CSC 311: Introduction to Machine Learning

Lecture 2 - Decision Trees

Amir-massoud Farahmand & Emad A.M. Andrews

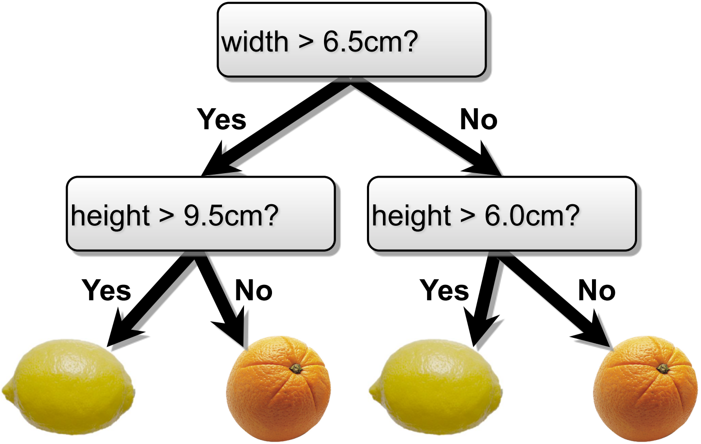
University of Toronto

Today

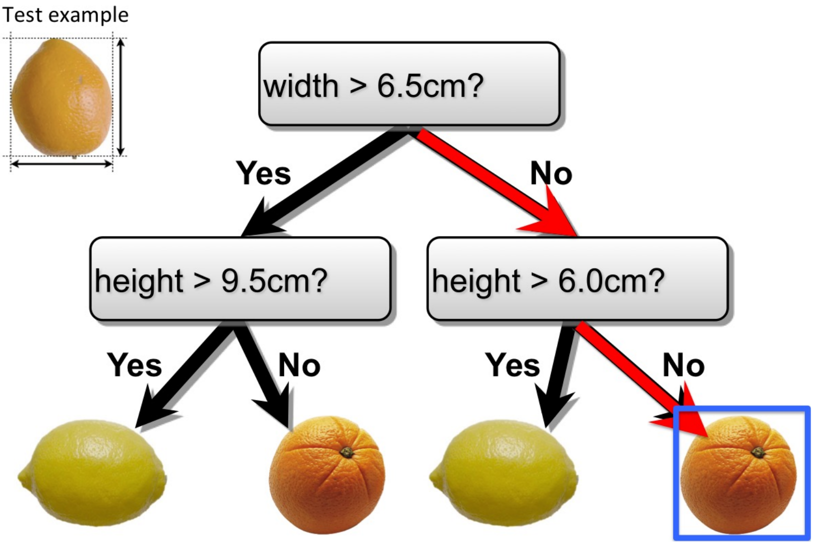
- **Decision Trees**
  - Simple but powerful learning algorithm
  - One of the most widely used learning algorithms in Kaggle competitions
  - Lets us introduce ensembles, a key idea in ML
- Useful information theoretic concepts (entropy, mutual information, etc.)

Decision Trees

- **Decision trees** make predictions by recursively splitting on different attributes according to a tree structure.
- Example: classifying fruit as an orange or lemon based on height and width

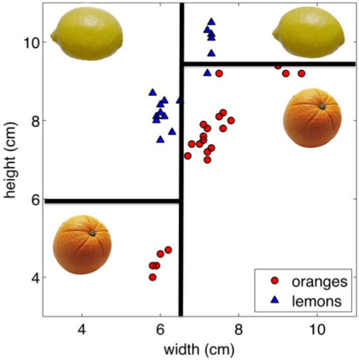


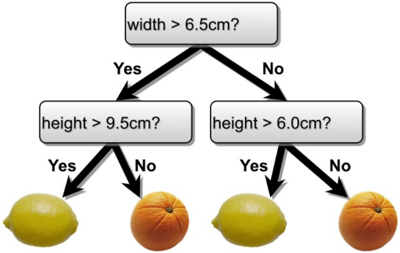
Decision Trees



Decision Trees

- For continuous attributes, split based on less than or greater than some threshold
- Thus, input space is divided into regions with boundaries parallel to axes





Example with Discrete Inputs

- What if the attributes are discrete?

