Decision Trees

CSC 442

Transition

- Last time: intro to machine learning
- This time: a supervised classifier — decision trees

Review: Machine Learning

- Key concepts:
  - Hypothesis Spaces, Supervised vs Unsupervised Learning, Interpolation vs Extrapolation, Generalization vs Overfitting, Regression vs Classification

Decision Tree Learning

- Suppose we have some **data** or **examples** which we would like to model or explain.
- Or we have a **decision problem** (i.e., **classification problem**) which we would like to automate.
- A **decision tree** is a learnable classifier which corresponds to a set of smaller decisions on individual attributes of the problem.

Decision Tree Learning

- Imagine you are going to dinner, but when you arrive at the restaurant you are told there is a wait to be seated.
- Do you wait for a table, or go somewhere else?
- Stuart Russel (AIMA author) provides the following summary of his preferences…

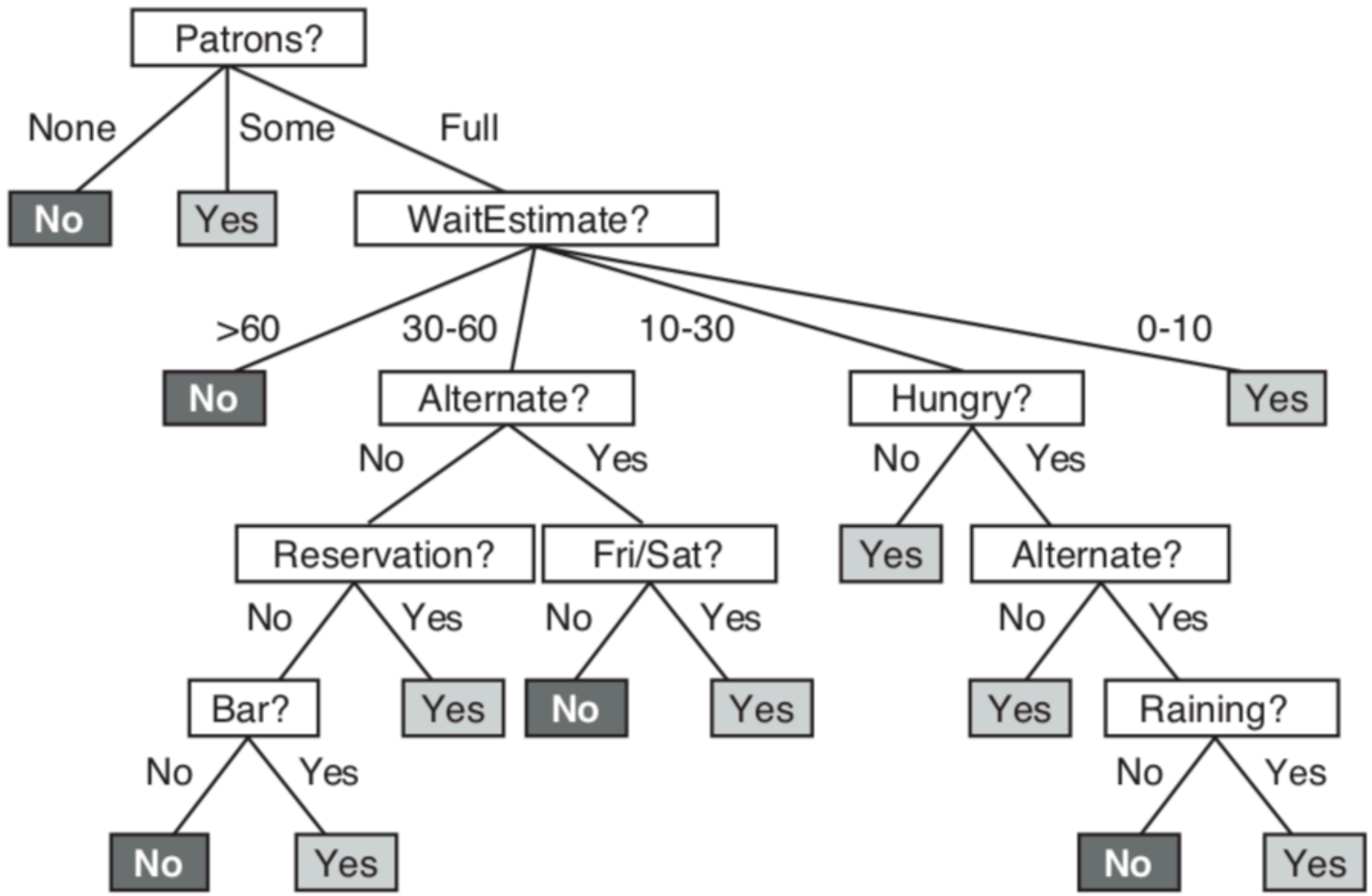


Figure 18.2 A decision tree for deciding whether to wait for a table.

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*

Figure 18.3 Examples for the restaurant domain.

We want to find a decision tree which is consistent with this dataset.

- A decision tree must have a root, so we should pick an initial attribute on which to test…
- maybe the type of restaurant?
- maybe the number of patrons?

