Learning

Agents that can improve their behavior through diligent study of their own experiences.

LEARNING

An agent is learning if it improves its performance on future tasks after making observations about the world.

Why would we want an agent to learn?

If the design of the agent can be improved, why wouldn't the designers just program in that improvement to begin with?

- designers cannot anticipate all possible situations that the agent might find itself in
- designers cannot anticipate all changes over time
- sometimes human programmers have no idea how to program a solution themselves.

Learning

Any component of an agent can be improved by learning from data.

Which component is to be improved.

What prior knowledge the agent already has.

What representation is used for the data and the component.

What feedback is available to learn from.

Components to be learned

1. A direct mapping from conditions on the current state to actions.
2. A means to infer relevant properties of the world from the percept sequence.
3. Information about the way the world evolves and
   about the results of possible actions the agent can take.
4. Utility information indicating the desirability of world states.
5. Action-value information indicating the desirability of actions.

Learning

Representation and prior knowledge

propositional and first-order logical sentences for the components in a logical agent

Bayesian networks for the inferential components of a decision-theoretic agent

Learning from inputs that form a factored representation—a vector of attribute values—
and outputs that can be either a continuous numerical value or a discrete value.

Inductive learning

learning a (possibly incorrect) general function or rule from specific input–output pairs

Analytical or Deductive learning

going from a known general rule to a new rule that is logically entailed, but is useful because it allows more efficient processing.

Learning

Feedback to learn from

Unsupervised learning

the agent learns patterns in the input even though no explicit feedback is supplied.

The most common unsupervised learning task is clustering – detecting potentially useful clusters of input examples

Reinforcement learning

the agent learns from a series of reinforcements—rewards or punishments

Supervised learning

the agent observes some example input–output pairs and learns a function that maps from input to output.

Semi-supervised learning

given a few labeled examples, agent must make what it can of a large collection of unlabeled examples.

Supervised Learning

Given a training set of N example input–output pairs

(x1, y1), (x2, y2), . . . (xN, yN) ,

where each yj was generated by an unknown function y = f(x),

discover a function h that approximates the true function f.

Here x and y can be any value; they need not be numbers.
The function h is a hypothesis.

Learning is a search through the space of possible hypotheses for one that will perform well,

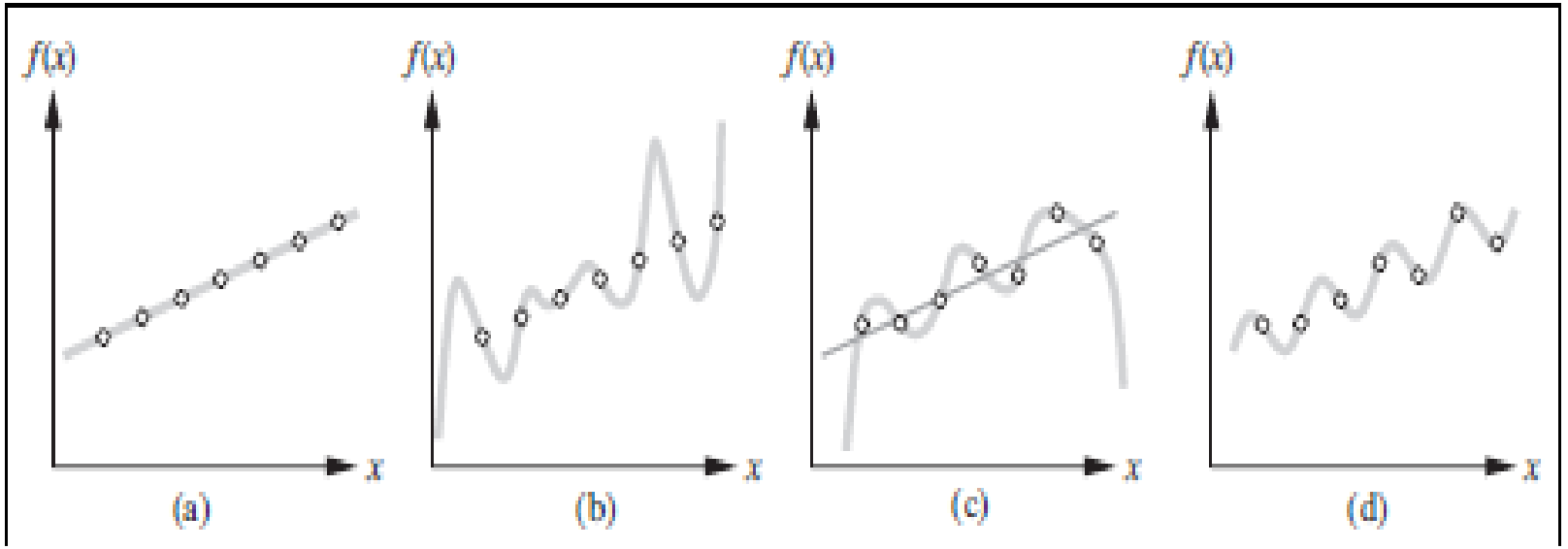
even on new examples beyond the training set.

To measure the accuracy of a hypothesis

a test set of examples that are distinct from the training set are used

We say a hypothesis generalizes well if it correctly predicts the value of y for novel examples.

Supervised Learning



fitting a function of a single variable to some data

(a) Example (x, f(x)) pairs and a consistent, linear hypothesis.

(b) A consistent, degree-7 polynomial hypothesis for the same data set.

(c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit.

(d) A simple, exact sinusoidal fit to the same data set.

Supervised Learning

CLASSIFICATION

When the output y is one of a finite set of values (such as sunny, cloudy or rainy), the learning problem is called classification,

Boolean or binary classification if there are only two values

REGRESSION

When y is a number (such as tomorrow's temperature), the learning problem is called regression.

solving a regression problem is finding a conditional expectation or average value of y

Example: fitting a function of a single variable to some data points:

The examples are points in the (x, y) plane, where y = f(x).

f is not known , but can be approximated with a function h selected from a hypothesis space, H.