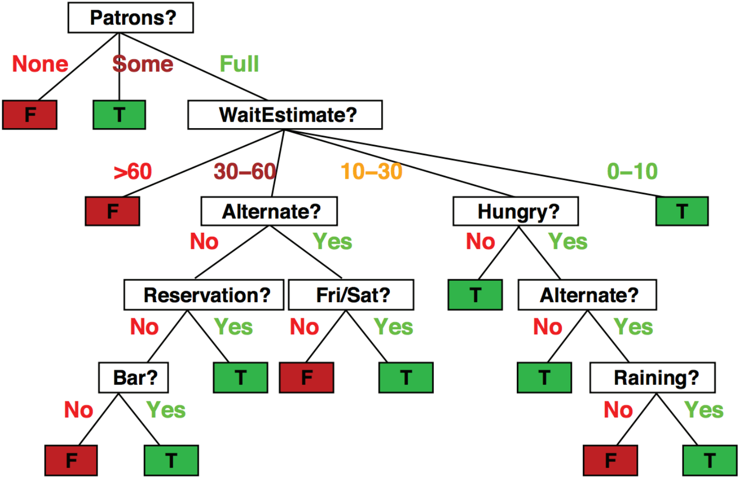
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3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

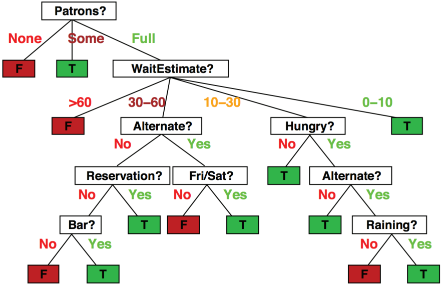
1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).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

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



Decision Trees

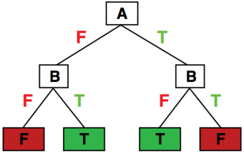


- **Internal nodes** test **attributes**
- **Branching** is determined by **attribute value**
- **Leaf nodes** are **outputs** (predictions)

Expressiveness

- **Discrete-input, discrete-output case**:
  - Decision trees can express any function of the input attributes
  - Example: For Boolean functions, the truth table row $\rightarrow$ path to leaf

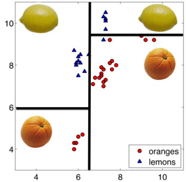
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- **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

[Slide credit: S. Russell]

Decision Tree: Classification and Regression



- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ 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 focus on classification

How do we Learn a DecisionTree?

- How do we construct a useful decision tree?

Learning Decision Trees

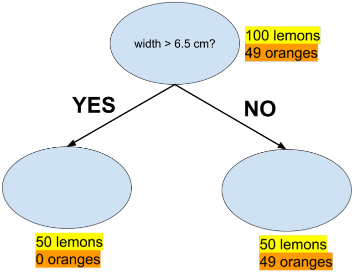
Learning the simplest (smallest) decision tree which correctly classifies training set is an NP complete problem (if you are interested, check: Hyafil & Rivest'76).

- Resort to a **greedy heuristic**! Start with empty decision tree and complete training set
  - Split on the "best" attribute, i.e. partition dataset
  - Recurse on subpartitions

- When should we stop?
- Which attribute is the "best" (and where should we split, if continuous)?
  - Choose based on accuracy?
  - Loss: misclassification error
  - Say region R is split in R_1 and R_2 based on loss $L(R)$.
  - Accuracy gain is $L(R) - \frac{|R_1|L(R_1)+|R_2|L(R_2)}{|R_1|+|R_2|}$

Choosing a Good Split

- Why isn't accuracy a good measure?
- Classify by the majority, loss is the misclassification error.



- Is this split good? Zero accuracy gain

$$L(R) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = \frac{49}{149} - \frac{50 \times 0 + 99 \times \frac{49}{99}}{149} = 0$$

- But we have reduced our uncertainty about whether a fruit is a lemon!

Choosing a Good Split

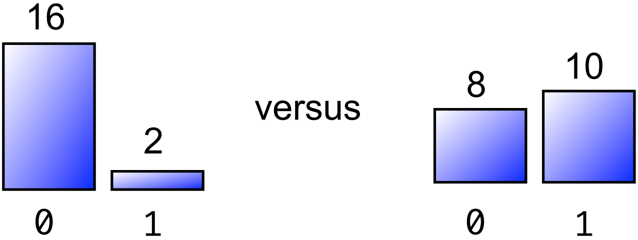
- How can we quantify uncertainty in prediction for a given leaf node?
  - All examples in leaf have the same class: good (low uncertainty)
  - Each class has the same number of examples in leaf: bad (high uncertainty)
- **Idea:** Use counts at leaves to define probability distributions, and use information theory to measure uncertainty

Flipping Two Different Coins

Q: Which coin is more uncertain?

