

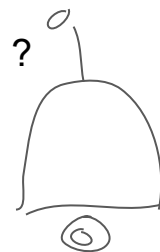
Decision Trees

CS 480

Intro to Artificial Intelligence

An example classification task

“Should you eat at this new restaurant?”



Binary Features

- Alternate: suitable alternative nearby
- Bar: has a bar to wait in
- Fri/Sat: it's Friday/Saturday
- Hungry: you are hungry
- Raining: is it currently raining
- Reservation: made a reservation

Categorical Features

- Patrons: how many people are inside (None, Some, Full)
- Price: (\$, \$\$, \$\$\$)
- Type: (French, Italian, Thai, Burger)
- WaitEstimate: estimated time to get a table (0-10, 10-30, 30-60, >60)

$\mathbf{x} = \langle y, y, n, y, \text{Some}, \$, n, n, \text{Thai}, 0-10 \rangle$

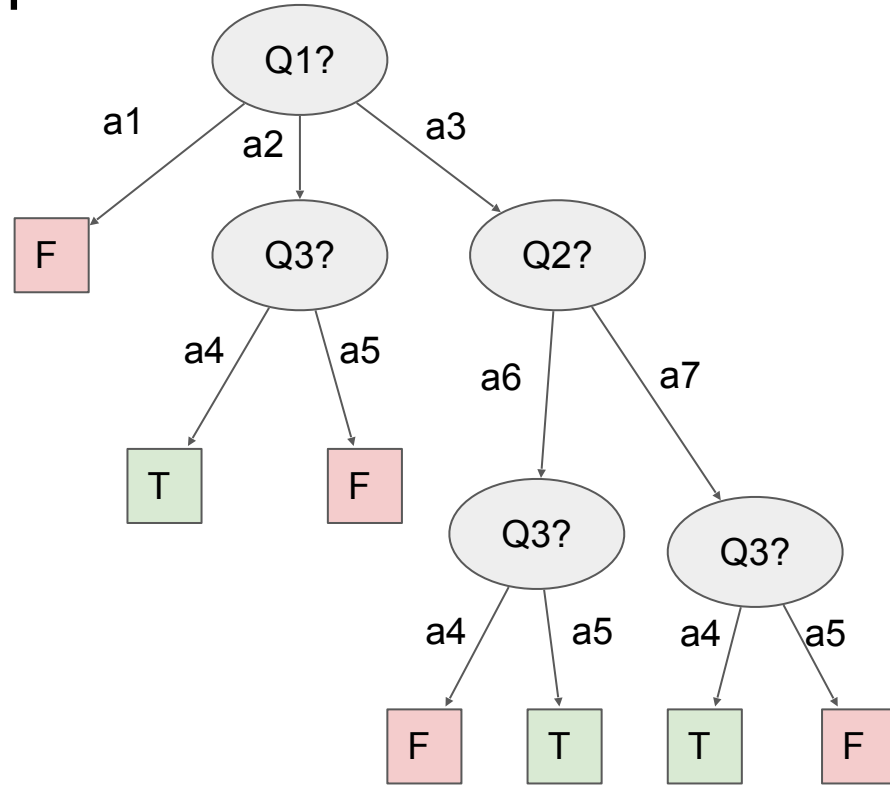
$y = \text{True}$

Given \mathbf{x} , how can we predict y ?