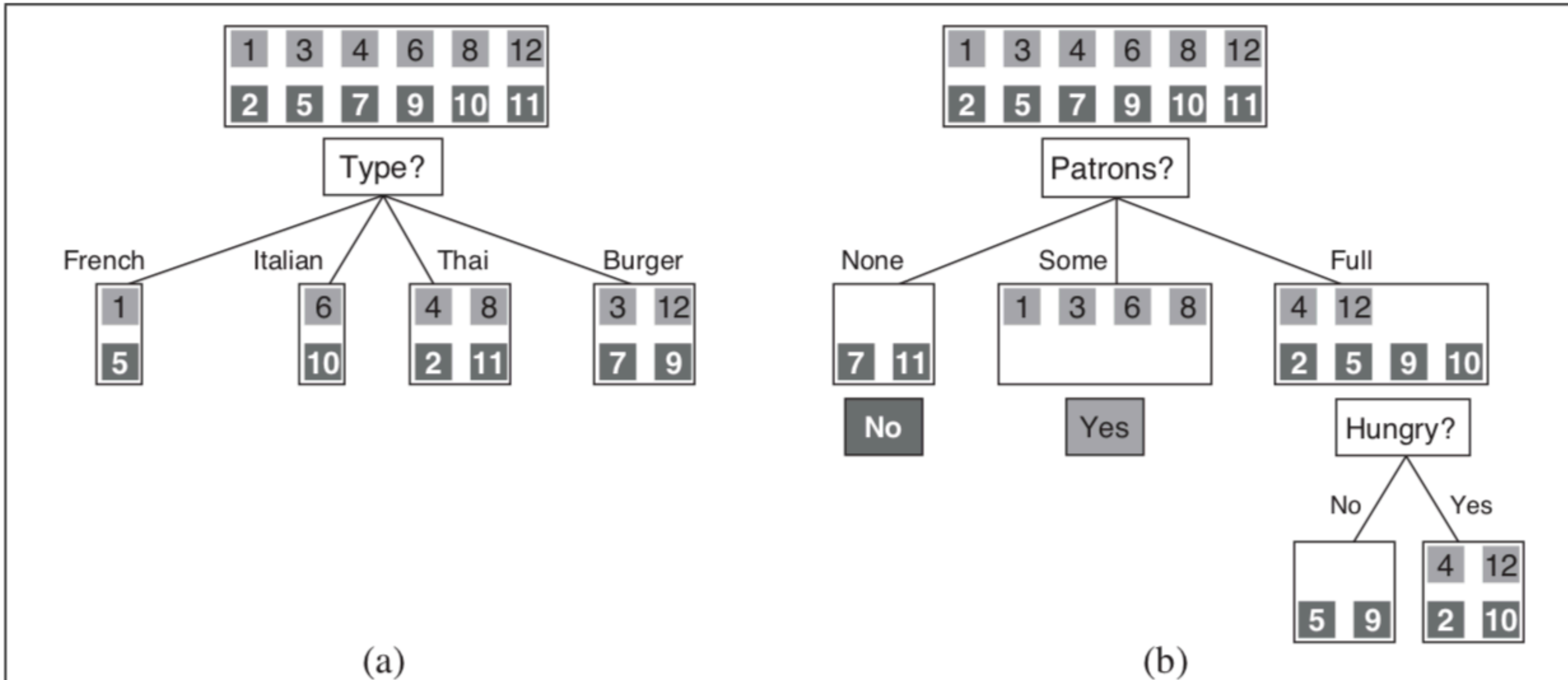


Figure 18.4 Splitting the examples by testing on attributes. At each node we show the positive (light boxes) and negative (dark boxes) examples remaining. (a) Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples. (b) Splitting on *Patrons* does a good job of separating positive and