H: set of polynomials, such as : x^5+3x^2+2; 0.4x + 3, …., ax + b + c sin(x) - choice of hypothesis space

A fundamental problem in inductive learning: how do we choose from among multiple consistent hypotheses?

Ockham's razor principle: chose the simplest hypothesis consistent with the data.

tradeoff between complex hypotheses that fit the training data well and simpler hypotheses that may generalize better

Decision Trees

Decision tree induction simplest of machine learning

decision tree representation

A decision tree represents a function that takes as **input a vector of attribute values** and **returns a "decision"—a single output value**.

Boolean classification:

problems where the inputs have discrete values and the output has exactly two possible values

each POSITIVE example input will be classified as true (a positive example) or false (a negative example).

A decision tree reaches its decision by performing a sequence of tests.

Each internal node in the tree corresponds to a test of the value of one of the input attributes, Ai,

the branches from the node are labeled with the possible values of the attribute, Ai =vik.

Each leaf node in the tree specifies a value to be returned by the function.

Decision Trees

Example

a decision tree to decide whether to wait for a table at a restaurant.

aim: learn a definition for the goal predicate: WillWait

Inputs - 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, or >60).

Decision Trees

Examples are processed by the tree starting at the root and following the appropriate branch until a leaf is reached.
example

with Patrons =Full and WaitEstimate =0–10 will be classified as positive

(i.e., yes, we will wait for a table)

A Boolean decision tree is logically equivalent to the assertion that

the goal attribute is true

if and only if the input attributes satisfy one of the paths leading to a leaf with value true.

in propositional logic, we have

$\text{Goal} \Leftrightarrow (\text{Path1} \vee \text{Path2} \vee \cdots)$,

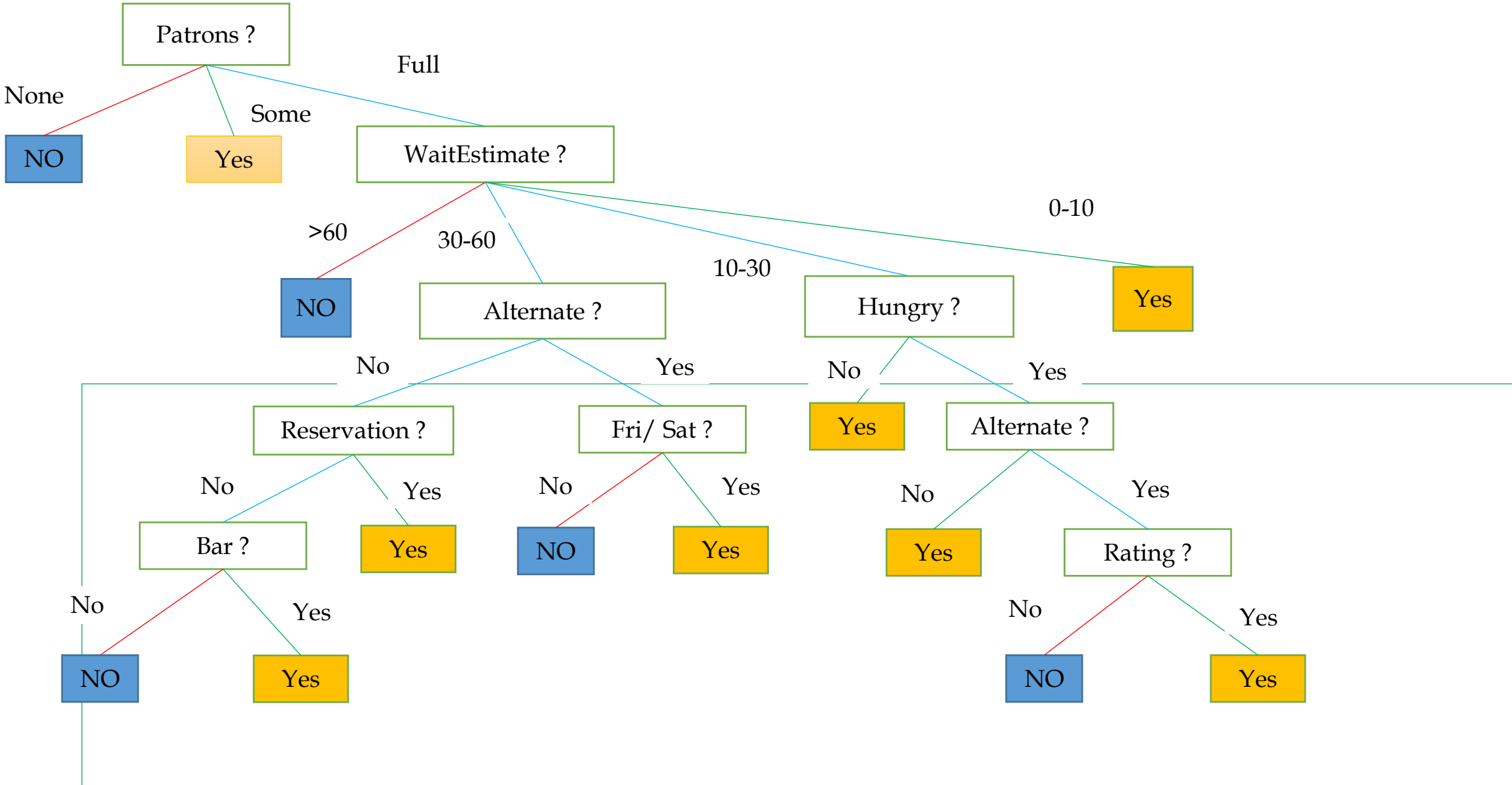
where each Path is a conjunction of attribute-value tests required to follow that path.