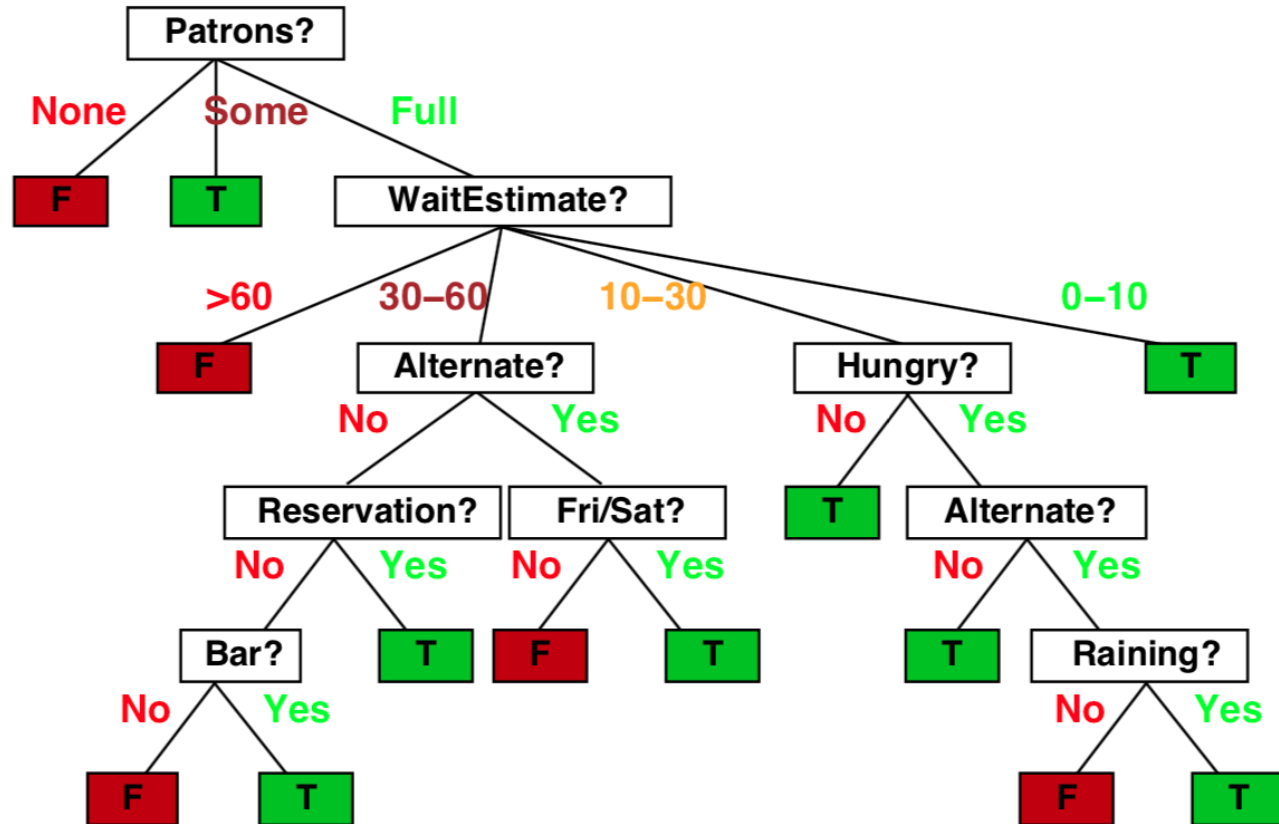
Decision Tree Representation

Decision is based on the answers to a sequence of questions, represented as a tree

- Nodes represent questions about x (“attributes” or “features”)
- Edges represent values that attributes take
- Leaves represent output (final decision)
- Sequence/number of questions may depend on answers (not a full/complete tree)
- Doesn't make sense to ask the same question more than once on the same path from the root (already know the answer)