negative examples. After splitting on *Patrons*, *Hungry* is a fairly good second test.

What to do After Splitting?

- No more examples?
  Yield the most common result for the parent.

- All examples have the same label?
  Return that label.

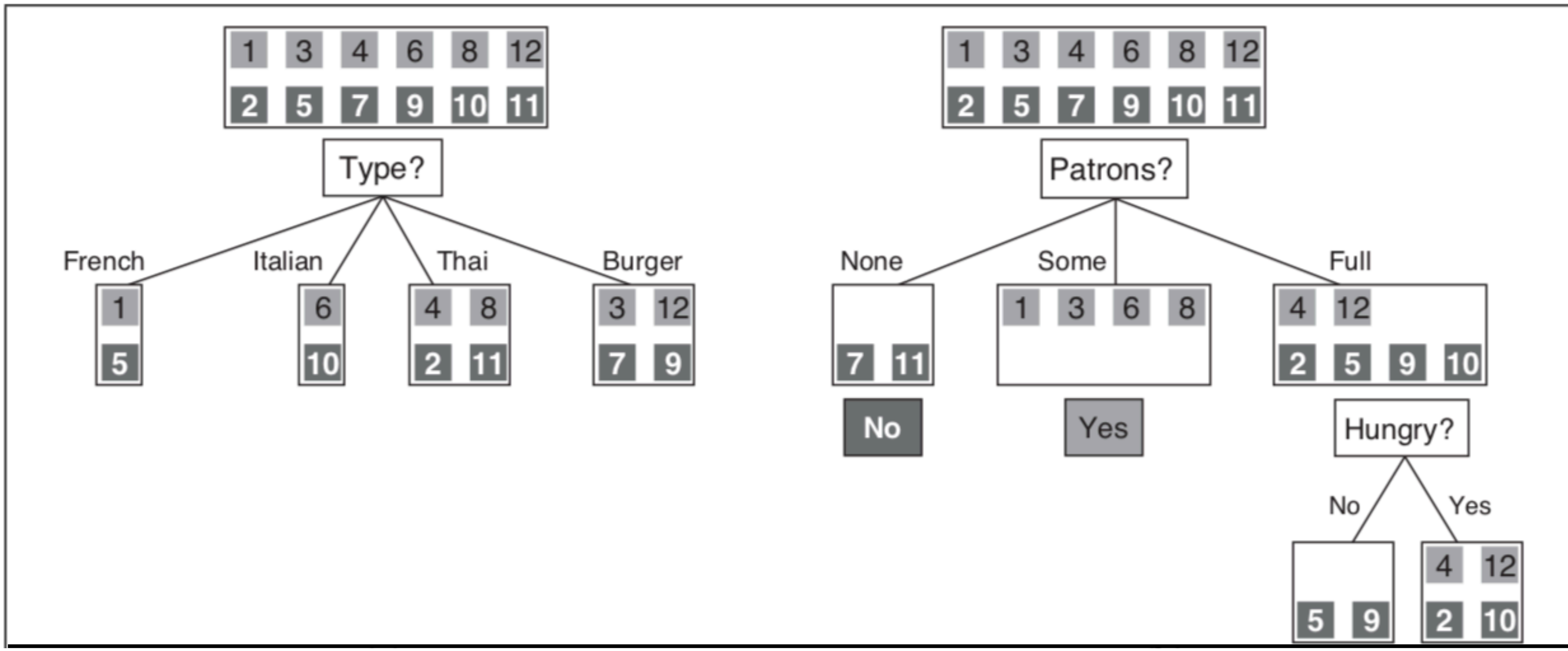
- Out of attributes?
  Return the most common label.