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

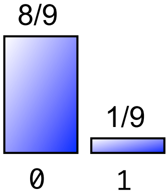
Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

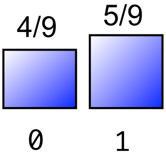


Quantifying Uncertainty

Entropy is a measure of expected "surprise": How uncertain are we of the value of a draw from this distribution?

$$H(X) = -\mathbb{E}_{X\sim p}[\log_2 p(X)] = -\sum_{x\in X} p(x)\log_2 p(x)$$





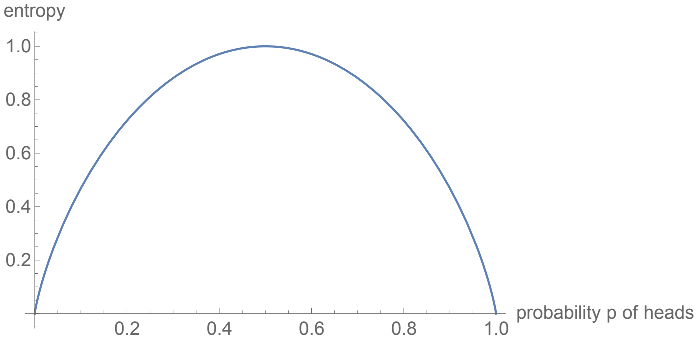
$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

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

- Averages over information content of each observation
- Unit = **bits** (based on the base of logarithm)
- A fair coin flip has 1 bit of entropy

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 peaks and valleys
  - Histogram has 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**?

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

- 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:

$$
\begin{aligned}
H(Y|X) &= \mathbb{E}_{X\sim p(x)}[H(Y|X)] \qquad (1) \\
&= \sum_{x\in X} p(x) H(Y|X=x) \\
&= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(y|x) \\
&= -\mathbb{E}_{(X,Y)\sim p(x,y)}[\log_2 p(Y|X)]
\end{aligned}
$$

Conditional Entropy

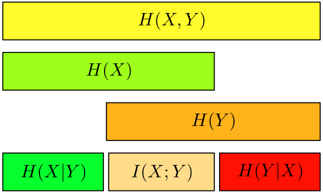
- Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	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?

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

Conditional Entropy



- Some useful properties for the discrete case:
  - 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 does not tell us anything about Y: $H(Y|X) = H(Y)$.
  - If X and Y independent, then $H(X,Y) = H(X) + H(Y)$.
  - But Y tells us everything about Y: $H(Y|Y) = 0$.
  - By knowing X, we can only decrease uncertainty about Y: $H(Y|X) \leq H(Y)$.

Exercise: Verify these!

The figure is reproduced from Fig 8.1 of MacKay, Information Theory, Inference, and ... .

Information Gain

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

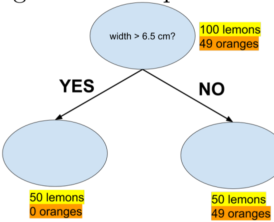
- How much information about cloudiness do we get by discovering whether it is raining?

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

- This is called the **information gain** in Y due to X, or the **mutual information** of Y and X
- If X is completely uninformative about Y: $IG(Y|X) = 0$
- If X is completely informative about Y: $IG(Y|X) = H(Y)$

Revisiting Our Original Example

- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree attribute!
- What is the information gain of this split?

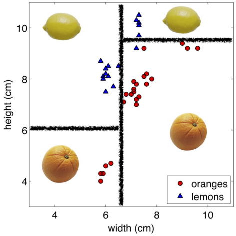


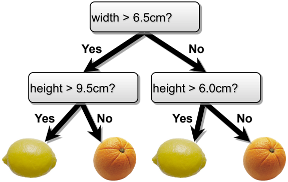
- Let Y be r.v. denoting lemon or orange, B be r.v. denoting whether left or right split taken, and treat counts as probabilities.
- Root entropy: $H(Y) = -\frac{49}{149}\log_2(\frac{49}{149}) - \frac{100}{149}\log_2(\frac{100}{149}) \approx 0.91$
- Leafs entropy: $H(Y|B = \text{left}) = 0$, $H(Y|B = \text{right}) \approx 1$.
- $IG(Y|B) = H(Y) - H(Y|B)$

$$= H(Y) - \{H(Y|B{=}\text{left})\mathbb{P}(B{=}\text{left}) + H(Y|B{=}\text{right})\mathbb{P}(B{=}\text{right})\}$$

$$\approx 0.91 - (0 \cdot \tfrac{1}{3} + 1 \cdot \tfrac{2}{3}) \approx 0.24 > 0$$