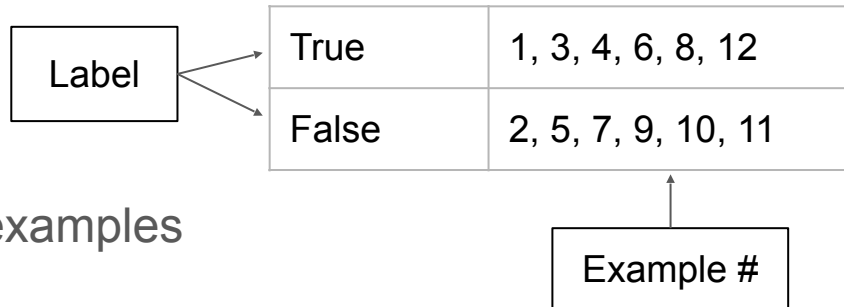
A tree for our example



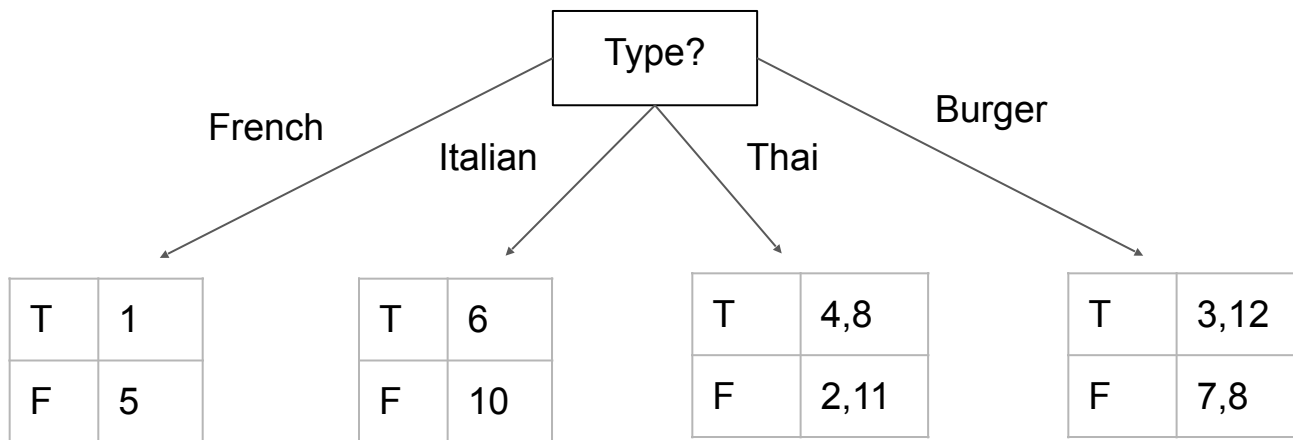
Data for our example

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
X_1	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>0–10</i>	<i>T</i>
X_2	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>30–60</i>	<i>F</i>
X_3	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>Some</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>0–10</i>	<i>T</i>
X_4	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>10–30</i>	<i>T</i>
X_5	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>>60</i>	<i>F</i>
X_6	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Italian</i>	<i>0–10</i>	<i>T</i>
X_7	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>0–10</i>	<i>F</i>
X_8	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Thai</i>	<i>0–10</i>	<i>T</i>
X_9	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>>60</i>	<i>F</i>
X_{10}	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>Italian</i>	<i>10–30</i>	<i>F</i>
X_{11}	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>0–10</i>	<i>F</i>
X_{12}	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>30–60</i>	<i>T</i>

Using data to build a tree (1)



Let's examine how the attributes split up the examples

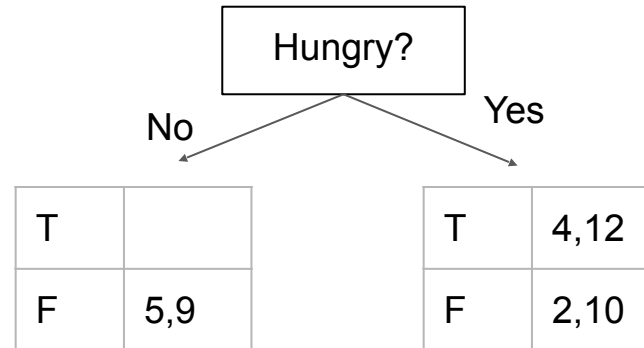
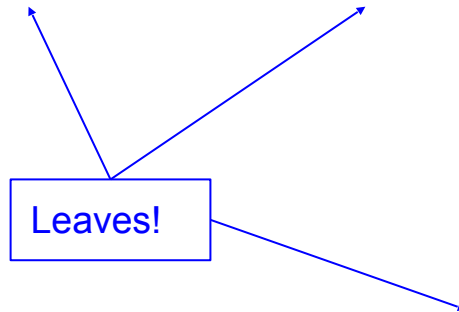
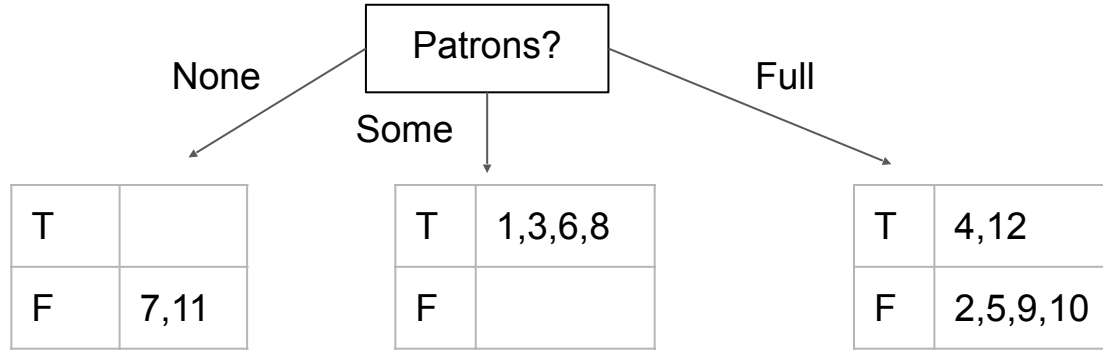


