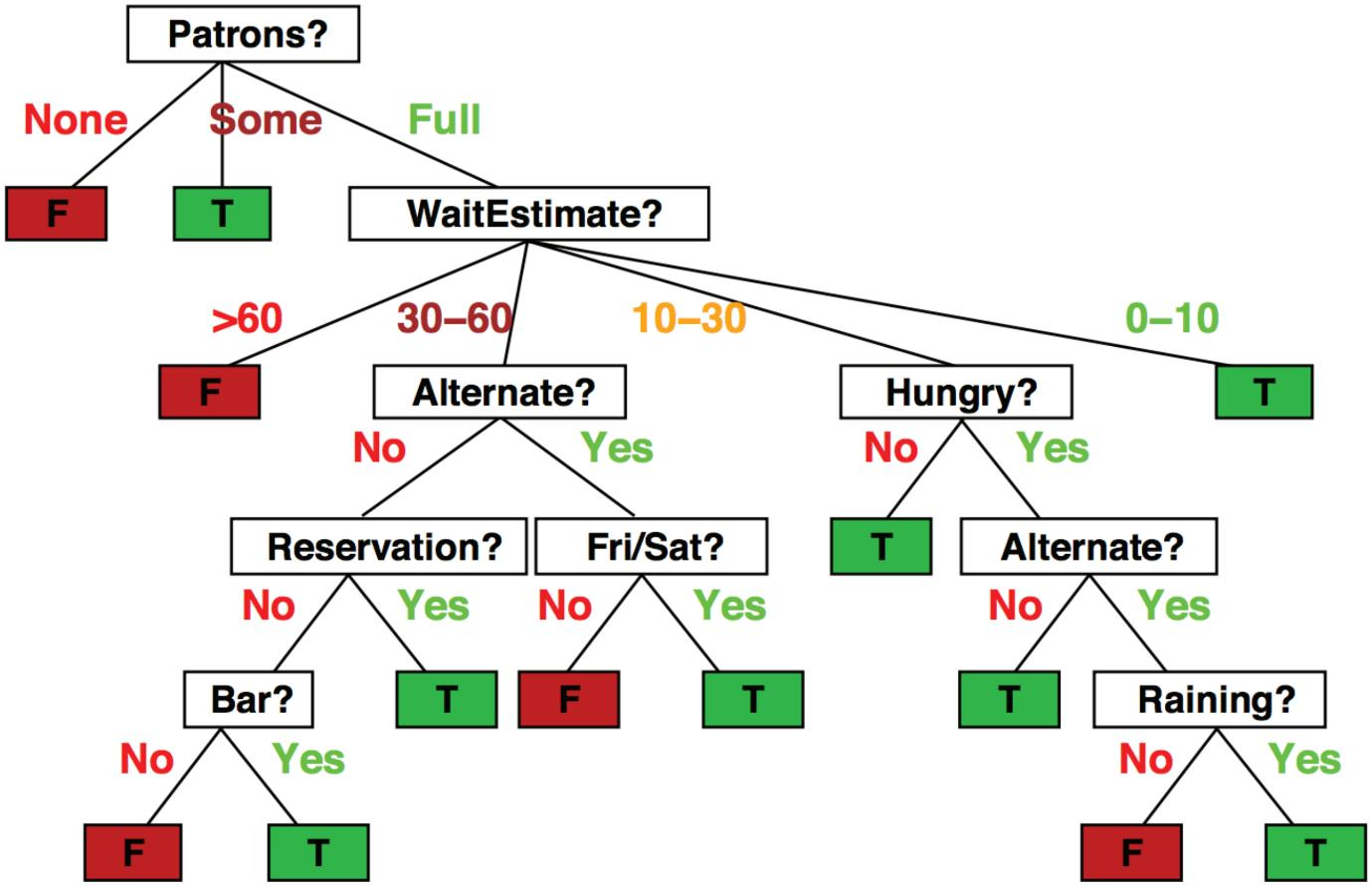
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$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

Decision Tree: Example with Discrete Inputs

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



Example with Discrete Inputs

- What if the attributes are discrete?

