

Chapter 18 (AIAMA)

Learning From Examples-02

Sukarna Barua

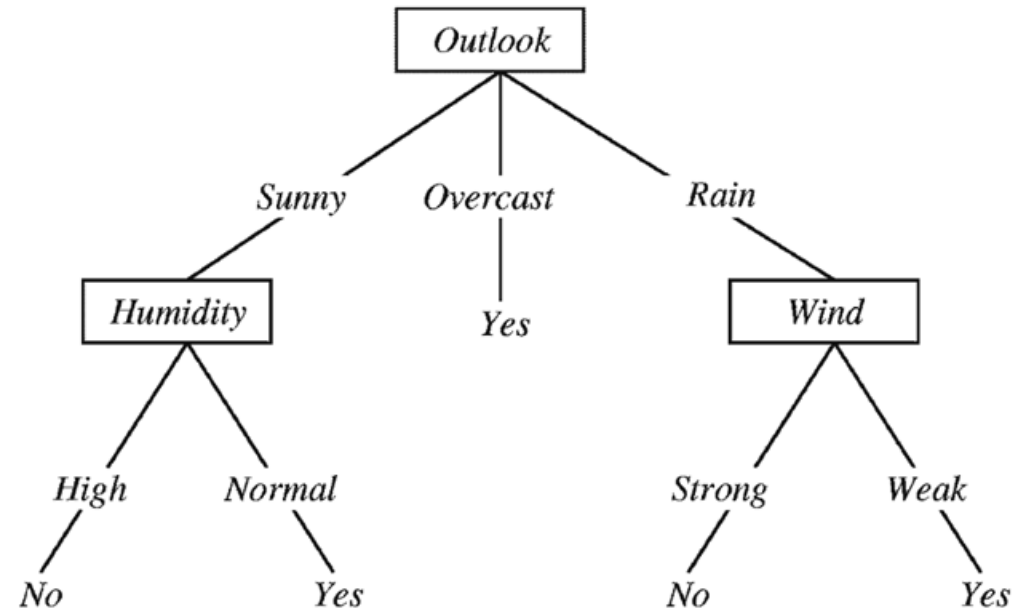
Associate Professor, CSE, BUET

Decision Tree

- **A decision tree:**
 - Represents a function that takes as input a vector of attribute values
 - Returns a “decision”—a single output value.
 - The input and output values can be discrete or continuous.
 - For now we will concentrate on problems where
 - Inputs have discrete values
 - Output has exactly two possible values; this is Boolean classification, where each example input will be classified as *true* (a **positive** example) or *false* (a **negative** example).

Decision Tree

- A decision tree for rain forecasting.
 - Input: {Outlook=Sunny, Humidity=Normal, Wind=Strong}
 - Output: Yes [Rainy]



Decision Tree

- **In a decision tree:**
 - Each **internal node** in the tree corresponds to a test of the value of one of the input attributes, A_i
 - The **branches** from the node are labeled with the possible values of the attribute, $A_i = v_{ik}$.
 - Each **leaf node** in the tree is marked with a outcome to be returned by the function.

Decision Tree: Example

- **Problem:** Build a decision tree to decide whether to wait at a restaurant.
- **Learning Goal:**
 - WillWait - to decide whether to wait for a table at a restaurant.
 - Goal is binary valued (i.e., binary classification task)
 - Values: {Yes, No}

Decision Tree: Example

- **Input 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, *false* otherwise.
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 (\$, \$\$, \$\$\$).

Decision Tree: Example

- **Input Attributes**

- 7. *Raining*: whether it is raining outside.

- 8. *Reservation*: whether we made a reservation.