Not the best split

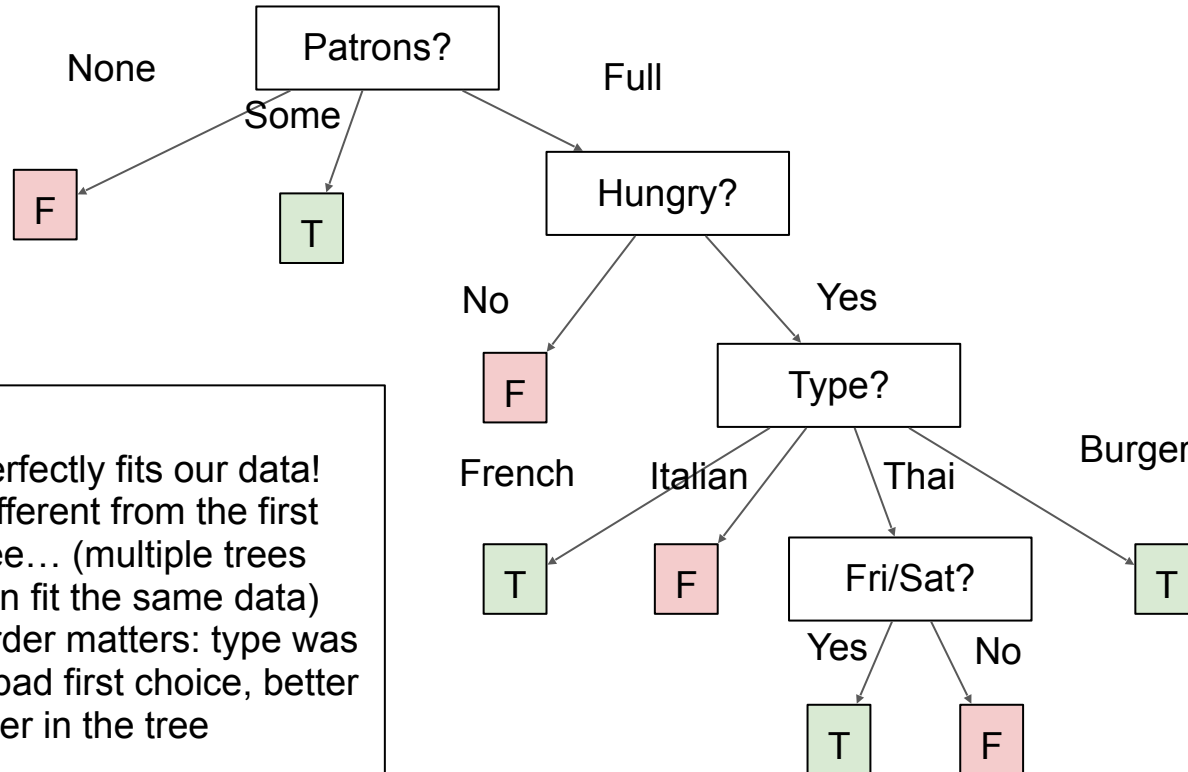
- Each branch has both T and F examples
- Try another attribute first