$\text{Path} = (\text{Patrons} = \text{Full} \wedge \text{WaitEstimate} = 0\text{–}10)$

the whole expression is equivalent to disjunctive normal form →

any function in propositional logic can be expressed as a decision tree.

Decision Tree



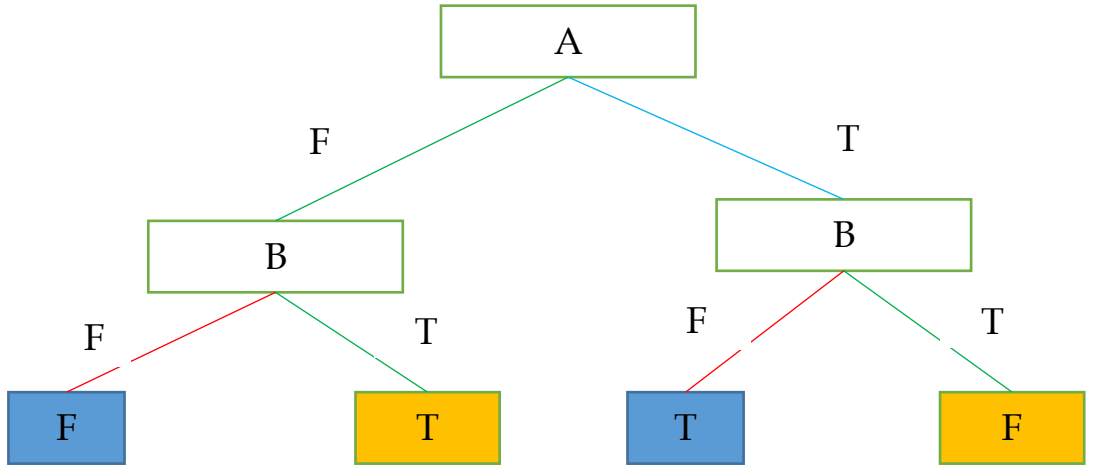
Expressiveness

Discrete-input, discrete-output case:

– Decision trees can express any function of the input attributes.

– E.g., for Boolean functions, truth table row → path to leaf:

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F



Decision Tree

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 bad for “additive threshold”-type functions such as

“at least 6 of the 10 inputs must be true,” but linear separators are good

Consider the set of all Boolean functions on n attributes.

How many different functions are in this set?

This is just the number of different truth tables that we can write down, because the function is defined by its truth table.

A truth table over n attributes has 2^n rows, one for each combination of values of the attributes.

We can consider the “answer” column of the table as a 2^n-bit number that defines the function.

That means there are 2pow(2^n)different functions

(and there will be more than that number of trees, since more than one tree can compute the same function).

Inducing decision trees from examples

How to construct meaningful Decision Trees?

example,

with just the ten Boolean attributes of restaurant problem there are 2^{1024} (= 2 pow(2^{10})) or about 10^{308} different functions to choose from,

And for 20 attributes there are over $10^{300,000}$

example for a Boolean decision tree: Restaurant - willwait

consists of an (x, y) pair, where x is a vector of values for the input attributes, and y is a single Boolean output value

A training set of 12 examples

Need a tree that is consistent with the examples and is as small as possible.

it is an intractable problem to find the smallest consistent tree;

there is no way to efficiently search through the 2pow(2^n) trees.

With some simple heuristics, however, we can find a good approximate solution:

Examples for the restaurant Domain

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*	$y_{12} = Yes$

positive examples are the ones in which the goal WillWait is true (x1, x3, . . .);
negative examples are the ones in which it is false (x2, x5, . . .).

DECISION-TREE-LEARNING algorithm

DECISION-TREE-LEARNING algorithm adopts a greedy divide-and-conquer strategy:
always test the most important attribute first.

This test divides the problem up into smaller sub-problems that can then be solved recursively.

"most important attribute," means the one that makes the most difference to the classification of an example.
With hope to get to the correct classification with a small number of tests -
meaning that all paths in the tree will be short and the tree as a whole will be shallow.

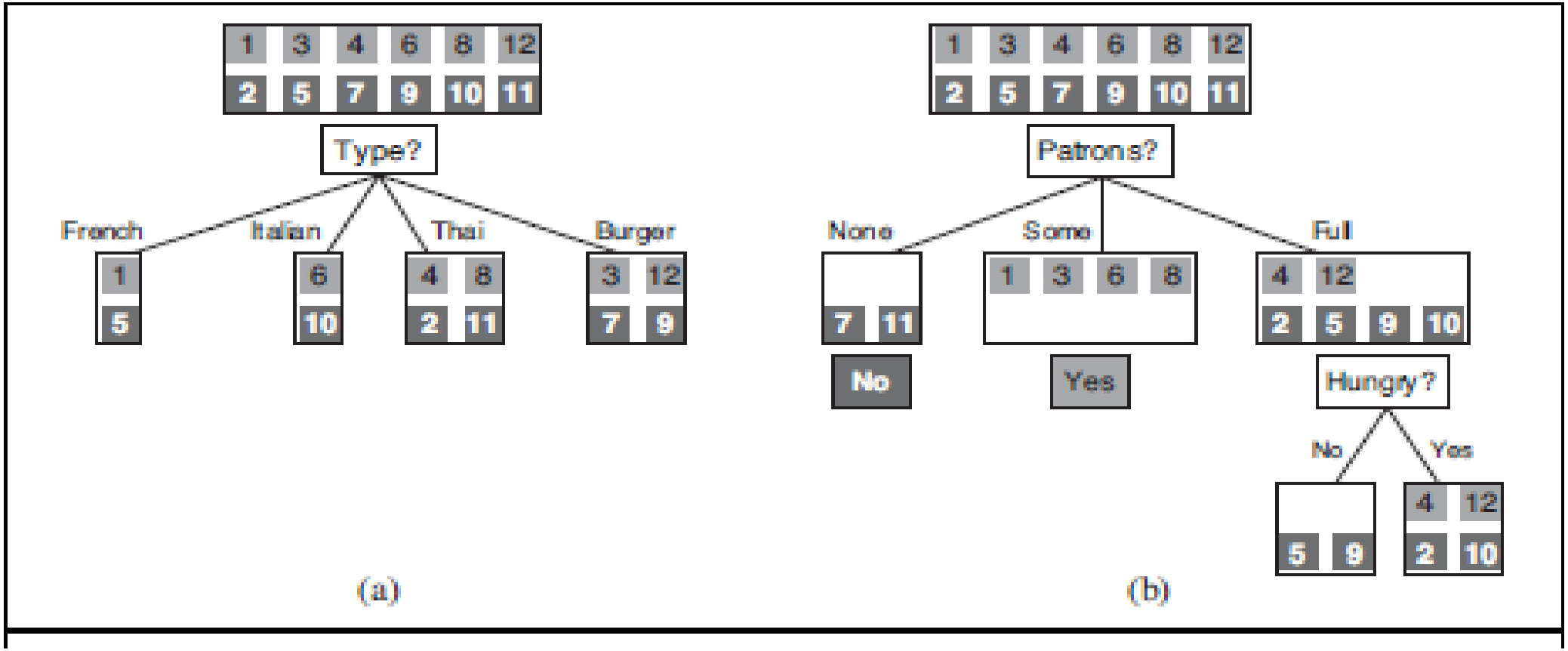
Decision Tree Learning Algorithm -
Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