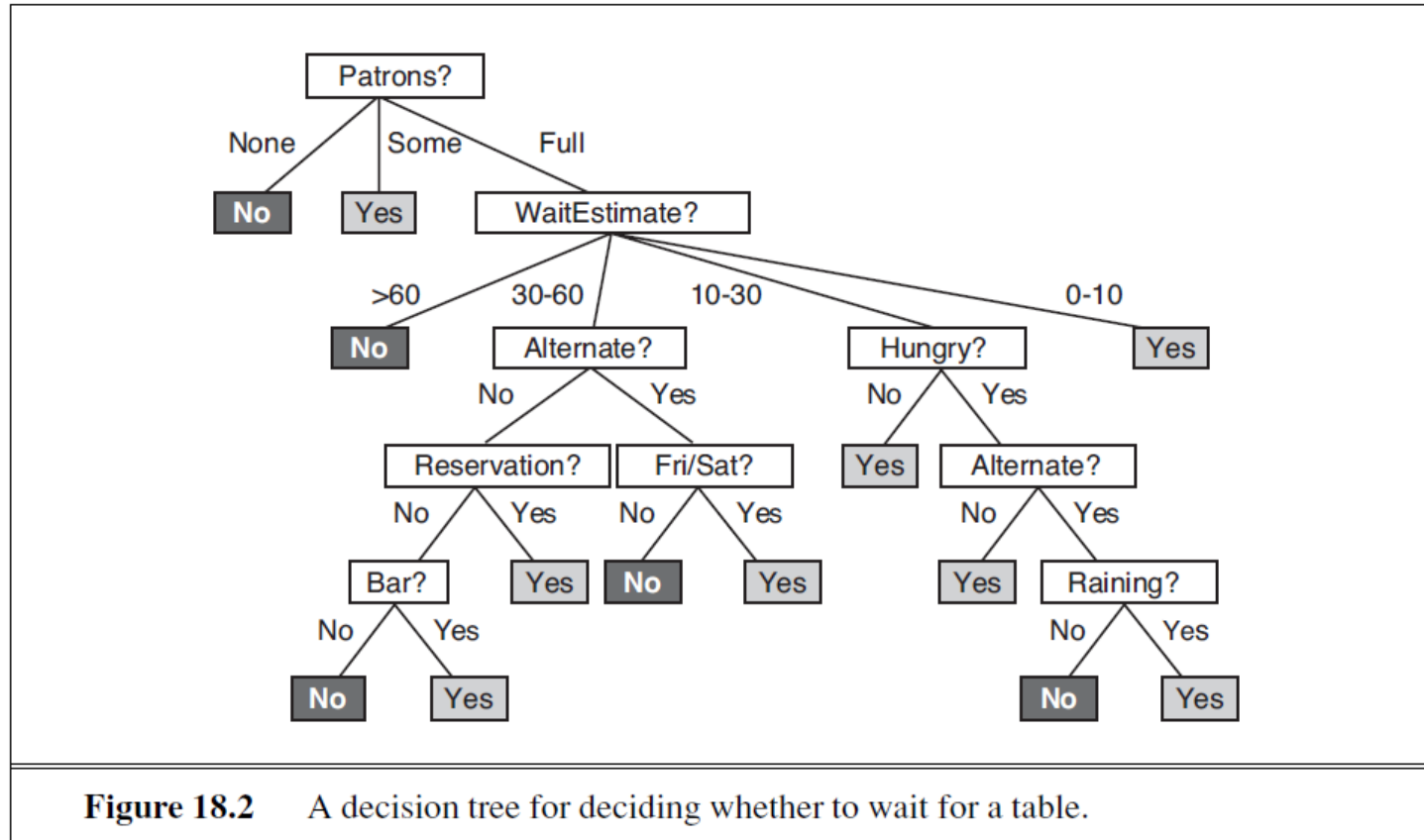
- 9. *Type*: the kind of restaurant (French, Italian, Thai, or burger).

- 10. *WaitEstimate*: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

- Note that every variable has a small set of possible values; the value of *WaitEstimate*, for example, is not an integer, rather it is one of the four discrete values 0–10, 10–30, 30–60, or >60.