Using data to build a tree (2)

True	1, 3, 4, 6, 8, 12
False	2, 5, 7, 9, 10, 11



Final tree for the 12 examples



Notes

- Perfectly fits our data!
- Different from the first tree... (multiple trees can fit the same data)
- Order matters: type was a bad first choice, better later in the tree

Algorithm sketch for Decision Trees

Given Data S:

1. Pick “best” question Q, make a node N
2. Split S into subsets for each possible response to Q
3. Make children of N by recursing on each subset

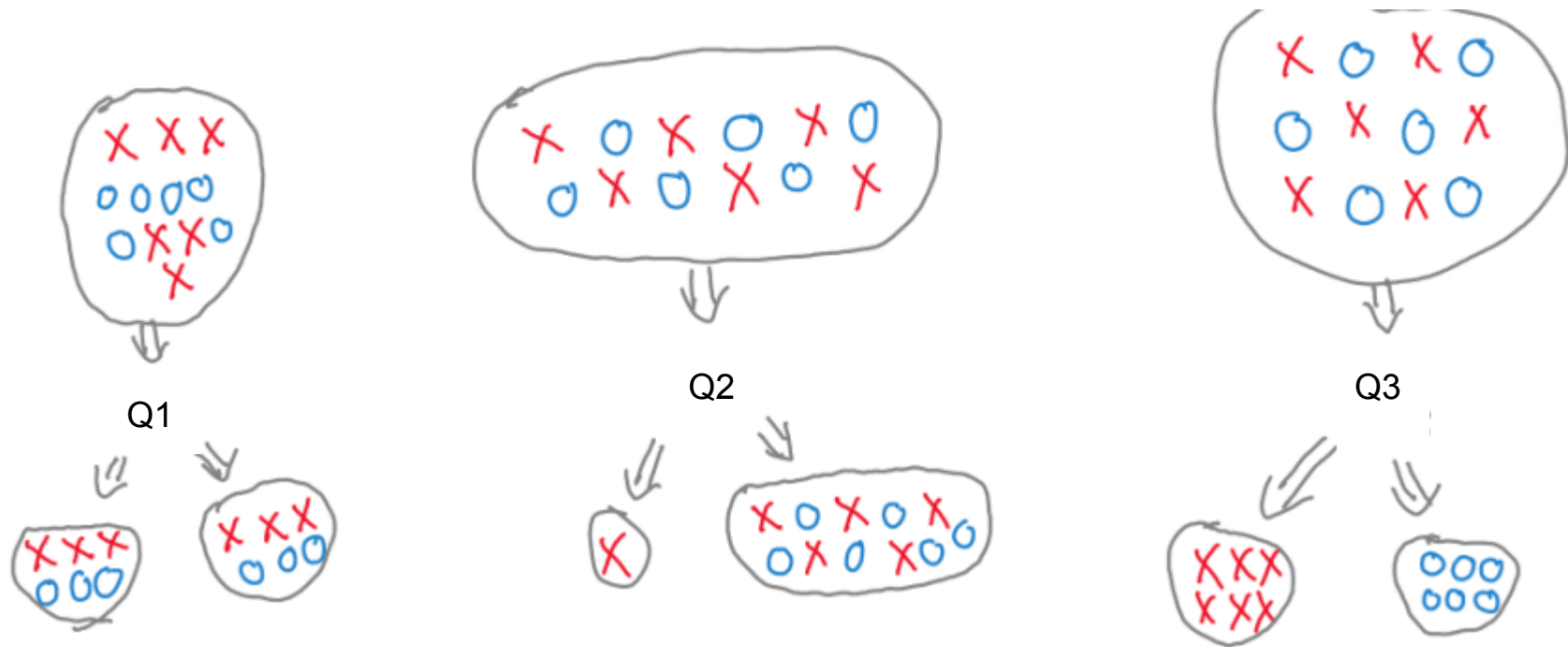
Base Cases

- Hit depth limit (leaf: majority label)
- All examples have the same label (leaf: that label)
- Subset is empty (leaf: majority label **of parent**)
- Run out of questions (leaf: majority label)

How do we define “best” question?

