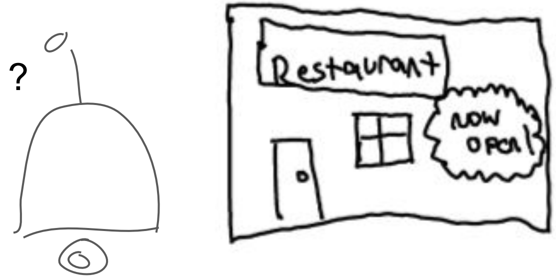
Decision Trees

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, Itallian, Thai, Burger)
- WaitEstimate: estimated time to get a table (0-10, 10-30, 30-60, >60)

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

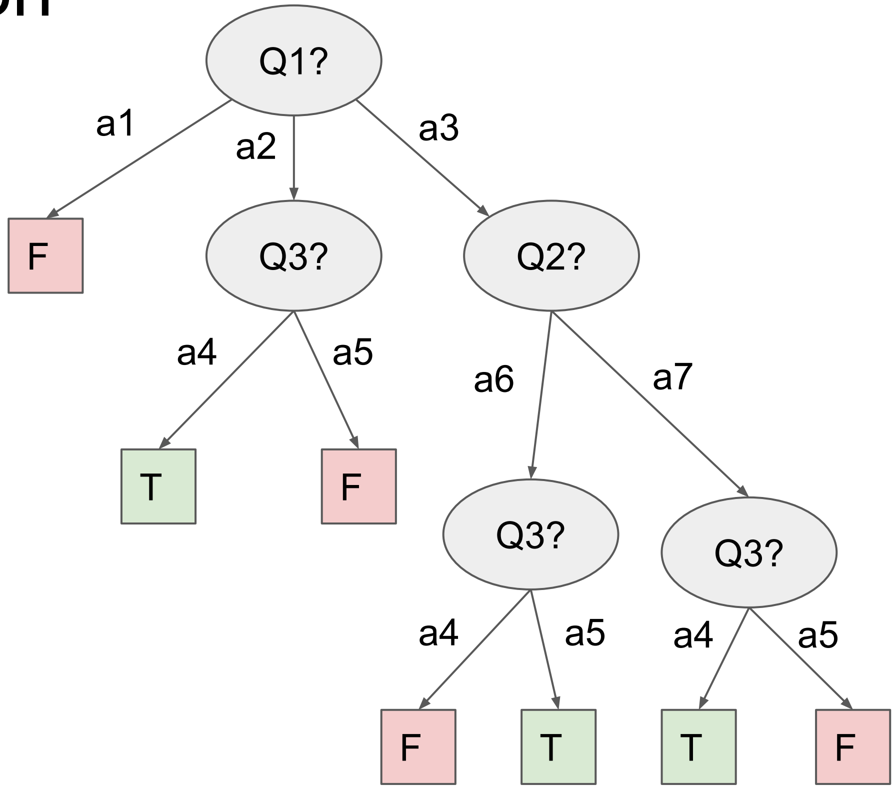