- In all other cases:
  Split the data again on a new attribute…

function DECISION-TREE-LEARNING($examples, attributes, parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE($a, examples$)
    $tree \leftarrow$ a new decision tree with root test A
    for each value v_k of A **do**
        $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
        $subtree \leftarrow$ DECISION-TREE-LEARNING($exs, attributes - A, examples$)
        add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
    return $tree$

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.



How do we quantify good vs bad attributes (for splitting)?

Importance

- As long as we use the same set of attributes, the order of attributes we test does not affect the accuracy of the decision tree.

- BUT, the order of the tests does affect the size of the tree. So, we want to choose the most "informative" attributes first. We want to choose the attribute which **reduces uncertainty** the most because then we have fewer examples left to classify, …

Entropy

- Entropy — the fundamental quantity of information theory.
  - Shannon and Weaver (1949)

Entropy

- Entropy is the measure of information gained by knowing the outcome of an event.
  - Entropy of a fair coin flip… 1 bit
  - Entropy of four-sided die… 2 bits
  - Entropy of loaded coin…. ?

Entropy

- The general formula for the entropy of a discrete random variable X is denoted by H(X) and has value:

$$H(X) = \sum_{x} Pr(X = x_k) \log_2 \frac{1}{Pr(X = x_k)}$$

$$= -\sum_{k} Pr(X = x_k) \log_2 Pr(x = x_k)$$

Entropy of a Fair Coin

$$H(X) = -\sum_k Pr(X = x_k) \log_2 Pr(x = x_k)$$

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Entropy of a Loaded Coin

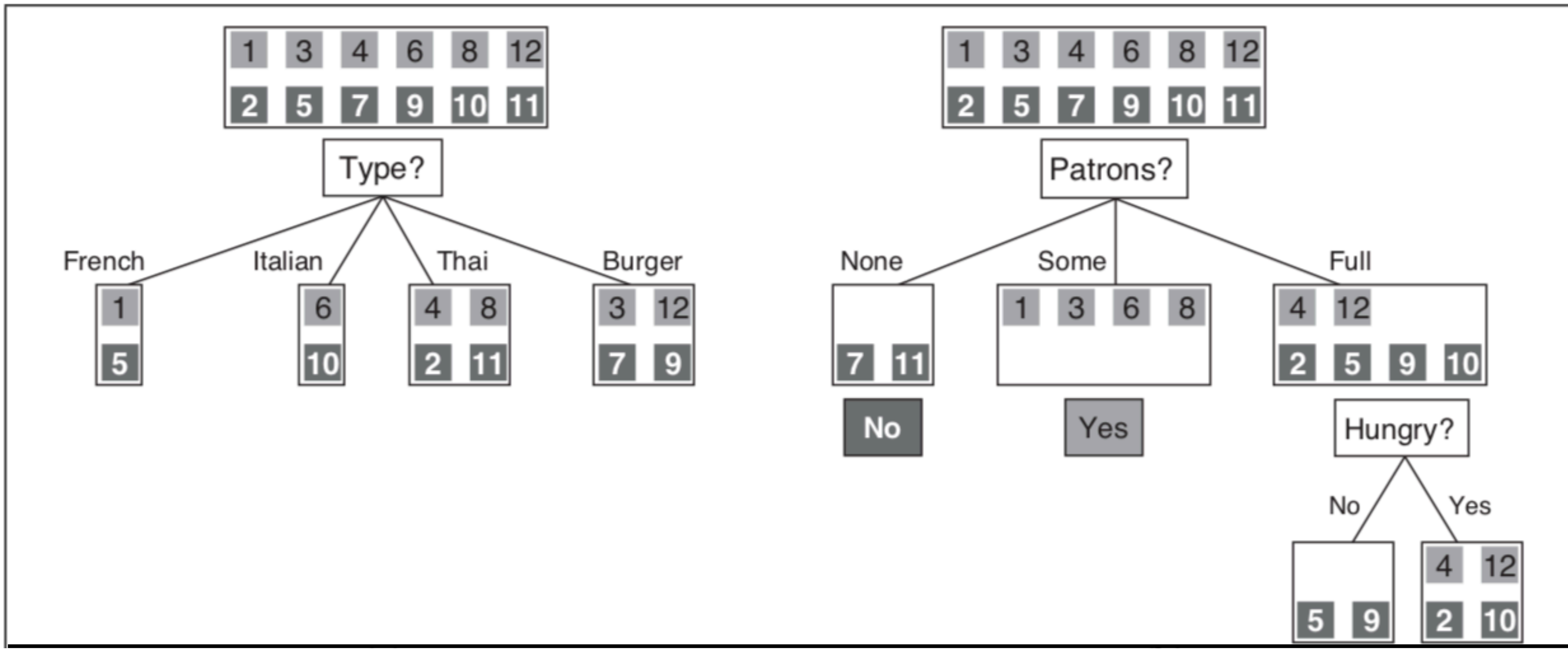
$$H(X) = -\sum_k Pr(X = x_k) \log_2 Pr(x = x_k)$$