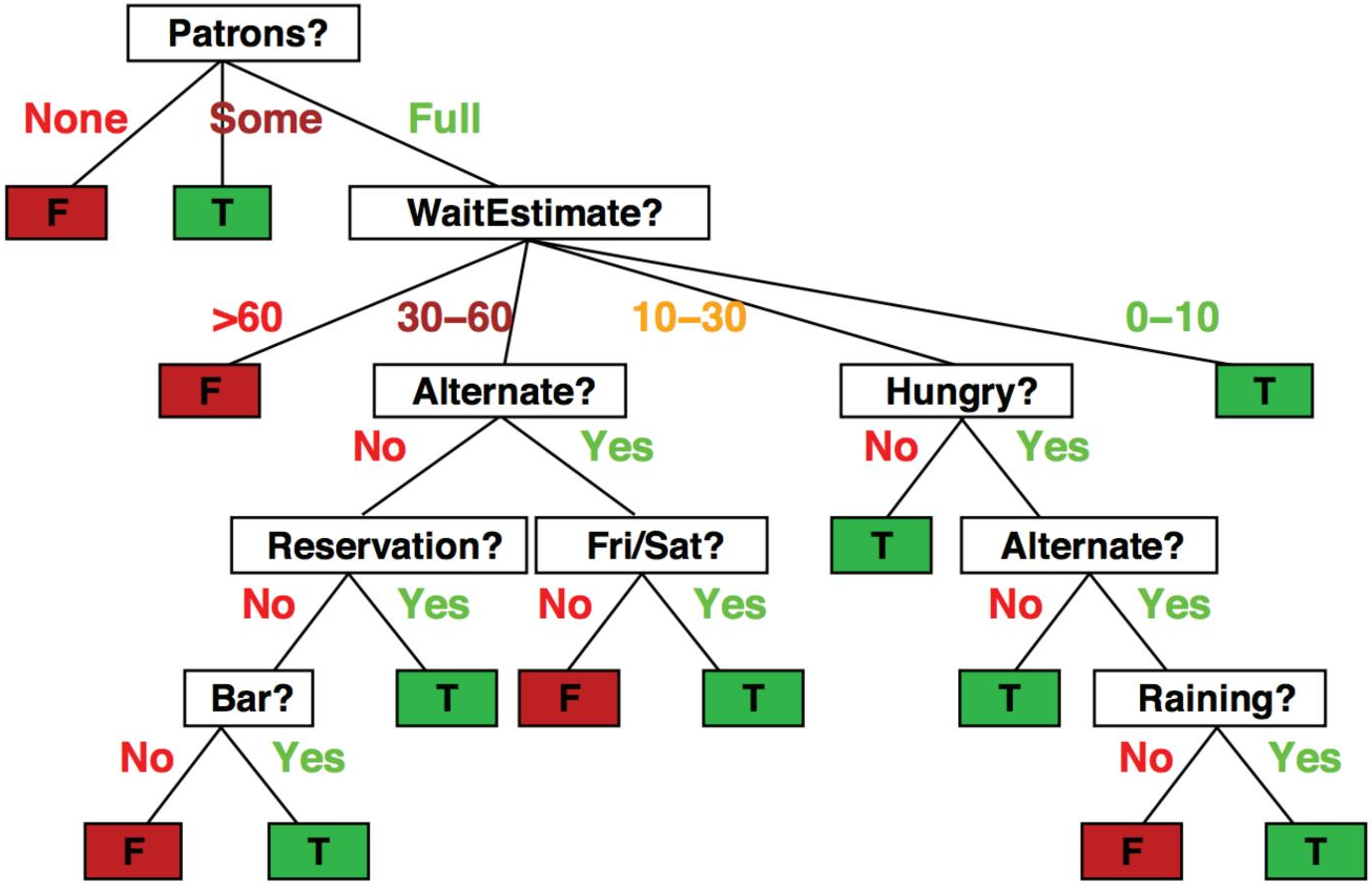
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$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

Decision Tree: Example with Discrete Inputs

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



Example with Discrete Inputs

- What if the attributes are discrete?