Constructing Decision Trees





- 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 **best** gain)

Decision Tree Construction Algorithm

- Simple, greedy, recursive approach, builds up tree node-by-node
- Start with empty decision tree and complete training set
  - Split on the most informative attribute, partitioning dataset
  - Recurse on subpartitions
- Possible termination condition: end if all examples in current subpartition share the same class

Back to Our Example

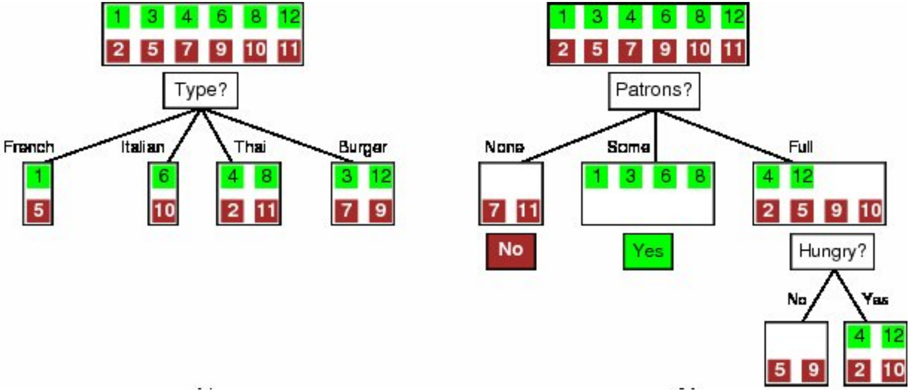
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3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

[from: Russell & Norvig]

Attribute Selection

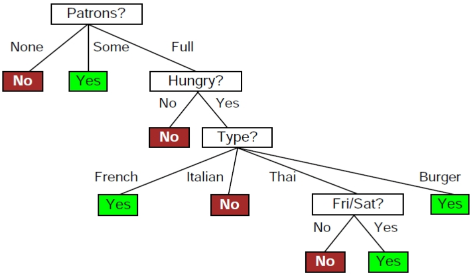


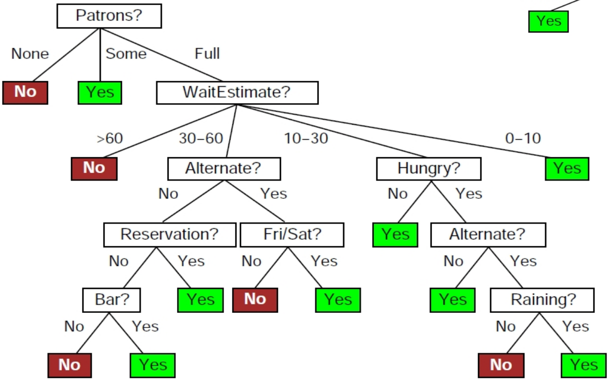
$$IG(Y) = H(Y) - H(Y|X)$$

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

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

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
  - Human interpretability
- "Occam's Razor": find the simplest hypothesis that fits the observations
  - Useful principle, but hard to formalize (how to define simplicity?)
  - See Domingos, 1999, "The role of Occam's razor in knowledge discovery"
- We desire small trees with informative nodes near the root

Decision Tree Miscellany

- Problems:
  - You have exponentially less data at lower levels
  - A large tree can overfit the data
  - Greedy algorithms don't necessarily yield the global optimum
  - Mistakes at top-level propagate down tree
- Handling continuous attributes
  - Split based on a threshold, chosen to maximize information gain
- There are other criteria used to measure the quality of a split, e.g., Gini index
- Trees can be pruned in order to make them less complex
- Decision trees can also be used for regression on real-valued outputs. Choose splits to minimize squared error, rather than maximize information gain.

Comparison to k-NN

Advantages of decision trees over k-NN

- Good with discrete attributes
- Easily deals with missing values (just treat as another value)
- Robust to scale of inputs; only depends on ordering
- Good when there are lots of attributes, but only a few are important
- Fast at test time
- More interpretable

Comparison to k-NN

Advantages of k-NN over decision trees

- Able to handle attributes/features that interact in complex ways
- Can incorporate interesting distance measures, e.g., shape contexts.