Importance of Attributes

Splitting the examples by testing on attributes. Each node shows 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. By Splitting on Patrons does a good job of separating positive and negative examples. After splitting on Patrons, Hungry is a fairly good second test.



Attributes Choice

Type

a poor attribute, because it leaves us with four possible outcomes, each of which has the same number of positive as negative examples.

Patrons

fairly important attribute,

if the value is None or Some, then we are left with example sets for which we can answer definitively (No and Yes, respectively).

If the value is Full , we are left with a mixed set of examples.

After the first attribute test splits up the examples, each outcome is a new decision tree learning problem in itself, with fewer examples and one less attribute.

Considerations in recursive solution

1. If the remaining examples are all positive (or all negative), then we are done: we can

answer Yes or No.

2. If there are some positive and some negative examples, then choose the best attribute to

split them.

3. If there are no examples left, it means that no example has been observed for this combination of attribute values, and we return a default value calculated from the plurality classification of all the examples that were used in constructing the node's parent. These are passed along in the variable parent examples.

4. If there are no attributes left, but both positive and negative examples, it means that these examples have exactly the same description, but different classifications.

This can happen because there is an error or noise in the data; because the domain is nondeterministic; or because we can't observe an attribute that would distinguish the examples.

The best we can do is return the plurality classification of the remaining examples.

Aim: find a small tree consistent with the training examples

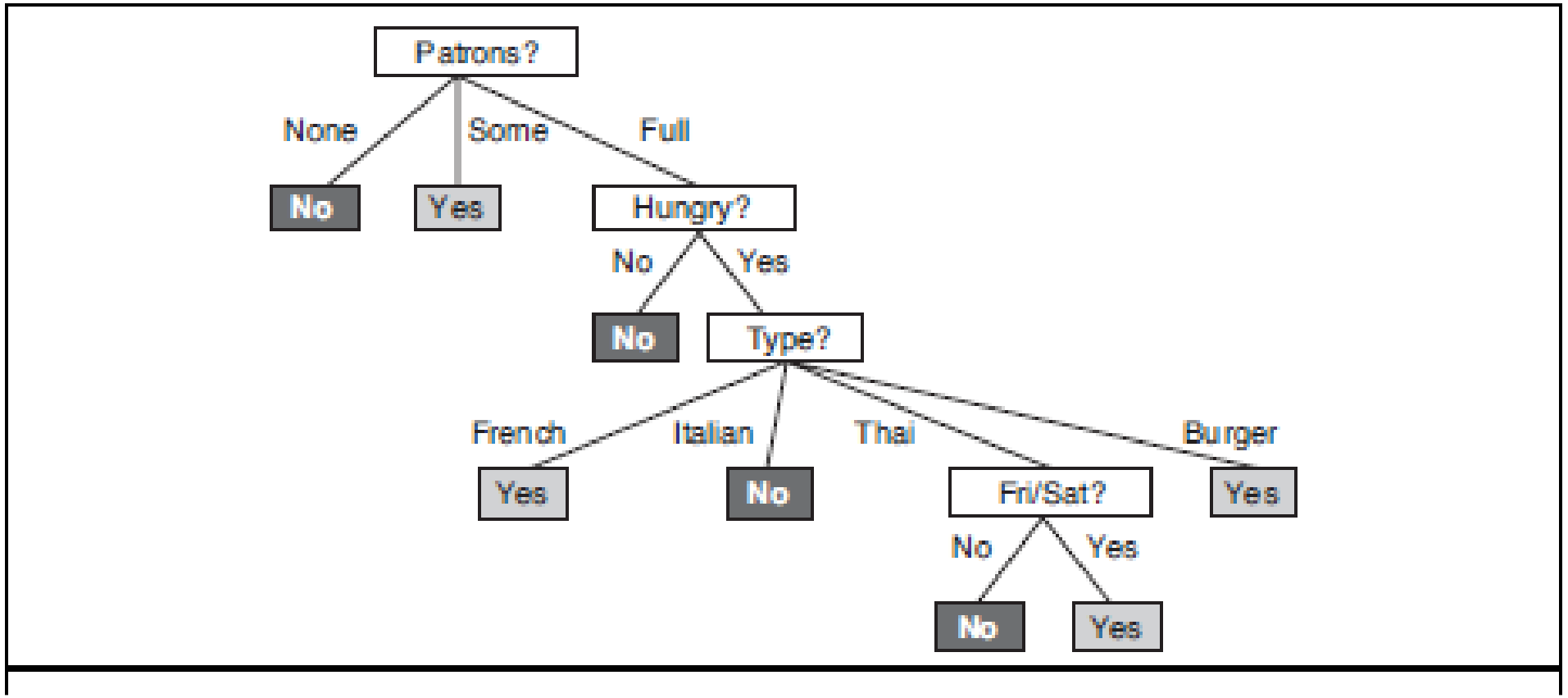
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