$$y = \mathrm{True}$$

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

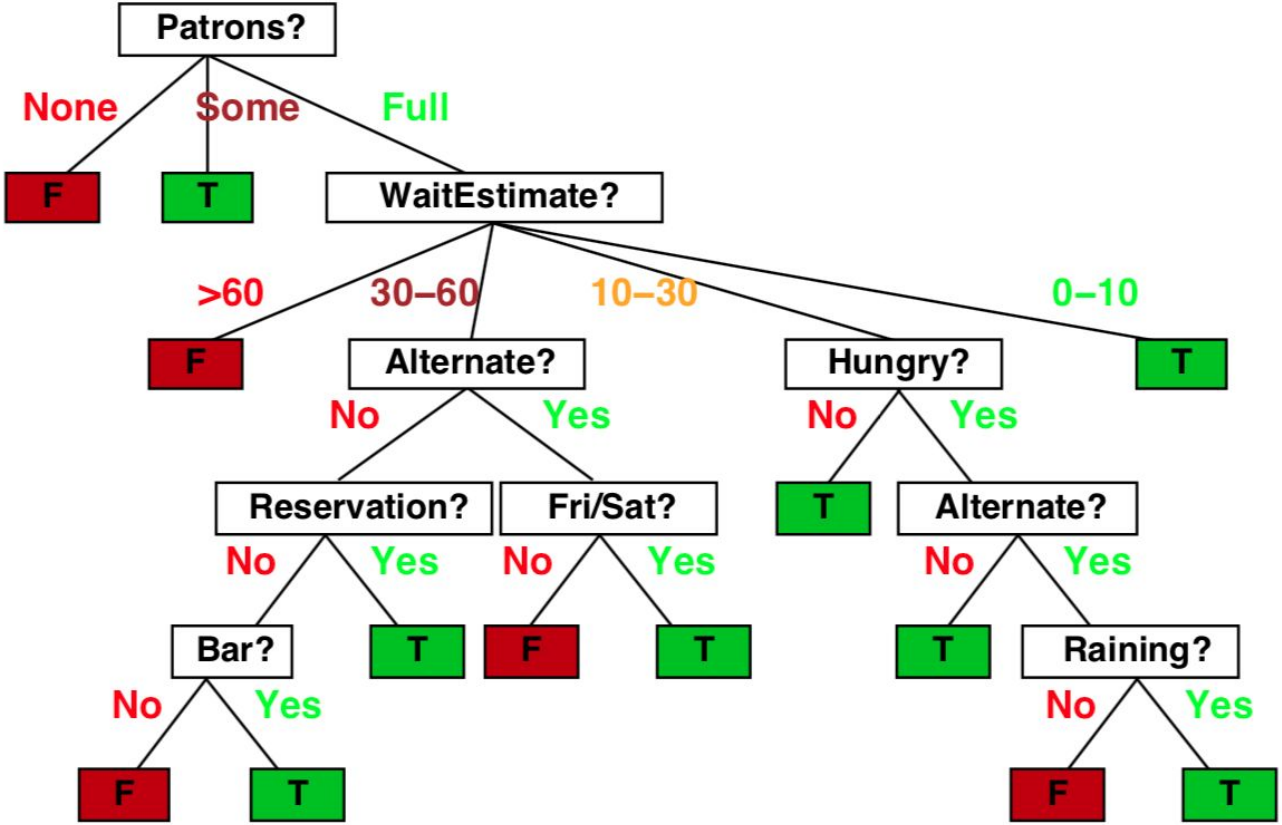
Decision Tree Representation

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

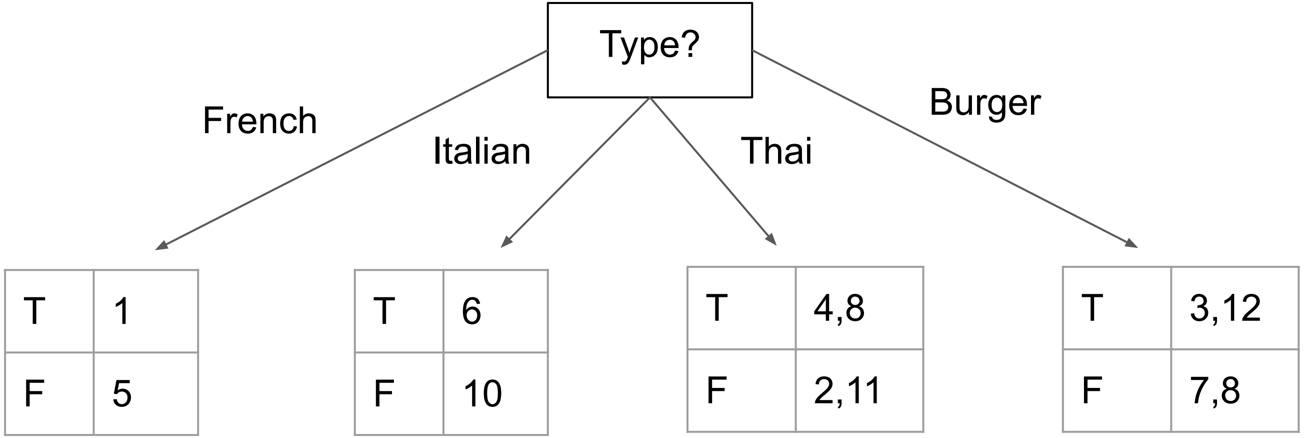
Using data to build a tree (1)