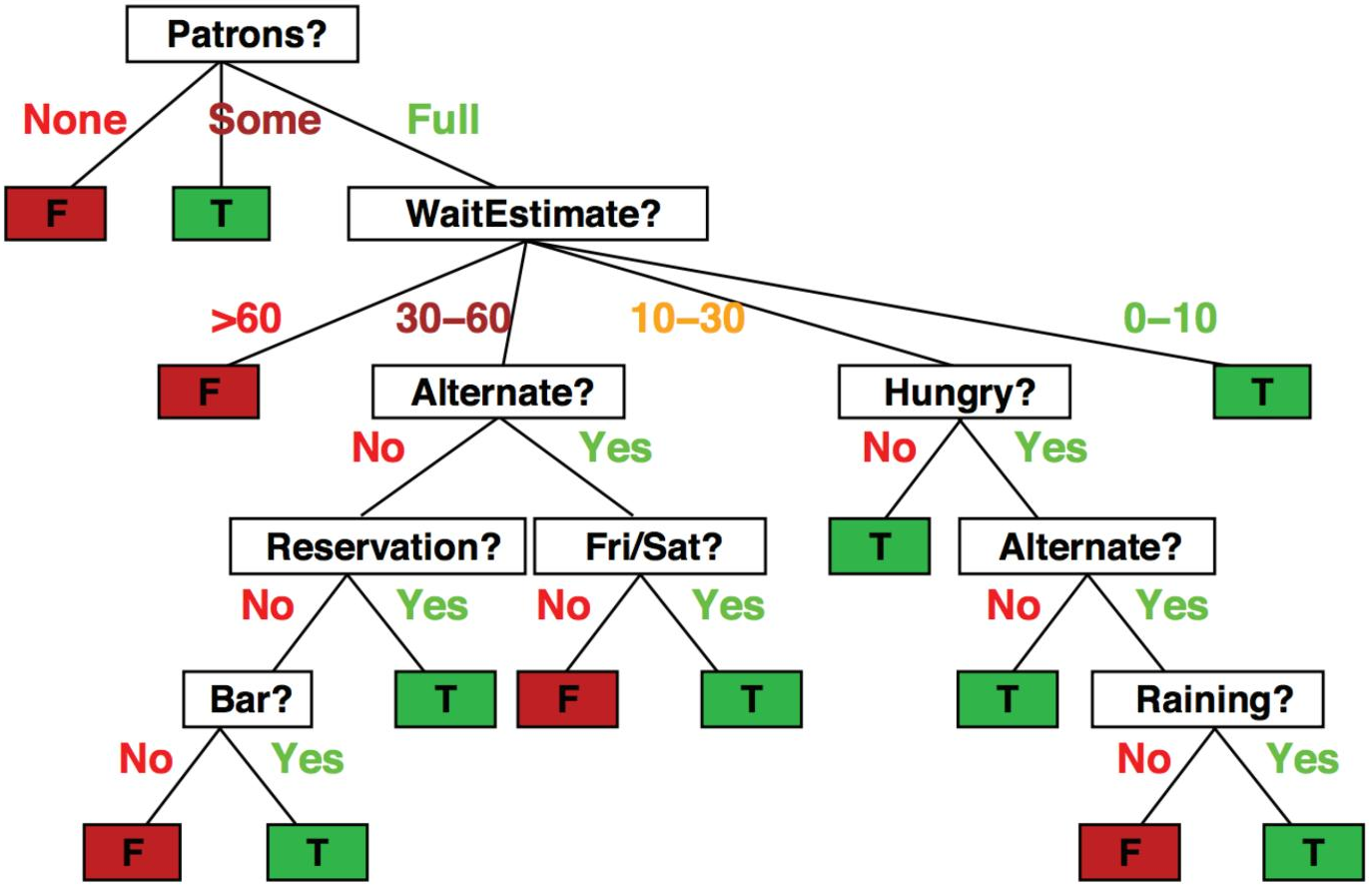
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$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

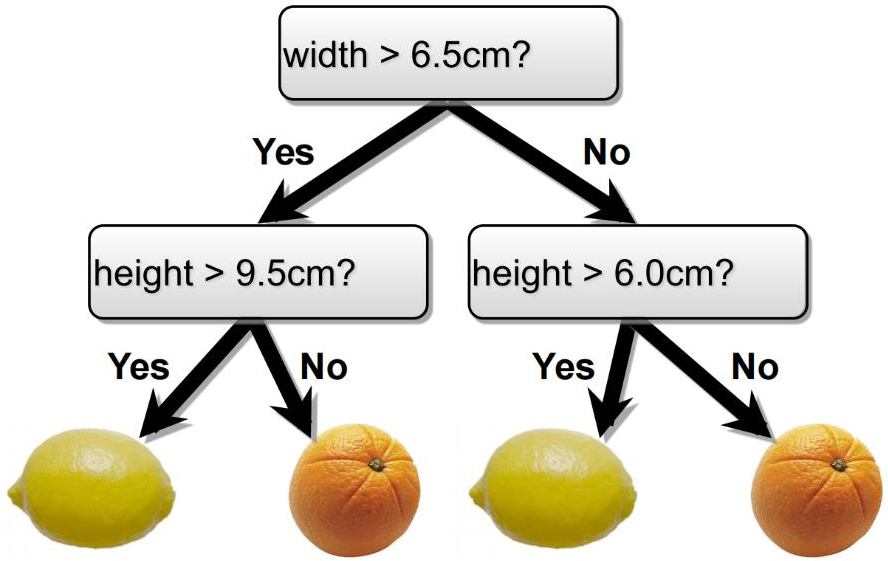
1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).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

Decision Tree: Example with Discrete Inputs

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



Decision Tree



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

Decision Tree

- Choose an attribute on which to descend ateach 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

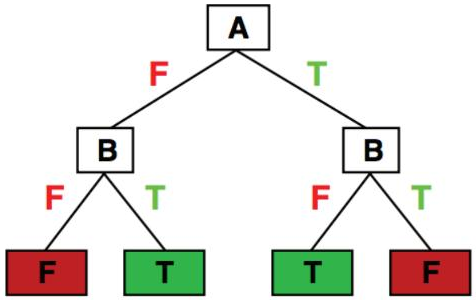
Decision Tree: Classification & Regression

- 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 $\{t^{(m_1)}, \cdots, t^{(m_k)}\}$
- **Regression tree:**
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \cdots, t^{(m_k)}\}$

Expressiveness

- **Discrete-input, discrete-output case:**
  - Decision trees can express any function of the input attributes
  - 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



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

How do we construct a useful decision tree?