$$H(Loaded) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) = 0.08$$

Entropy of a Two-Headed Coin

$$H(X) = -\sum_k Pr(X = x_k) \log_2 Pr(x = x_k)$$

$$H(Deterministic) = -(1 \log_2 1 + 0.00 \log_2 0.0) = 0$$



How do we quantify good vs bad attributes (for splitting)?

Decision Tree Learning (Attribute Selection)

- Let's try to choose attributes where there is as little remaining uncertainty as possible.

Decision Tree Learning (Attribute Selection)

- We need a supporting intermediate definition:

Information Gain

- We need a supporting definition, and to make some assumptions about the problem. First, the definition:

- Let $B(q)$ be the entropy of a boolean variable q, where $Pr(q = \text{true}) = q$. Then,

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

Information Gain

- Suppose at the current level of learning, we have p positive examples and n negative examples, and we are considering splitting on the attribute A.

- If A has d outcomes, then splitting on A would partition the remaining examples into sets $E_1 \ldots , E_d$.

- Suppose subset E_k has p_k positive examples and n_k negative examples….

- Then the *expected entropy* which remains after splitting on attribute A would be:

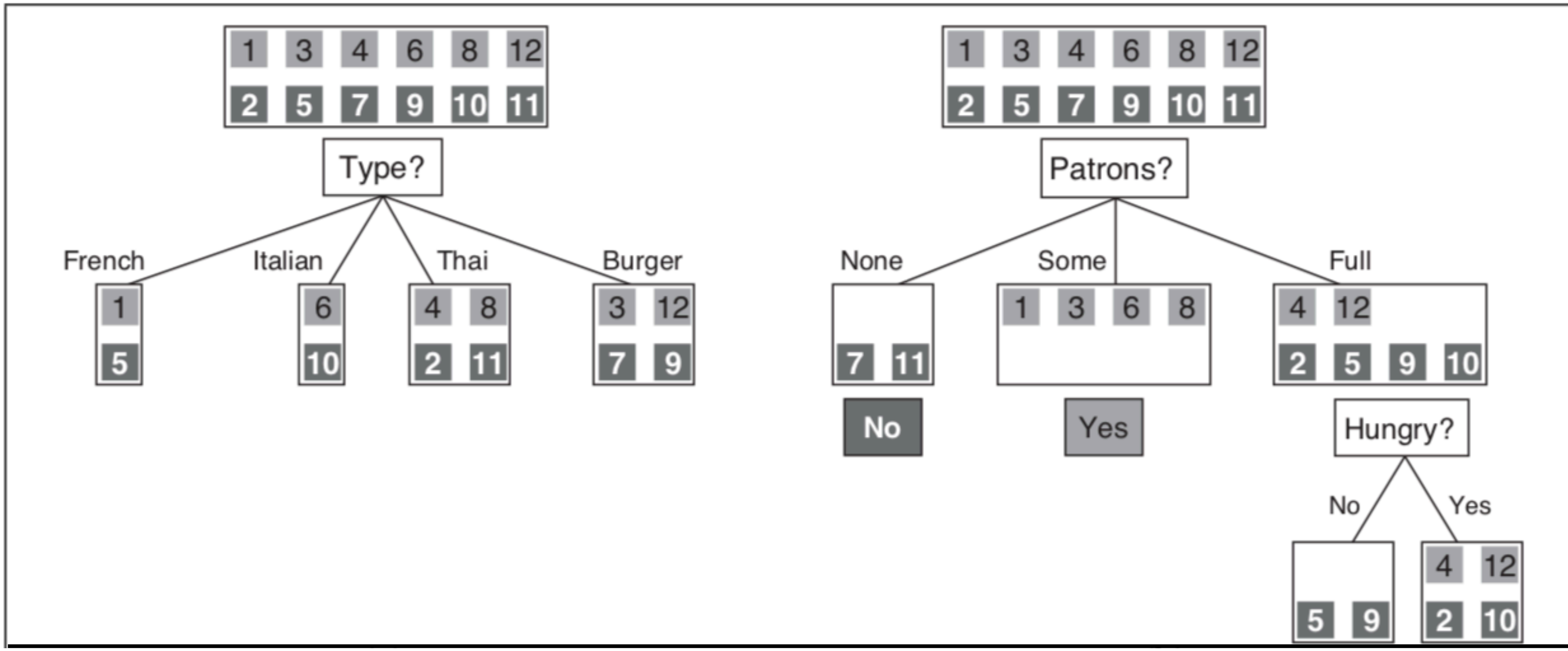
$$Rem(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