Decision Tree: Example

- An example decision tree for the restaurant problem



Expressiveness of Decision Trees

- In a Boolean decision tree, the goal attribute is true if and only if the input attributes satisfy one of the paths leading to a leaf with value true.
- $Goal \Leftrightarrow (Path1 \vee Path2 \vee \dots)$, where each $Path$ is a conjunction of attribute-value tests required to follow that path.
- Thus, the whole expression is equivalent to *disjunctive normal form*, which means that any function in propositional logic can be expressed as a decision tree.

Expressiveness of Decision Trees

- Any Boolean function can be represented by a decision tree.
 - Consider the following Boolean function: $f(A, B) = A + B$.
 - Draw a decision tree for the function:

Expressiveness of Decision Trees

- For a wide variety of problems, the decision tree format yields a nice, concise result.
- But some functions cannot be represented concisely:
 - For example, the majority function, which returns true if and only if more than half of the inputs are true, requires an exponentially large decision tree.
- Decision trees are good for some kinds of functions and bad for others.

Expressiveness of Decision Trees

- **How many different decision trees can be obtained for a Boolean function with n variables?**
 - A truth table over n attributes has 2^n rows, one for each combination of values of the attributes.
 - There are 2^{2^n} different functions.
 - Each function can be represented by a decision tree. Hence, number of decision trees at least 2^{2^n} [why more?]

Expressiveness of Decision Trees

- **How many different decision trees can be obtained for a Boolean function with n variables?**
 - For 10 Boolean attributes of our restaurant problem there are 2^{1024} or about 10^{308} different functions to choose from.
 - Number of possible decision trees $\geq 10^{308}$

Inducing Decision Trees

- **Training data:** A set of examples where each example a pair (\underline{x}, y) pair where \underline{x} is a vector of values for the input attributes, y is a single Boolean output value.


Example	Input Attributes										Goal
	<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>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0-10</i>	$y_1 = \text{Yes}$
x_2	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30-60</i>	$y_2 = \text{No}$
x_3	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0-10</i>	$y_3 = \text{Yes}$
x_4	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10-30</i>	$y_4 = \text{Yes}$
x_5	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	$y_5 = \text{No}$
x_6	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0-10</i>	$y_6 = \text{Yes}$
x_7	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0-10</i>	$y_7 = \text{No}$
x_8	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0-10</i>	$y_8 = \text{Yes}$
x_9	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	$y_9 = \text{No}$
x_{10}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10-30</i>	$y_{10} = \text{No}$
x_{11}	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0-10</i>	$y_{11} = \text{No}$
x_{12}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30-60</i>	$y_{12} = \text{Yes}$

Figure 18.3 Examples for the restaurant domain.

Inducing Decision Trees

- ✓ ■ Build a decision tree that is -
 - ✓ Consistent with the examples [*Not always expected though, generalization may suffer*]
 - ✓ Is as small as possible. [*Occam's razor*]

Inducing Decision Trees

- **Question:** Can we always find a consistent decision tree given a set of training examples?.
- **Answer:** Yes.
 -  However, if two training examples (\mathbf{x}, y) and (\mathbf{x}', y') have different outputs but same input attributes ($\mathbf{x} = \mathbf{x}'$), then a consistent decision tree is not possible.

Inducing Decision Trees

- Unfortunately, it is an intractable problem to find the smallest consistent tree; there is no way to efficiently search through *hypotheses space*.
 - An NP-hard problem!
- What can we do if cannot find the smallest decision tree?
- **Solution:** Use heuristics to find a closest one.
 - With some simple heuristics, we can find a good approximate solution: a small (*but not the smallest*) consistent tree.
 - This is a greedy approach. [Remember what is *greedy*]