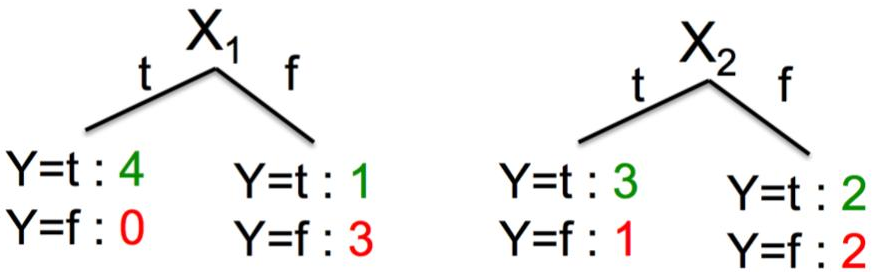
Learning Decision Trees

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?
- 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?

Detour: Information Theory

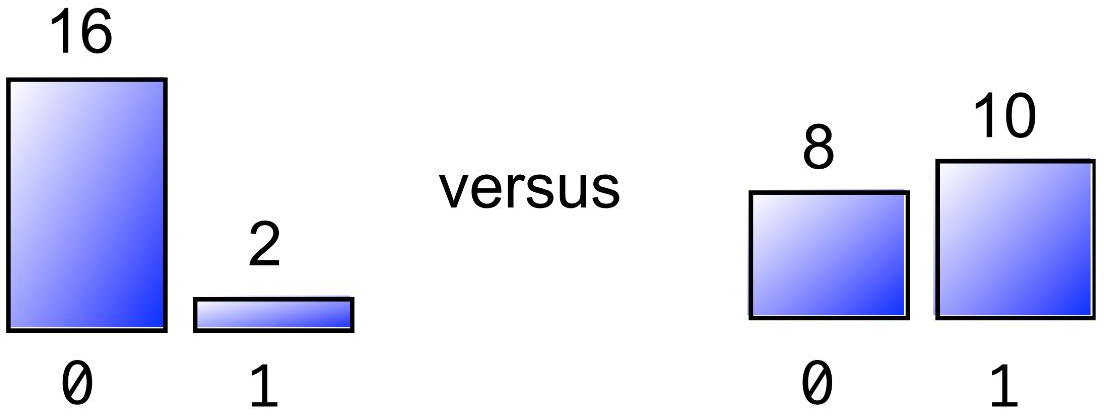
We Flip Two Different Coins

Sequence 1:

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

Sequence 2:

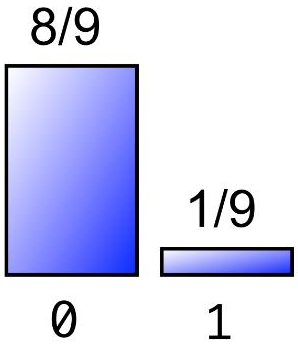
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



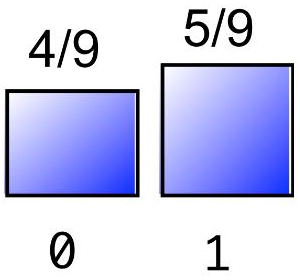
Quantifying Uncertainty

- **Entropy** H:

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



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

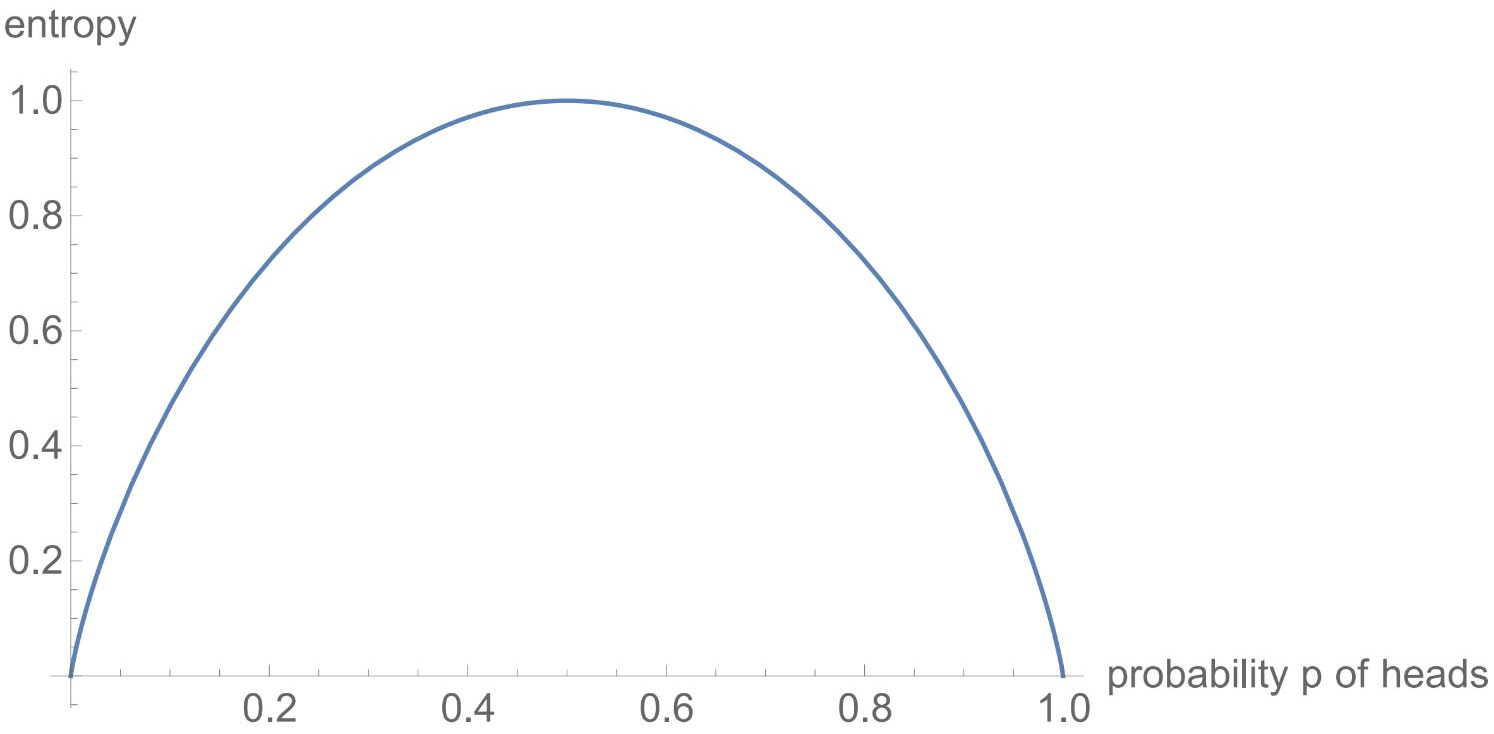


$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

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

Quantifying Uncertainty

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



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

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

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

Specific Conditional Entropy

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

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

	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**?

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

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

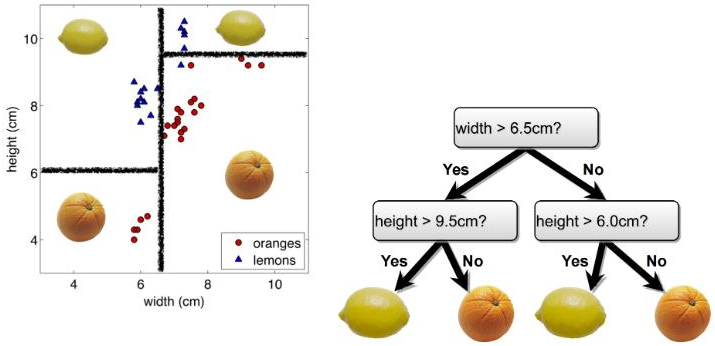
	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

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

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

- 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?

Construct Decision Tree



- 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:
  - Which variable to split.
  - Possibly where to split it.

Decision Tree Construction Algorithm

- Choose them based on how much information we would gain from the decision! (choose attribute that gives the highest gain)
- Simple, greedy, recursive approach, builds up tree node-by-node

  i. **pick an attribute to split at a non-terminal node**

  ii. **split examples into groups based on attribute value**

  iii. **for each group:**

    - **if** no examples - return majority from parent
    - **else if** all examples in same class - return class
    - **else loop** to step i.