Defining what makes a good question (1)

Which of these three splits is “best”?



Defining what makes a good question (2)

- Q1 subdivided the examples into sets with equal proportions of both labels (have we learned anything?)
- Q2 split off a single example
- Q3 took a completely **disordered** set of examples and divided them perfectly by class

How do we measure **disorder**?

Entropy:

$$H(S) = - \sum_{y_i} p(y_i) \log p(y_i)$$

Base 2

Possible labels

$p(y_i) = (\text{\# examples labeled } y_i) / (\text{\# examples})$

Entropy example: flipping a coin

- Fair coin: $p(\text{heads}) = p(\text{tails}) = 0.5$

$$H(\text{fair coin}) = -(0.5) \log (0.5) - (0.5) \log (0.5) = (-0.5)(-1) + (-0.5)(-1) = 1.0$$

- Biased coin: $p(\text{heads})=0.75$, $p(\text{tails})=0.25$

$$H(\text{biased coin}) = -(0.75) \log (0.75) - (0.25) \log (0.25) = 0.8112\dots$$

- Two-headed coin: $p(\text{heads})=1$, $p(\text{tails})=0$

$$H(2\text{-h coin}) = -(1) \log (1) - (0) \log (0) = 0$$

$x \log x \rightarrow 0 \text{ as } x \rightarrow 0$

Information Gain

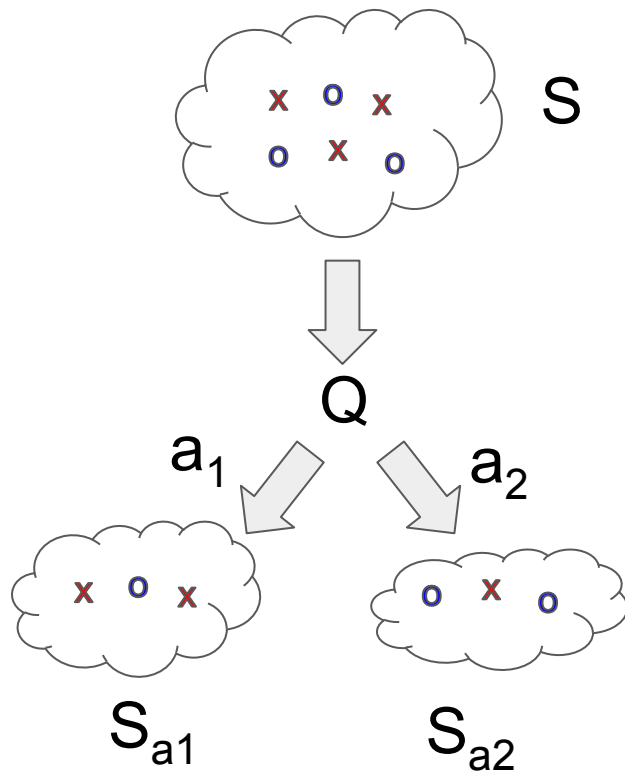
- We want to measure how much entropy a question **removes**
- Measure entropy before and after split, averaging over all branches