Label

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

Example #

Let’s examine how the attributes split up the examples

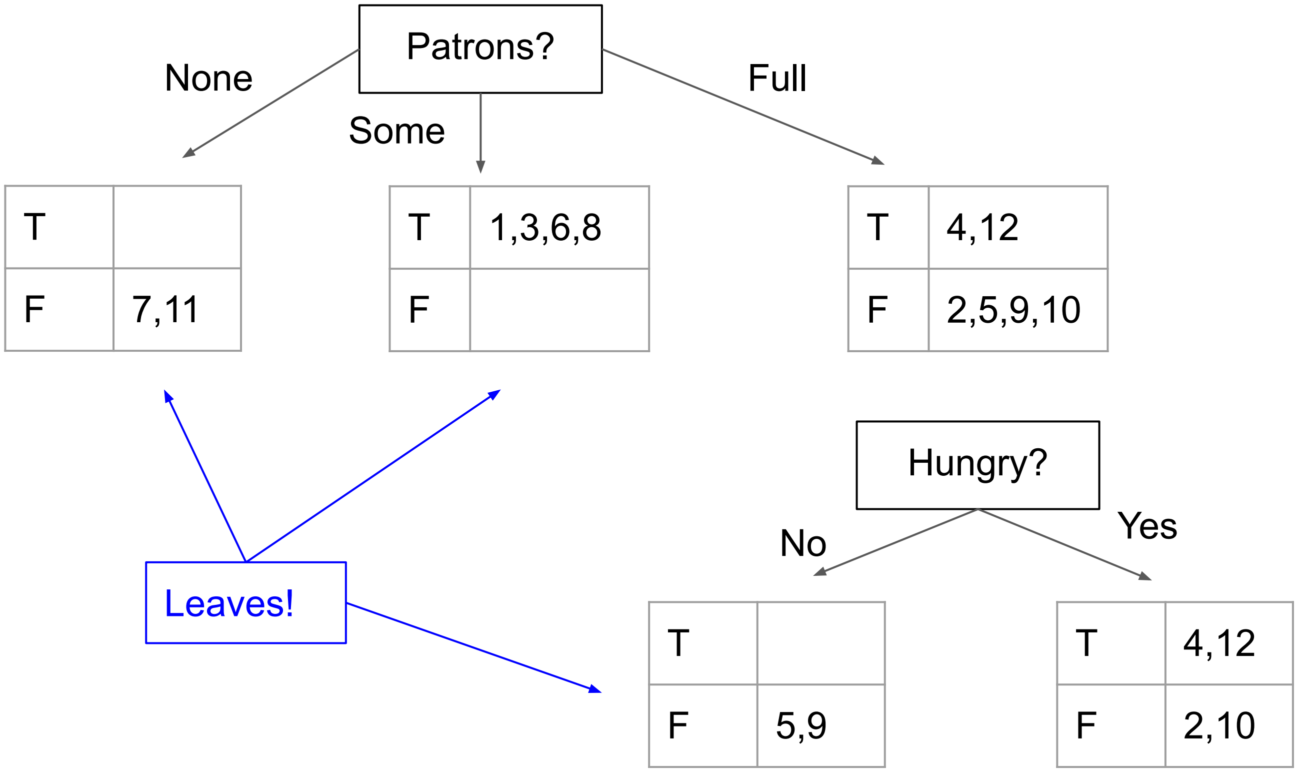


Not the best split

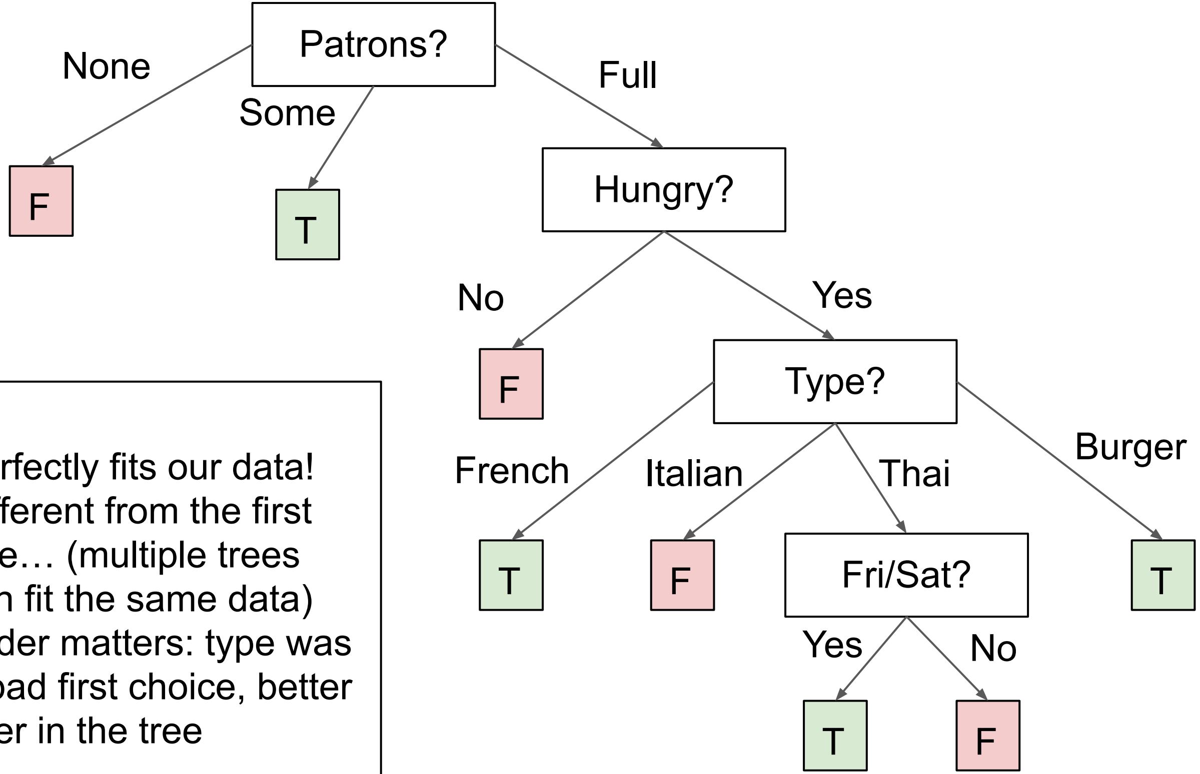
- Each branch has both T and F examples
- Try another attribute first