Decision Tree Learning

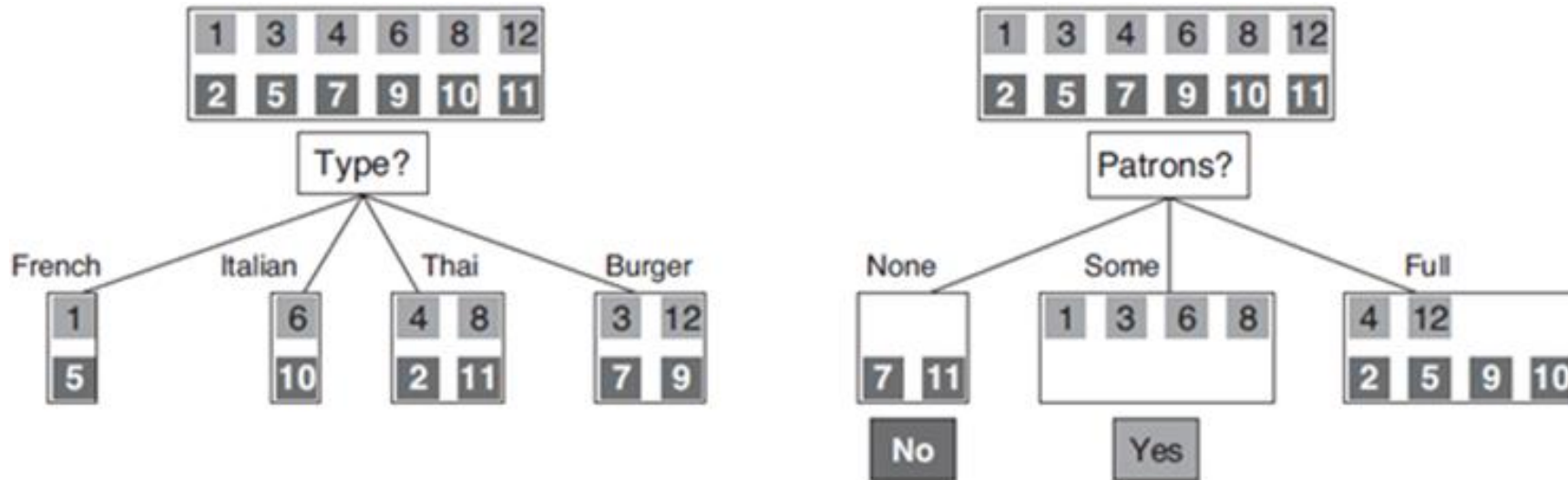
- **Greedy approach to build a decision tree:**
 - Start with empty decision tree.
 - Select an attribute to test at the next level [node]:
 - *Always select the most important attribute to test first. [greedy strategy]*
 - The test creates new branches and divides the problem into smaller subproblems.
 - Recurse on each child (created for each branch)

Decision Tree Learning

- **Greedy strategy:** *Always select the most important attribute to test first.*
 - *Most important attribute* implies the one that makes the most difference to the classification of an example. [*Get leaves as early as possible*]
 - Get correct classification with a small number of test
 - All paths in the tree will be short
 - Tree as a whole will be shallow.
 - *Above greedy strategy is a local optimal choice, may not necessarily leads to the globally smallest tree!*

Decision Tree Learning

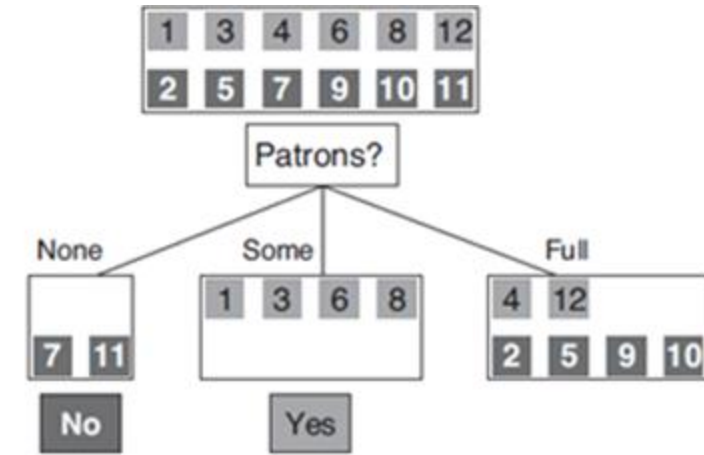
- Which attribute to test at root? Type vs Patron?
 - Type: all subsets (i.e., branches) needs further exploration.
 - Patron: two branches become leaves, only one needs further exploration.