Decision Tree Algorithm

```
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←argmaxa ∈ attributes IMPORTANCE(a, examples)
        tree←a new decision tree with root test A
        for each value vk of A do
                exs ←{e : e ∈examples and e.A = vk}
                subtree←DECISION-TREE-LEARNING(exs, attributes −A, examples)
                add a branch to tree with label (A = vk) and subtree subtree
        return tree
```

The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

decision tree induced from the 12-example training set with decision tree algorithm



Decision Tree algorithm

the set of examples is crucial for constructing the tree,

but nowhere do the examples appear in the tree itself.

A tree consists of just tests on attributes in the interior nodes,

values of attributes on the branches, and

output values on the leaf nodes.

The learning algorithm looks at the examples, not at the correct function

its hypothesis not only is consistent with all the examples, but is considerably simpler than the original tree!

The learning algorithm has no reason to include tests for Raining and Reservation, because it can classify all the examples without them.

It has also detected an interesting and previously unsuspected pattern: the patron will wait for Thai food on weekends.

It is also bound to make some mistakes for cases where it has seen no examples. –

it has never seen a case where the wait is 0–10 minutes but the restaurant is full.

In that case it says not to wait when Hungry is false,

With more training examples the learning program could correct this mistake.

Decision Tree algorithm

When there are several variables of similar importance,

the choice between them is somewhat arbitrary

with slightly different input examples, a different variable would be chosen to split on first,

and the whole tree would look completely different.

The function computed by the tree would still be similar,

but the structure of the tree can vary widely.

Assessing the performance of the learning algorithm

A learning algorithm is good if it produces hypotheses that do a good job of predicting the classifications of unseen examples.

a prediction is good if it turns out to be true, so we can assess the quality of a hypothesis by checking its predictions against the correct classification once we know it.

We do this on a set of examples known as **the test set.**

methodology:

1. Collect a large set of examples.
2. Divide it into two disjoint sets: the training set and the test set.
3. Use the learning algorithm with the training set as examples to generate a hypothesis H.
4. Measure the percentage of examples in the test set that are correctly classified by H.
5. Repeat steps 1 to 4 for different sizes of training sets and different randomly selected training sets of each size.

The result of this is a set of data that can be processed to give the average prediction quality

as a function of the size of the training set.

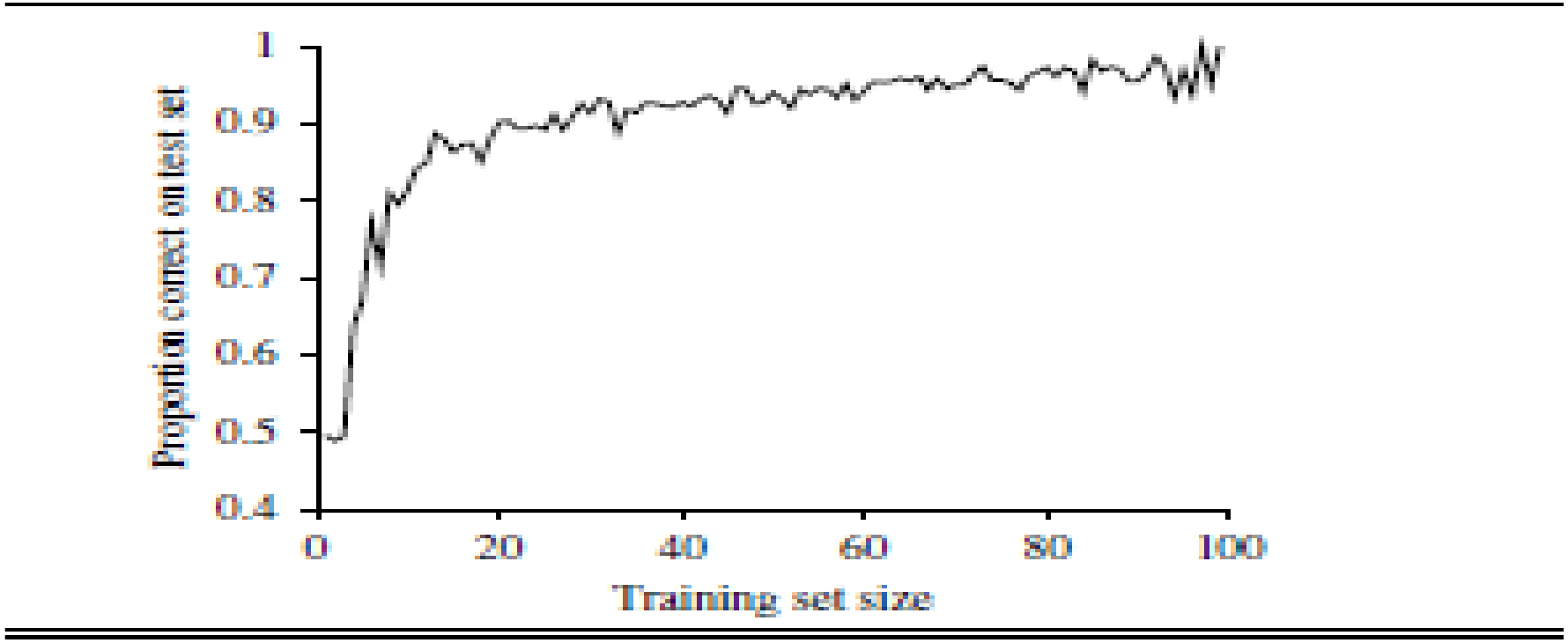
plotted on a graph, give the **learning curve** for the algorithm on the particular domain.