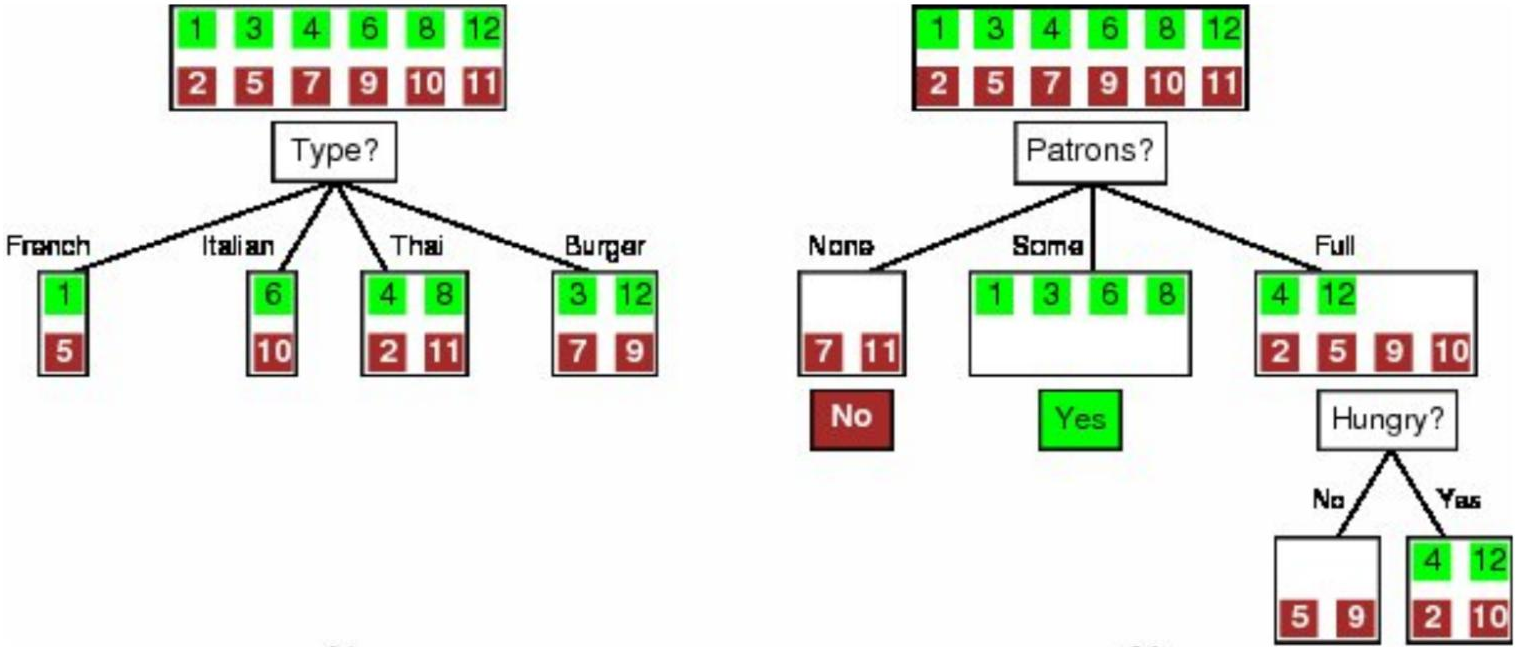
Back to Our Example

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

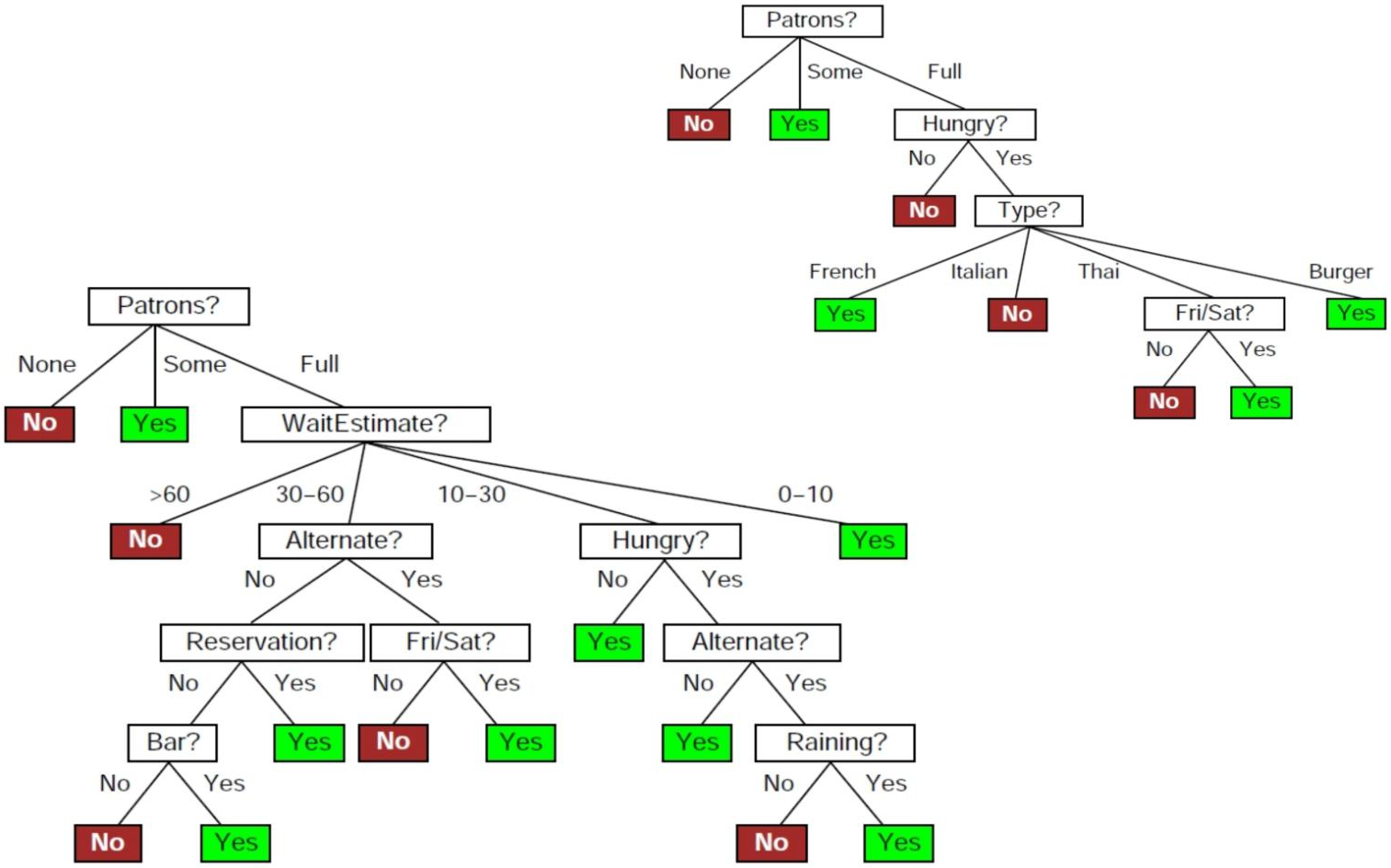


$$IG(Y) = H(Y) - H(Y|X)$$

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

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

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

An Example using Decision Tree

> Dataset

Suppose we have a dataset about fruits with the following attributes and classes:

- **Attributes:** Color (Color), Size (Size), Ripe (Ripe)
- **Classes:** Apple (A), Orange (O)

An Example: Data Set

Index	Color	Size	Ripe	Fruit
1	Green	Small	No	O
2	Yellow	Large	Yes	A
3	Red	Small	No	O
4	Yellow	Large	Yes	A
5	Red	Small	No	O
6	Green	Large	No	O
7	Red	Large	Yes	A
8	Yellow	Small	Yes	A
9	Green	Small	Yes	A
10	Red	Large	Yes	A

Step 1: Calculate the Entropy of the Target

The entropy $H(Y)$ is calculated as:

$$H(Y) = -p_{\text{A}} \log_2 p_{\text{A}} - p_{\text{O}} \log_2 p_{\text{O}}$$

where p_{A} and p_{O} are the proportions of Apples and Oranges in the dataset.

- From the dataset: **Total instances: 10; Apples (A): 6; Oranges (O): 4**

$$p_{\text{A}} = \frac{6}{10} = 0.6$$

$$p_{\text{O}} = \frac{4}{10} = 0.4$$

$$H(Y) = -0.6 \log_2 0.6 - 0.4 \log_2 0.4 \approx 0.971$$

Step 2: Information Gain for Each Attribute

Color

- Green: 3 instances (2 O, 1 A)
- Yellow: 3 instances (0 O, 3 A)
- Red: 4 instances (2 O, 2 A)

$$
\begin{aligned}
H(Y|Color) &= \tfrac{3}{10}H(Y|green) + \tfrac{3}{10}H(Y|Yellow) + \tfrac{4}{10}H(Y|red) \\
&= \tfrac{3}{10}\left[-\tfrac{2}{3}\log_2\tfrac{2}{3} - \tfrac{1}{3}\log_2\tfrac{1}{3}\right] \\
&\quad + \tfrac{3}{10}\left[-\tfrac{0}{3}\log_2\tfrac{0}{3} - \tfrac{3}{3}\log_2\tfrac{3}{3}\right] \\
&\quad + \tfrac{4}{10}\left[-\tfrac{2}{4}\log_2\tfrac{2}{4} - \tfrac{2}{4}\log_2\tfrac{2}{4}\right] \\
&\approx 0.8
\end{aligned}
$$

The Information Gain $IG(Color)$ is calculated as:

$$IG(Color) = H(Y) - H(Y|Color) = 0.971 - 0.8 = 0.171$$

Step 2: Information Gain for Each Attribute

Size

- Small: 5 instances (4 O, 1 A)
- Large: 5 instances (2 O, 3 A)

$$\begin{aligned} H(Y|Size) &= \tfrac{1}{2}H(Y|small) + \tfrac{1}{2}H(Y|large) \\ &= \tfrac{1}{2}\left[-\tfrac{4}{5}\log_2\tfrac{4}{5} - \tfrac{1}{5}\log_2\tfrac{1}{5}\right] \\ &\quad + \tfrac{1}{2}\left[-\tfrac{2}{5}\log_2\tfrac{2}{5} - \tfrac{3}{5}\log_2\tfrac{3}{5}\right] \\ &\approx 0.8631 \end{aligned}$$

The Information Gain $IG(Size)$ is calculated as:

$$IG(Size) = H(Y) - H(Y|Size) = 0.971 - 0.8631 = 0.1079$$

Step 2: Information Gain for Each Attribute

Ripe

- No: 6 instances (4 O, 2 A)
- Yes: 4 instances (0 O, 4 A)