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

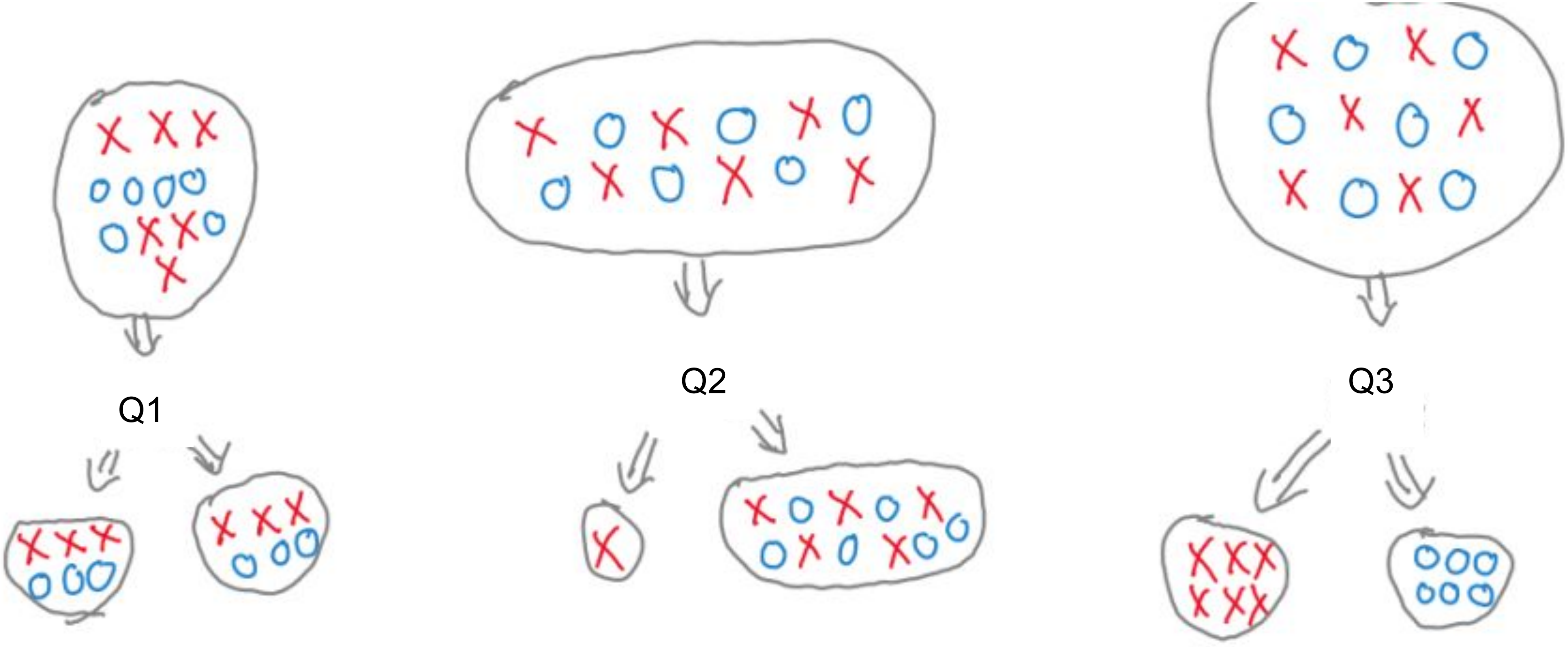
Base Cases

How do we define “best” question?

- 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)

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:

Base 2

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

Possible labels

$p(y_i)$ = (# examples labeled y_i)/(# examples)

Entropy example: flipping a coin

- Fair coin: p(heads) = p(tails) = 0.5
  H(fair coin) = -(0.5) log (0.5) - (0.5) log (0.5) = (-0.5)(-1) + (-0.5)(-1) = 1.0

- Biased coin: p(heads)=0.75, p(tails)=0.25
  H(biased coin) = -(0.75) log (0.75) - (0.25) log (0.25) = 0.8112…