Decision Tree Learning Algorithm

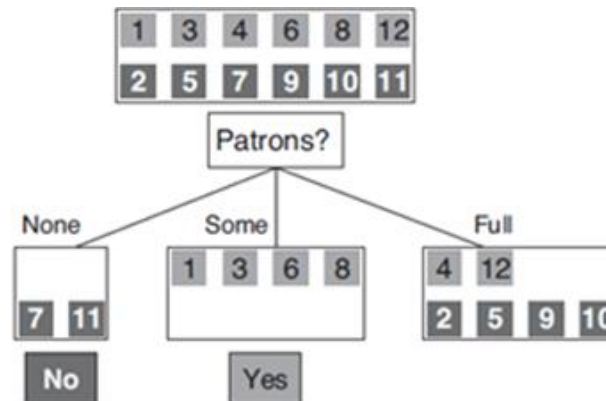
■ Decision tree construction:

- Step 1: Test an attribute at each node.
- Step 2: Partition the examples according to values and create child nodes with relevant examples.
- Step 3: Now consider child node for further tests of attributes except the one which have already been tested in the hierarchy (*recursive operation*)



Decision Tree Learning Algorithm

- **Four cases to consider at a child node:**
 - Case 1: All examples are positive (or all negative) → we are done; we can answer Yes or No. [e.g., *Patron=None*]
 - Case 2: Some positive and some negative examples → then choose the best attribute to split them recursively. [e.g., *Patron=Full*]



Decision Tree Learning Algorithm

- **Four cases to consider at a child node:**
 - Case 3: No examples left → No example has been observed for this combination of attribute values, and we return a default value (e.g., *plurality of parent node*)
 - Case 4: No attributes left, but both positive and negative examples remain → These examples have exactly the same description, but different classifications. This can happen because there is an error or **noise** in the data.
 - *In this case return the plurality classification of the remaining examples.*
[tree will not be consistent]

Decision Tree Learning Algorithm

- **Algorithm pseudocode**

```
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  
     $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$   
    tree  $\leftarrow$  a new decision tree with root test A  
    for each value  $v_k$  of A do  
       $\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v_k\}$   
      subtree  $\leftarrow$  DECISION-TREE-LEARNING(exs, attributes – A, examples)  
      add a branch to tree with label (A =  $v_k$ ) and subtree subtree  
  return tree
```

- Function PLURALITY-VALUE selects the most common class/output among the examples

Choosing the Most Important Attribute

- **Which attribute is the most important now?**
 - **Perfect attribute:** One that splits into subsets where each subset contain either all positive or all negative examples. [*all branches become leaf nodes*]
 - **Useless attribute:** One that splits into subsets where each subset contain fairly equal mix of positive and negative examples. [*all branches need recursive exploration*]

Choosing the Most Important Attribute

- **Perfect vs. useless? How to measure?**
 - A formal measure of perfect vs useless: **Entropy**
 - The fundamental quantity in information theory (*Shannon and Weaver, 1949*).
 - Entropy is a measure of the uncertainty of a random variable.
 - Acquisition of information corresponds to a reduction in entropy.
 - A random variable with only one value—a coin that always comes up heads—has no uncertainty and thus its entropy is defined as zero; thus, we gain no information by observing its value.

Choosing the Most Important Attribute

- **Entropy:** Average number of bits per symbol to encode information.
- The roll of a fair *four*-sided die has 2 bits of entropy, because it takes two bits to describe one of four equally probable choices.
- An unfair coin that comes up heads 99% of the time.
 - This coin has less uncertainty than the fair coin—if we guess heads we'll be wrong only 1% of the time—it's entropy measure should be close to zero, but positive.

Entropy Measure

- In general, the entropy $H(V)$ of a random variable V with values v_k , each with probability $P(v_k)$, is defined as :

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(v_k)$$

Entropy Measure

- Verify that entropies measures are correct.