- And we can define the **information gain** as:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Rem(A)$$

$$Gain(Type) = 1 - \left[\tfrac{2}{12}B(\tfrac{1}{2}) + \tfrac{2}{12}B(\tfrac{1}{2}) + \tfrac{4}{12}B(\tfrac{2}{4}) + \tfrac{4}{12}B(\tfrac{2}{4})\right] = 0 \text{ bits}$$

$$Gain(Patrons) = 1 - \left[\tfrac{2}{12}B(\tfrac{0}{2}) + \tfrac{4}{12}B(\tfrac{4}{4}) + \tfrac{6}{12}B(\tfrac{2}{6})\right] \approx 0.541 \text{ bits}$$



How do we quantify good vs bad attributes (for splitting)?

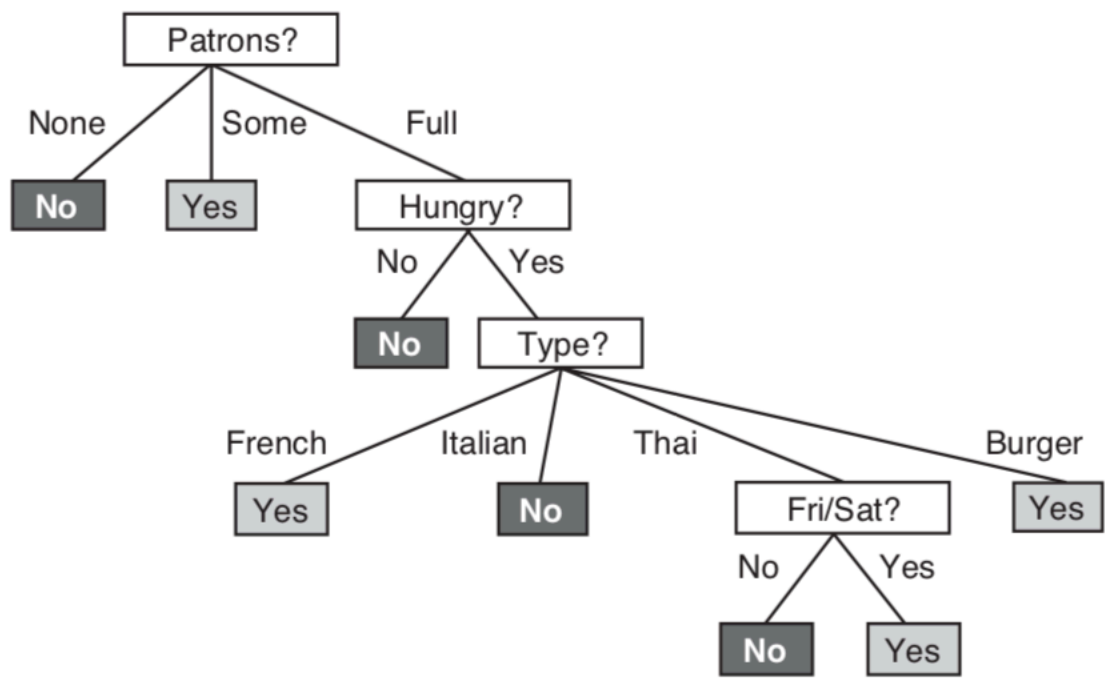


Figure 18.6 The decision tree induced from the 12-example training set.