$$IG(Q, S) = H(S) - \overset{\text{"Remainder"}}{H(S | Q)}$$

$$H(S) = \sum_{y_i} p(y_i) \log p(y_i)$$

$$H(S | Q) = \sum_{a_i} \frac{|S_{a_i}|}{|S|} H(S_{a_i})$$

S_{a_i} = Subset of S that matches a_i



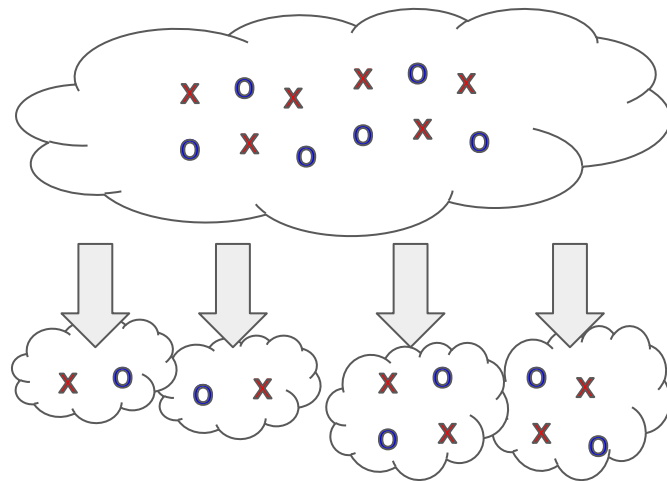
Restaurant Example: “Type?”

Before splitting: 6 True, 6 False

- $H(S) = -(0.5) \log(0.5) - (0.5) \log(0.5) = 1.0$

After splitting

- “French”: 1 True, 1 False
 - $H(S|Type=French) = 2/12 * (-0.5 * \log(0.5) - 0.5 * \log(0.5)) = 2/12$
- “Italian”: 1 True, 1 False, $H(S|Type=Italian) = 2/12$
- “Thai”: 2 True, 2 False, $H(S|Type=Thai) = 4/12$
- “Burger”: 2 True, 2 False, $H(S|Type=Burger) = 4/12$



$$\begin{aligned} \text{Remainder}(S, \text{Type?}) &= 2/12 + 2/12 + 4/12 + 4/12 = 1 \\ \text{Gain}(\text{Type?}) &= H(S) - \text{Remainder}(S, \text{Type}) = 1 - 1 = 0 \end{aligned}$$

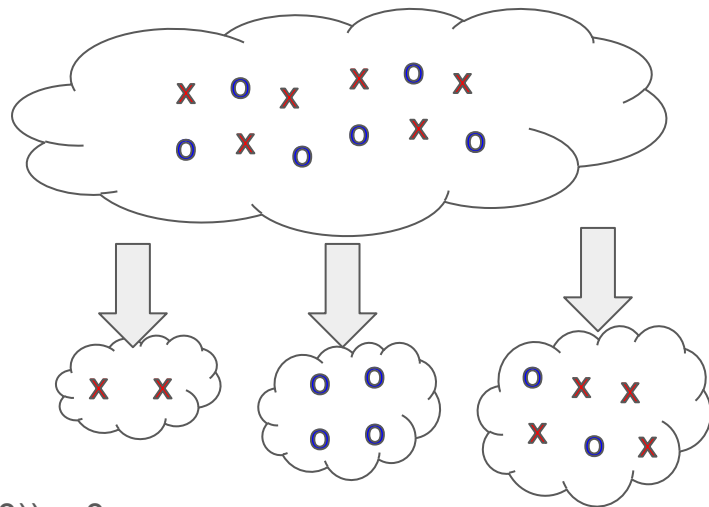
Restaurant Example: “Patrons?”

Before splitting: 6 True, 6 False

- $H(S) = -(0.5) \log(0.5) - (0.5) \log(0.5) = 1.0$

After splitting

- “Some”: 4 True, 0 False
 - $H(S|Patrons=Some) = 4/12 * (-1.0 * \log(1.0) - 0.0 * \log(0.0)) = 0$
- “None”: 0 True, 2 False, $H(S|Patrons=None) = 0$
- “Full”: 2 True, 4 False, $H(S|Type=Thai)$
 - $H(S|Patrons=Full) = 6/12 * ((-2/6) * \log(2/6) - (4/6) * \log(4/6)) = 0.459$



$$\begin{aligned} \text{Remainder}(S, \text{Patrons?}) &= 0 + 0 + 0.459 = 0.459 \\ \text{Gain}(\text{Patrons?}) &= H(S) - \text{Remainder}(S, \text{Patrons}) = 1 - 0.459 = 0.541 \end{aligned}$$