We can check that the entropy of a fair coin flip is indeed 1 bit:

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

If the coin is loaded to give 99% heads, we get

$$H(Loaded) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08 \text{ bits.}$$

Entropy Measure

- Define $B(q)$ as the entropy of a Boolean random variable that is true with probability q :

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

- If a training set contains p positive examples and n negative examples, then the entropy of the goal attribute on the whole set is:

$$H(\textit{Goal}) = B\left(\frac{p}{p + n}\right) .$$

Entropy Measure in Decision Tree

- **Decision tree contest:** Entropy represents an impurity measure of the set.
 - A set with 5 positive and 5 negative examples: Most impure, entropy should be highest.
 - A set with 10 positive and 0 negative examples: Purest, entropy should be the lowest.

Entropy Before Split

- A set with p positive and n negative examples.

- Entropy of the set is: $B\left(\frac{p}{p+n}\right)$

Entropy After Splitting

- An attribute A with d distinct values divides the training set E into subsets E_1, E_2, \dots, E_d .
- Each subset E_k has p_k positive examples and n_k negative examples, with entropy of $B(p_k/(p_k + n_k))$ bits of information. [E_k contains examples with $A = kth$ value]
- A randomly chosen example from the training set has the kth value for the attribute with probability $(p_k + n_k)/(p + n)$, so the expected entropy (*weighted average*) of the d subsets after splitting on attribute A is:

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

Information Gain

- **Information Gain:** The amount of information gain is the amount of entropy reduction after the split on attribute A.
- Assume E_{BS} = Entropy before split, E_{AS} = Entropy after split [*weighted average*]
 - Hence, information gain (= *reduction of entropy*):

$$Gain = E_{BS} - E_{AS} \text{ [reduction of entropy]}$$

- Hence, this is simply:

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A)$$

Information Gain

- **Information Gain:** $Gain(A) = B(\frac{p}{p+n}) - Remainder(A)$
- Compute the information gain for the attributes Patron and Type:

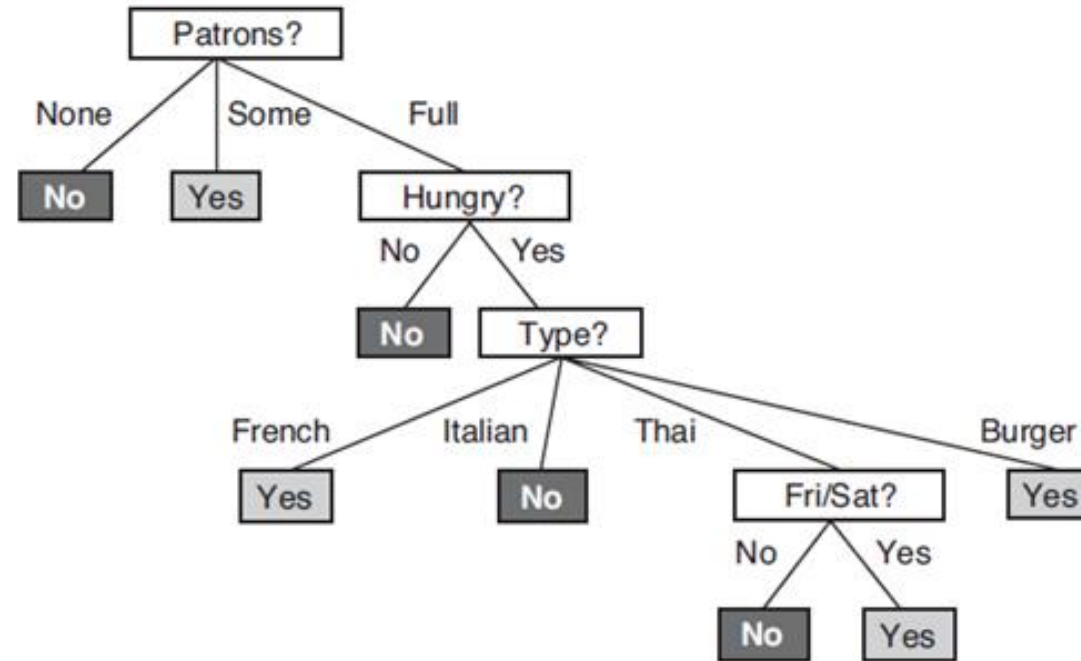
$$Gain(Patrons) = 1 - [\frac{2}{12}B(\frac{0}{2}) + \frac{4}{12}B(\frac{4}{4}) + \frac{6}{12}B(\frac{2}{6})] \approx 0.541 \text{ bits},$$