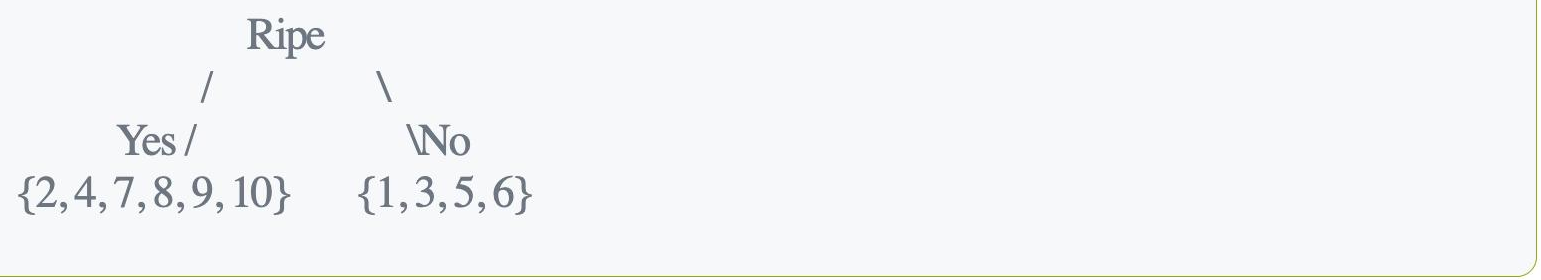
$$
\begin{aligned}
H(Y|Ripe) &= \tfrac{6}{10} H(Y|no) + \tfrac{4}{10} H(Y|yes) \\
&= \tfrac{6}{10}\left[-\tfrac{4}{6}\log_2 \tfrac{4}{6} - \tfrac{2}{6}\log_2 \tfrac{2}{6}\right] \\
&\quad + \tfrac{4}{10}\left[-\tfrac{0}{4}\log_2 \tfrac{0}{4} - \tfrac{4}{4}\log_2 \tfrac{4}{4}\right] \\
&\approx 0.5510
\end{aligned}
$$

The Information Gain $IG(Ripe)$ is calculated as:

$$IG(Ripe) = H(Y) - H(Y|Ripe) = 0.971 - 0.5510 = 0.42$$

Step 3: Choose the Attribute with the Highest Information Gain

- We compare the Information Gain for each attribute and choose the one with the highest value to split on.
- In this case, the attribute with the highest Information Gain is **Ripe**.



Data Set: Ripe-Yes-Subset

Index	Color	Size	Ripe	Fruit
2	Yellow	Large	Yes	A
4	Yellow	Large	Yes	A
7	Red	Large	Yes	A
8	Yellow	Small	Yes	A
9	Green	Small	Yes	A
10	Red	Large	Yes	A

Data Set: Ripe-No-Subset

Index	Color	Size	Ripe	Fruit
1	Green	Small	No	O
3	Red	Small	No	O
5	Red	Small	No	O
6	Green	Large	No	O

Step 4: Build the Decision Tree

- We repeat the process for **each subset of the data**, recursively building the tree until all instances are classified correctly or we run out of attributes.
- To prevent overfitting, we might prune the tree by removing branches that have little effect on the overall performance.

Handle Continuous Attributes

- When using Information Gain to construct a decision tree and the input attribute is continuous, we need to select an appropriate threshold to split the data.
- Selecting the right threshold is crucial for building an effective decision tree.
- Here are the general steps to choose a threshold:
  - **Step 1: Sort the Data**

    First, sort the data based on the values of the continuous attribute.
  - **Step 2: Calculate All Possible Thresholds**

    For the sorted data, thresholds can be taken as the midpoint between any two consecutive values. For example, if the sorted data is [1, 2, 3, 4, 5], then the possible thresholds are 1.5, 2.5, 3.5, and 4.5.

Handle Continuous Attributes

- Cont'd
  - **Step 3: Calculate the Information Gain for Each Threshold**

    For each possible threshold, split the data into two parts: values less than or equal to the threshold and values greater than the threshold. Then, calculate the Information Gain for each split.
  - **Step 4: Choose the Threshold with Maximum Information Gain**

    From all possible thresholds, select the one with the maximum Information Gain as the final threshold.

Example Code

```
import numpy as np
from collections import Counter

def calculate_entropy(labels):
    total_count = len(labels)
    counts = Counter(labels)
    entropy = 0.0
    for count in counts.values():
        probability = count / total_count
        entropy -= probability * np.log2(probability)
    return entropy
```

Example Code (cont'd)

```
def information_gain(data, labels, threshold):
    total_entropy = calculate_entropy(labels)    # 计算总熵

    # 根据阈值分割数据
    left_indices = [i for i, x in enumerate(data) if x <= threshold]
    right_indices = [i for i, x in enumerate(data) if x > threshold]

    # 计算分割后的熵
    n = len(labels)
    left_labels = [labels[i] for i in left_indices]
    right_labels = [labels[i] for i in right_indices]
    left_entropy = calculate_entropy(left_labels)
    right_entropy = calculate_entropy(right_labels)

    # 计算信息增益
    weighted_entropy = (len(left_labels) / n) * left_entropy + (len(right_labels) / n) * right_entropy
    info_gain = total_entropy - weighted_entropy
    return info_gain
```

Example Code (cont'd)

```
# 示例数据
data = [1, 2, 3, 4, 5]
labels = ['A', 'A', 'B', 'B', 'B']

# 计算所有可能阈值的信息增益
thresholds = [(data[i] + data[i+1]) / 2 for i in range(len(data)-1)]
info_gains = [information_gain(data, labels, th) for th in thresholds]

# 选择信息增益最大的阈值
best_threshold = thresholds[np.argmax(info_gains)]
print(f"Best threshold: {best_threshold}")
```

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

- 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