A Learning Curve

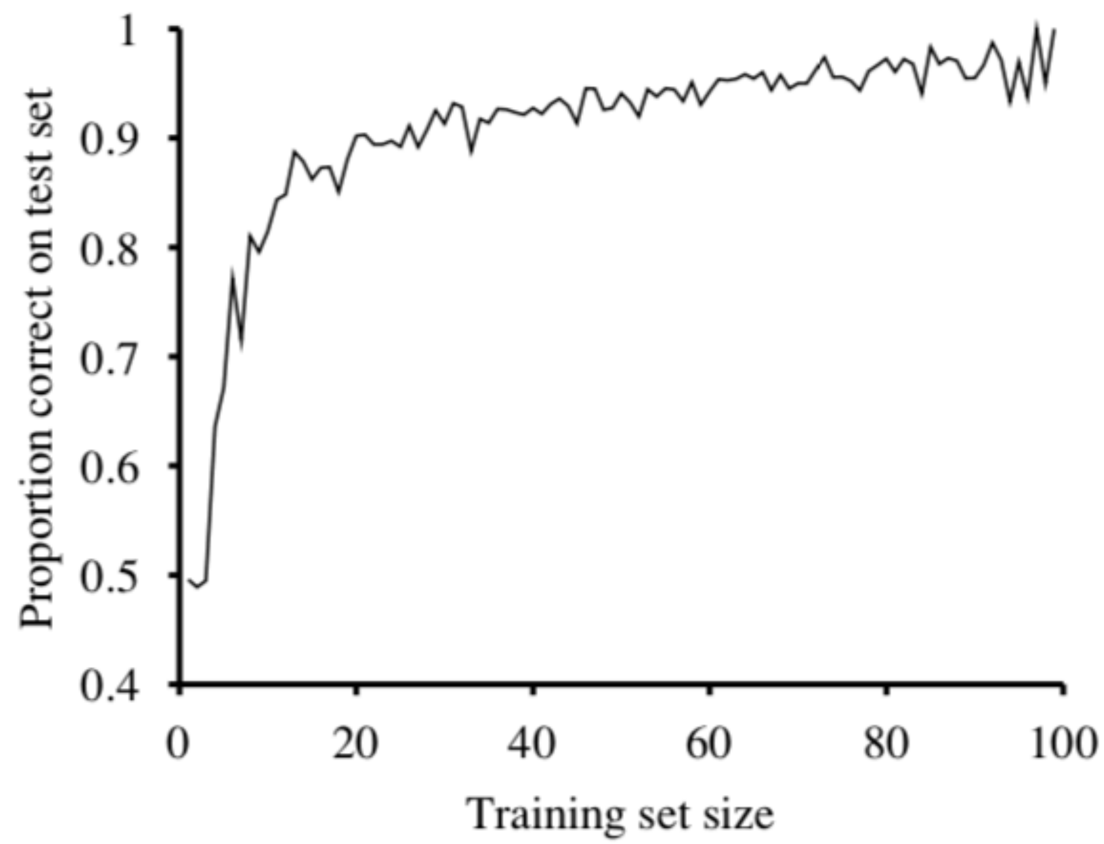


Figure 18.7 A learning curve for the decision tree learning algorithm on 100 randomly generated examples in the restaurant domain. Each data point is the average of 20 trials.

Decision Tree Pruning

- Start with a full tree.
- Consider the test nodes top to bottom.
- If a test is uninformative, replace that node with a leaf.

- Why not just stop when information gain is low?
  - The XOR problem.
- How do decide if a test is uninformative?
  - Statistical Significance Testing