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

- Patron is a better attribute than Gain! Hence, choose Patron over Type.
- *Choose the attribute which gives the highest information gain!*

Final Decision Tree

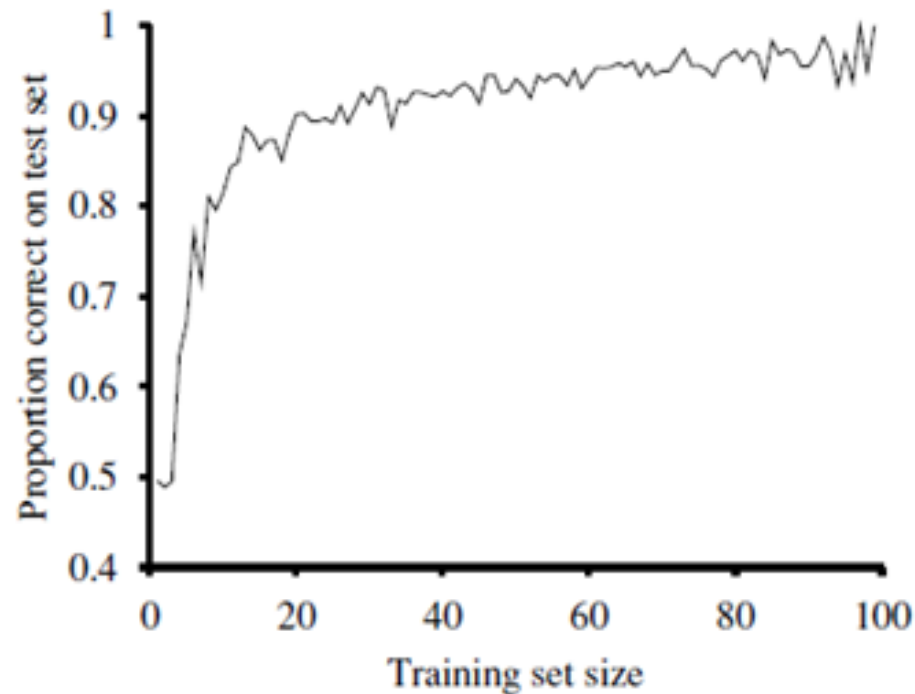
- Final decision tree constructed from given examples.



- Question:** What is the information gain for Type attribute at level 3?

Decision Tree Learning Curve

- Experiment with 100 examples: Construct decision trees
 - Split into train and test (e.g., 1 and 99, 2 and 98, etc.)
 - Random split 20 times and report average accuracy on test set
 - Note: As the training size grows, accuracy increases.



Overfitting

- **Overfitting:** Too much importance on every training example
 - Complex tree
 - Suffers generalization: very low training error, but very high test error.
 - Misses important concepts
- A consistent decision tree over training data may result in a complex tree
 - Noisy data may also induce complexity in the tree

Overfitting Solution

- **Pre-prune:** Prune the tree before it gets large
 - Early stopping: limit depth during tree construction
- **Post-pruning:** Prune the tree after construction
 - Remove irrelevant nodes
 - Replace internal nodes with most common class [*Only replace if test error do not increase*]

Decision Tree Issues

- **Missing values:** Some examples have missing value in some attributes
 - Solution: Replace missing value with mean of the attribute over the entire dataset
- **Continuous-valued attribute:** For example, height is continuous-valued.
 - Convert to categorical attribute:
 - Height > 40cm: *Tall* and Height <40cm: *Short*