DT learning algorithm revisited

decision-tree-learning(examples, questions, default_val):

1. IF examples is empty THEN RETURN leaf(default_val)
 2. ELSE IF all examples have same label THEN RETURN leaf(label)
 3. ELSE IF remaining questions is empty THEN RETURN leaf(majority-label(examples))
- ELSE

best_q = question with highest info-gain

node = new DT node with question best_q

subtree_default = majority-label(examples)

subtree_questions = questions without best_q

FOREACH “v” response to best_q DO:

subset = {element of examples where best_q(example)=v}

subtree = decision-tree-learning(subset, subtree_questions, subtree_default)

add branch to node for v pointing to subtree

return node

Base cases (make a leaf)

1. Empty subset: leaf value = most common label of parent
2. All one label: leaf value = that label
3. No more questions to ask: leaf value = most common label of subset

recursion



Complexity of Decision Trees

Restriction: binary inputs, binary outputs

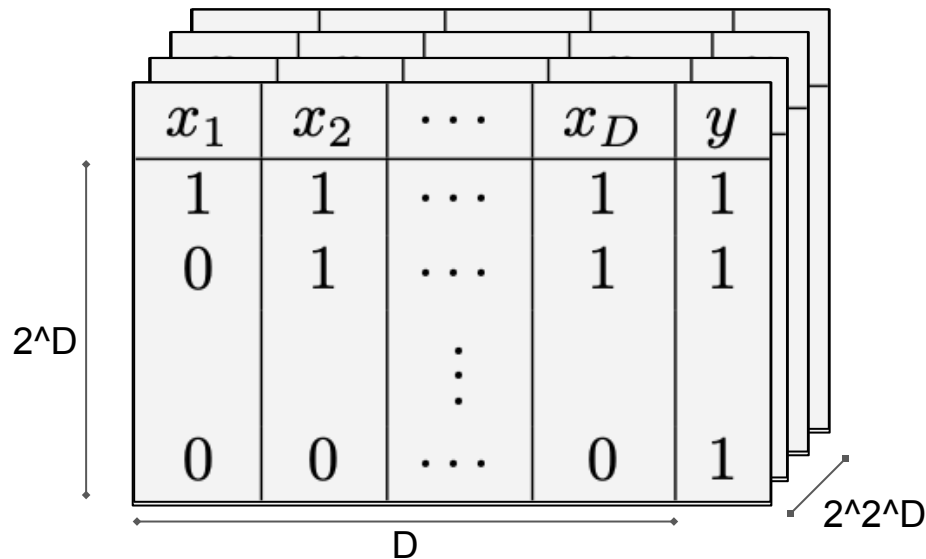
$$\mathcal{X} = \{0, 1\}^D$$

$$\mathcal{Y} = \{0, 1\}$$

Can represent functions as truth tables

# columns	D
# distinct inputs	2^D
# possible tables(/trees)	2^{2^D}

Hypothesis class is quite complex! How can we combat overfitting?



Handling Overfitting in DTs

Set a max depth

- 4th base case to terminate recursion
- Shallow trees represent less complex hypotheses
- Easy to code, ignores the data

Maybe there's a better way...
Boosting and Bagging, next time!

Pre-pruning

- If the best attribute has info-gain $<$ threshold, make a leaf instead
- Easy to code, requires tuning of threshold

Post-pruning

- Build the full tree
- Start at the bottom, merge leaves into parent if criteria met (CV error, statistical measures)

Summary and preview

Wrapping up

- Represent a decision process as a sequence of answers to questions →
Decision Tree
 - Nodes are questions, edges are answers, leaves are decisions/labels
 - How to classify a new point x ? Traverse the tree, following path according to x 's attributes
- When building DTs given data, **order matters**
- “Best” attributes give the most information (Info-Gain, Gini, Variance, etc.)
- Info-Gain = Entropy - Remainder
- DTs are expressive! Need to handle overfitting.

Next time: Ensemble methods