Assessing the performance of the learning algorithm

As the training set grows, the prediction quality increase. (learning curves are also called happy graphs.)

The key idea of the methodology is to keep the training and test data separate

accuracy of a learning algorithm with a learning curve.



Choosing attributes

The greedy search used in decision tree learning is designed to approximately minimize the

depth of the final tree.

The idea is to pick the attribute that goes as far as possible toward providing an exact classification of the examples.

A perfect attribute divides the examples into sets, each of which are all positive or all negative and thus will be leaves of the tree.

The Patrons attribute is not perfect, but it is fairly good.

A really useless attribute, such as Type, leaves the example sets with roughly the same proportion of positive and negative examples as the original set.

need for formal measure of “fairly good” and “really useless”

Information gain in terms of Entropy - measure of uncertainty of a random variable - acquisition of information corresponds to a reduction in entropy.

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;

no information is gained by observing its value.

A flip of a fair coin is equally likely to come up heads or tails, 0 or 1 --- counts as “1 bit” of entropy.

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

$$\text{Entropy:} \quad H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k) \,.$$

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

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

Entropy

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

$$H(Goal) = B (p/(p+n))$$

Restaurant example $p = n = 6$

$$H(Goal) = B(0.5) = 1 \text{ bit}$$

A test on a single attribute A may give us only a part of this 1 bit.

How much - Can be measured by looking at the entropy remaining after the attribute test

Entropy

An attribute A with d distinct values divides the training set E into subsets E1, . . . ,Ed.

Each subset Ek has pk positive examples and nk negative examples,

if we go along that branch, we will need an additional B(pk/(pk + nk)) bits of information to answer the question.

A randomly chosen example from the training set has the kth value for the attribute

with probability (pk + nk)/(p + n),

the expected entropy remaining after testing attribute A is

$$\text{Remainder (A)} = \sum_{k=1}^{d} ((pk+nk)/(p+n))\, B\,(pk/(pk+nk))$$

Information gain from the attribute test on A = expected reduction in entropy

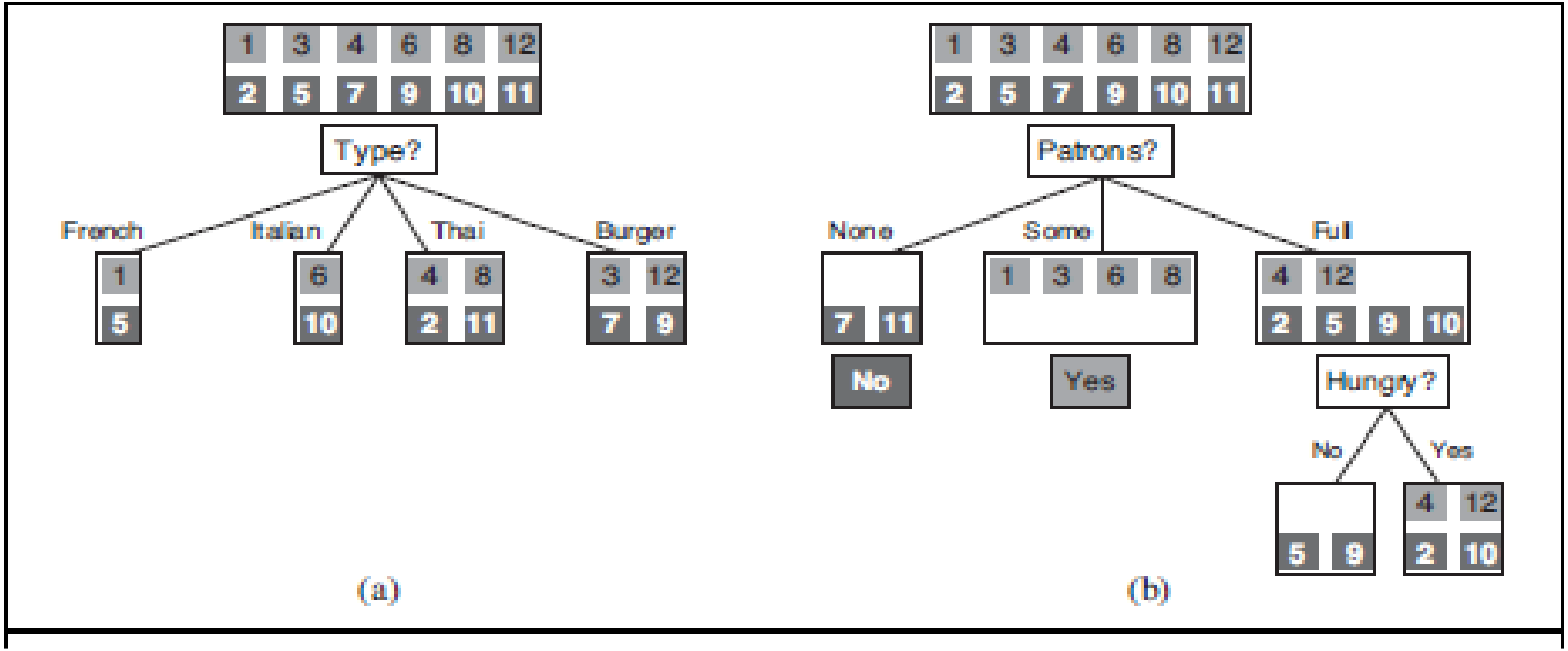
$$\text{Gain (A)} = B\,(p/(p+n)) - \text{Remainder (A)}$$

Importance of Attributes

Splitting the examples by testing on attributes. At each node the positive (light boxes) and negative (dark boxes) examples remaining are shown

(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.