- Two-headed coin: p(heads)=1, p(tails)=0
  H(2-h coin) = -(1) log (1) - (0) log (0) = 0

$x \log x \to 0$ as $x \to 0$

Information Gain

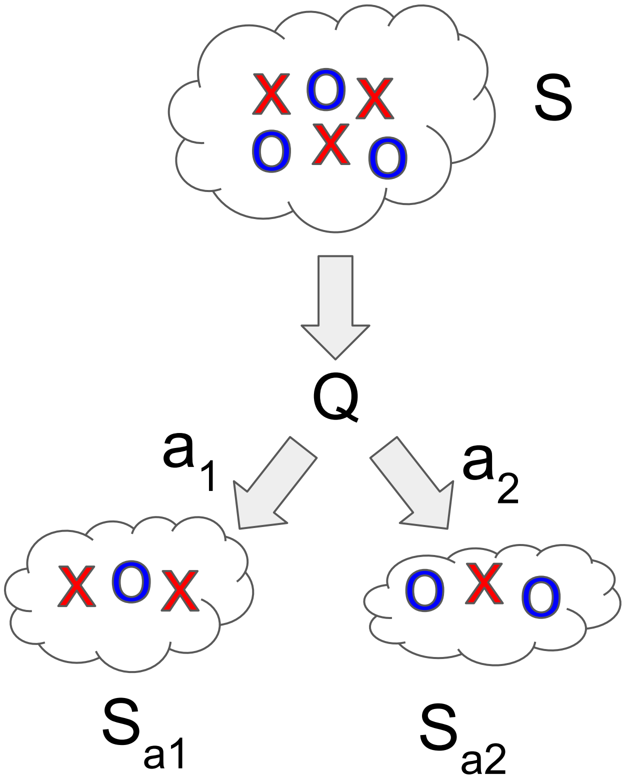
- We want to measure how much entropy a question **removes**
- Measure entropy before and after split, averaging over all branches

$$IG(Q,S) = H(S) - \boxed{\overset{\text{“Remainder”}}{H(S \mid Q)}}$$

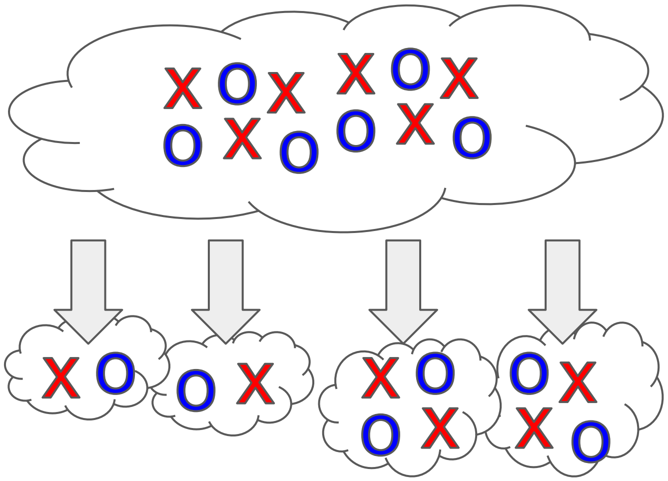
$$H(S) = \sum_{y_i} p(y_i) \log p(y_i)$$

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

$$S_{a_i} = \text{Subset of S that matches } a_i$$



Restaurant Example: “Type?”



Before splitting: 6 True, 6 False

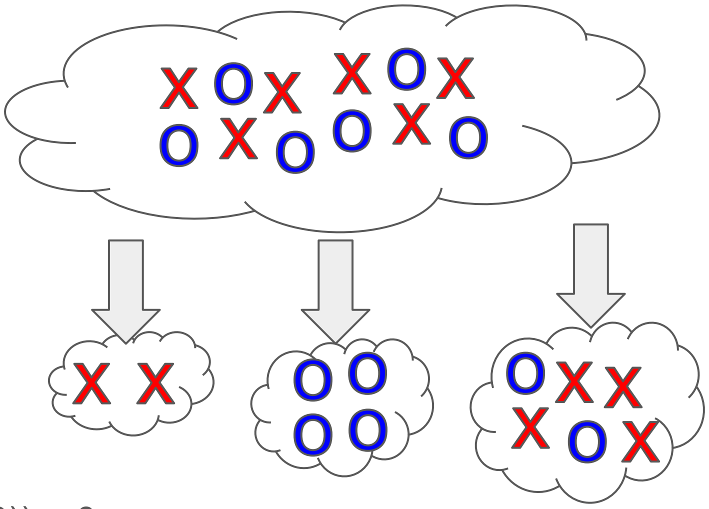
- H(S) = -(0.5) log (0.5) - (0.5) log (0.5) = 1.0

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

Remainder(S,Type?) = 2/12 + 2/12 + 4/12 + 4/12 = 1
Gain(Type?) = H(S) - Remainder(S,Type) = 1-1 = 0

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

Remainder(S,Patrons?) = 0 + 0 + 0.459 = 0.459
Gain(Patrons?) = H(S) - Remainder(S,Patrons) = 1-0.459 = 0.541

DT learning algorithm revisited

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

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
```

recursion

Complexity of Decision Trees

Restriction: binary inputs, binary outputs

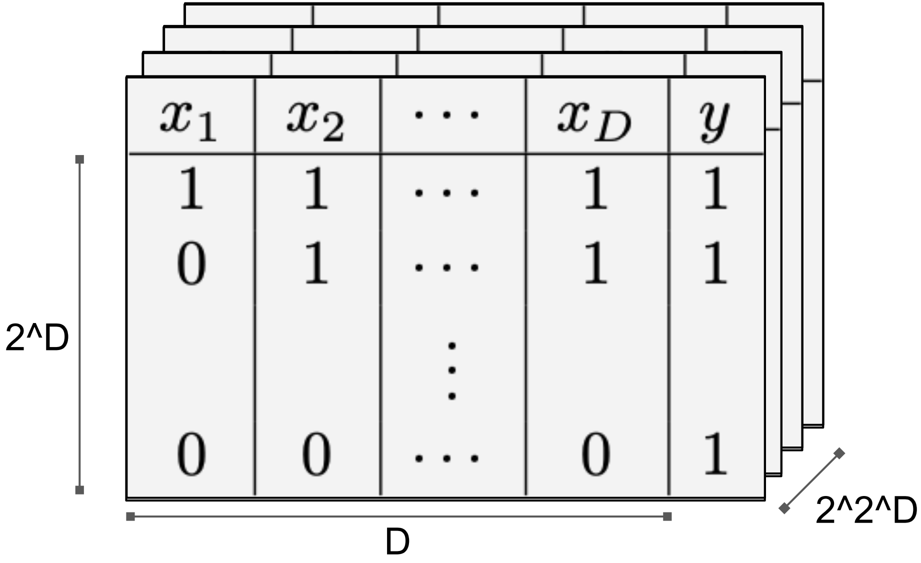
$$\mathcal{X} = \{0,1\}^D$$
$$\mathcal{Y} = \{0,1\}$$

Can represent functions as truth tables

columns: D

distinct inputs: 2^D

possible tables: 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