Entropy

$$\text{Remainder}(A) = \sum_{k=1}^{d} ((pk+nk)/(p+n))\, B(pk/(pk+nk))$$

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

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

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

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

Patrons is a better attribute to split on.

Patrons has the maximum gain of any of the attributes and would be chosen by the decision-tree learning algorithm as the root.

Overfitting and Pruning

DECISION-TREE-LEARNING algorithm will generate a large tree when there is actually no pattern to be found.

Overfitting problem

example

If it turns out that there are 2 rolls of a 7-gram blue die with fingers crossed and they both come out 6,

then the algorithm may construct a path that predicts 6 in that case.

Overfitting - more likely as the hypothesis space and the number of input attributes grows

less likely as we increase the number of training examples.

decision tree pruning to combat overfitting

eliminating nodes that are not clearly relevant ….

start with a full tree, as generated by DECISION-TREE-LEARNING.

look at a test node that has only leaf nodes as descendants.

If the test appears to be irrelevant—detecting only noise in the data— then,

eliminate the test, replacing it with a leaf node

repeat this process, considering

each test with only leaf descendants, until each one has either been pruned or accepted as is.

Overfitting and Pruning

how to detect that a node is testing an irrelevant attribute?

information gain is a good clue to irrelevance.

Consider a node consisting of p positive and n negative examples.

If the attribute is irrelevant, it would split the examples into subsets

that each have roughly the same proportion of positive examples as the whole set, p/(p +n),

and so the information gain will be close to zero.

how large a gain should we require in order to split on a particular attribute?

use a statistical significance test. (χ2 pruning)

early stopping—

have the decision tree algorithm stop generating nodes when there is no good attribute to split on, rather than going to all the trouble of generating nodes and then pruning them away.

problem with early stopping

it stops us from recognizing situations where there is no one good attribute, but there are combinations of attributes that are informative.

Ensemble Learning

Predictions are made using learning methods with a single hypothesis, chosen from a hypothesis space, so far.

Ensemble learning methods select a collection, or ensemble, of hypotheses from the hypothesis space and combine their predictions.

Example:

during cross-validation we might generate twenty different decision trees, and have them vote on the best classification for a new example.

Consider an ensemble of K =5 hypotheses and Combine their predictions using simple majority voting.

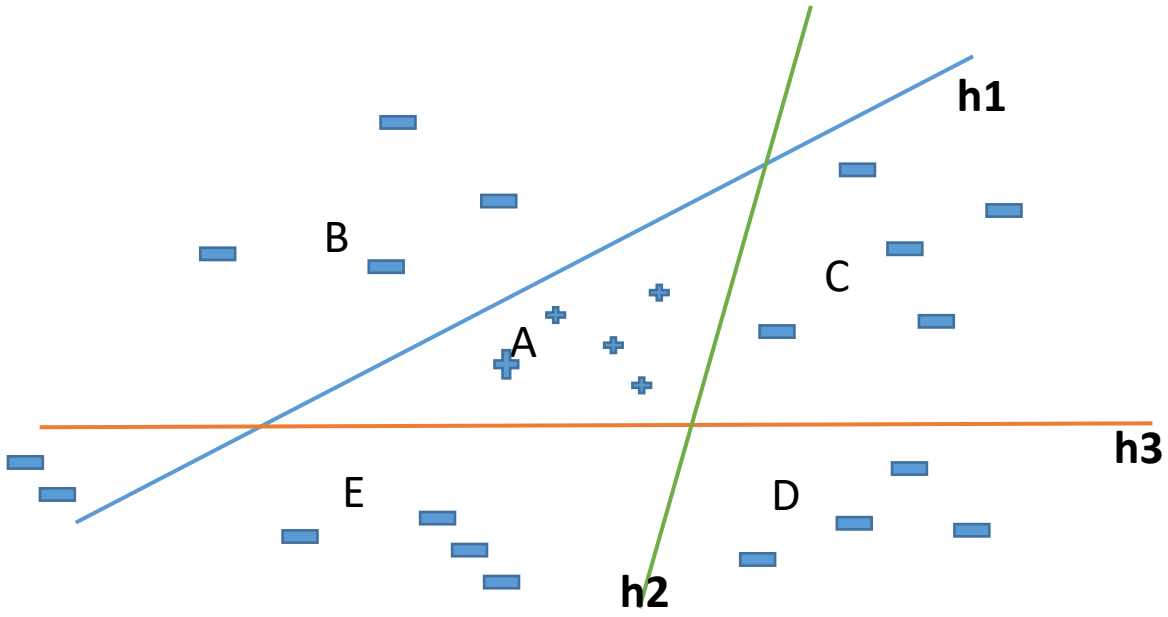
For the ensemble to misclassify a new example, at least three of the five hypotheses have to misclassify it.

The hope is that this is much less likely than a misclassification by a single hypothesis.

assume that each hypothesis hk in the ensemble has an error of p—that is, the probability that a randomly chosen example is misclassified by hk is p. errors made by each hypothesis are assumed to be independent.

Ensemble Learning

increased expressive power obtained by ensemble learning. We take three linear threshold hypotheses, each of which classifies positively on the unshaded side, and classify as positive any example classified positively by all three. The resulting triangular region is a hypothesis not expressible in the original hypothesis space.



Ensemble Learning

ensemble idea: a generic way of enlarging the hypothesis space

think of the ensemble itself as a hypothesis and the new hypothesis space as the set of all possible ensembles constructable from hypotheses in the original space.

If the original hypothesis space allows for a simple and efficient learning algorithm,

then the ensemble method provides a way to learn a much more expressive class of hypotheses

without incurring much additional computational or algorithmic complexity.

Boosting: most widely used ensemble method

weighted training set: In training set, each example has an associated weight $w_j \geq 0$.

The higher the weight of an example, the higher is the importance attached to it during the learning of a hypothesis

Ensemble Learning – Boosting

Boosting: most widely used ensemble method

weighted training set: In training set, each example has an associated weight $w_j \geq 0$.

> The higher the weight of an example, the higher is the importance attached to it during the learning of a hypothesis

Boosting starts with $w_j = 1$ for all the examples (i.e., a normal training set).

From this set, it generates the first hypothesis, h1.

This hypothesis will classify some of the training examples correctly and some incorrectly

the next hypothesis needs to do better on the misclassified examples, so increase their weights

while decreasing the weights of the correctly classified examples.

From this new weighted training set, we generate hypothesis h2.

The process continues until K hypotheses are generated, where K is an input to the boosting algorithm.

The final ensemble hypothesis is a weighted-majority combination of all the K hypotheses,

each weighted according to how well it performed on the training set.

Ensemble Learning - Boosting

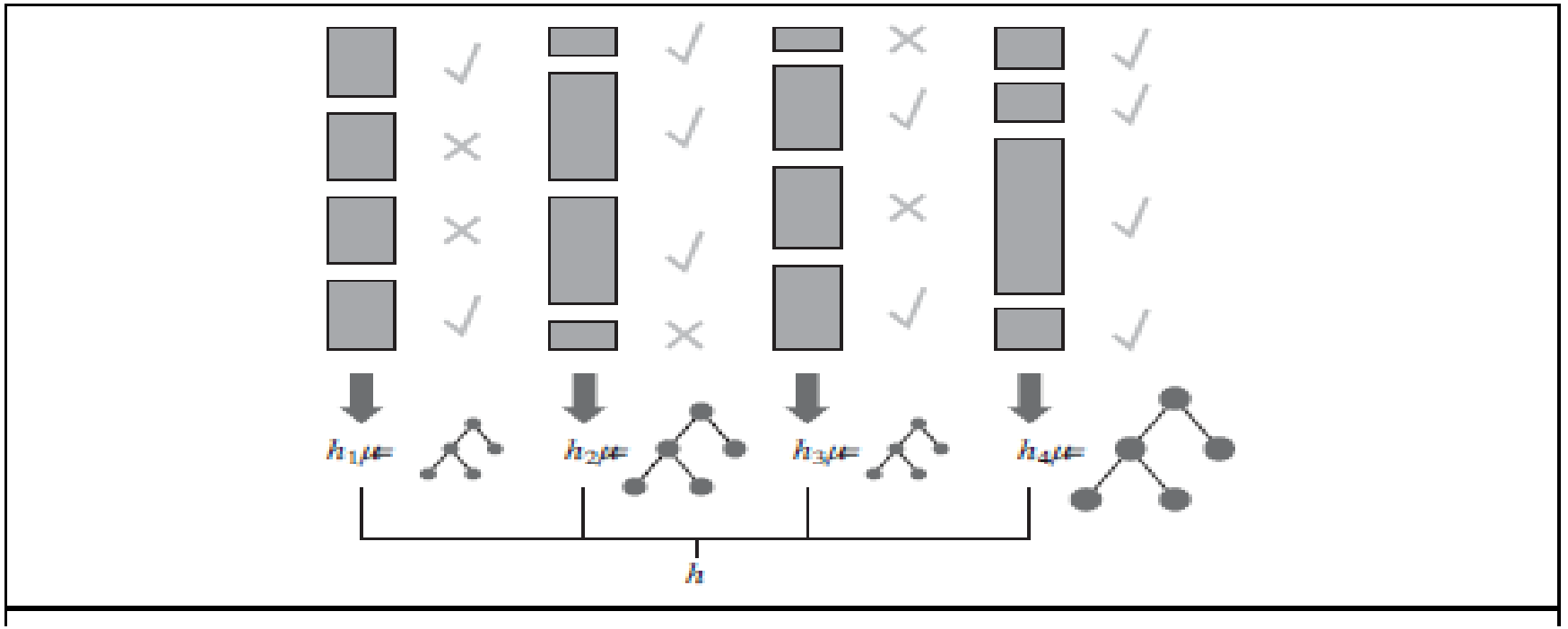
boosting algorithm:

Each shaded rectangle corresponds to an example;

the height of the rectangle corresponds to the weight.

The checks and crosses indicate whether the example was classified correctly by the current hypothesis.

The size of the decision tree indicates the weight of that hypothesis in the final ensemble.



Ensemble Learning - Boosting

Example	Age	Income	Student	Credit-Rating	Buys-Computer
x1	<=30	High	no	Fair	no
x2	<=30	High	no	Excellent	no
x3	31-40	High	no	Fair	yes
x4	>40	Medium	no	Fair	no
x5	>40	low	yes	Fair	yes
x6	>40	low	yes	Excellent	no
x7	31-40	low	yes	Excellent	yes
x8	<=30	medium	no	Fair	no
x9	<=30	low	yes	Fair	yes
x10	>40	medium	yes	Fair	no
x11	<=30	medium	yes	Excellent	yes
x12	31-40	medium	no	Excellent	yes
x13	31-40	high	yes	Fair	yes
x14	>40	medium	no	Excellent	no

Ensemble Learning - Boosting

Information Gain (Age) =0.246

Information Gain (Income) =0.029

Information Gain (Student) = 0.151

Information Gain (credit_rating) =0.048

Homework

1.

Identify attributes and values for attributes for decision making in joining an engineering college (you can use examples from your own class)

Draw a suitable decision tree with the attributes identified

Try decision tree algorithm to generate decision tree for choice of college decision.

2.

Identify attributes and values for attributes for decision making in selecting a job/occupation post graduation

Draw a suitable decision tree with the attributes identified

Try decision tree algorithm to generate decision tree for choice of occupation decision.