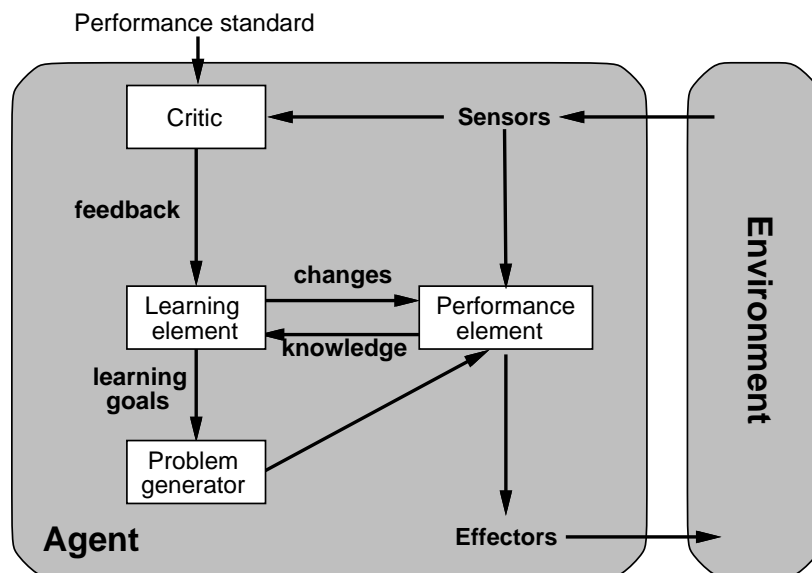


Learning Agents: Learning from Observations

Chapter 18, Sections 1-7

A General Model of Learning Agents

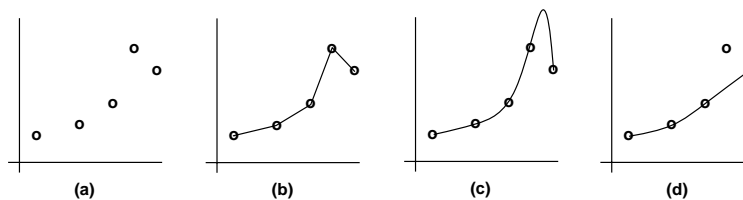


Inductive Learning

- In supervised learning, the learning element is given the correct (or approximately correct) value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback.
- An example is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x .
- The task of **pure inductive inference** (or **induction**) is this: given a collection of examples of f , return a function h that approximates f .
- The function h is called a **hypotesis**.

Example

Points (x, y) in the plane, where $y = f(x)$, find a function $h(x)$ that fits the points well.



In the figure: a) example points, b) piecewise linear fit, c) polynomial, d) simple function that ignores some sample points.

Preference for one hypotesis over another, beyond mere consistency with the examples, is called **bias**.

Skeleton for a Simple Reflex Learning Agent

```

global examples ← {}


---


function REFLEX-PERFORMANCE-ELEMENT(percept) returns an action
  if (percept, a) in examples then return a
  else
    h ← INDUCE(examples)
    return h(percept)


---


procedure REFLEX-LEARNING-ELEMENT(percept, action)
  inputs: percept, feedback percept
           action, feedback action
  examples ← examples ∪ {(percept, action)}

```

Learning Decision Trees

In classification, one way of generalizing from a set of examples is just to forget the features that weren't useful in discriminating between examples from different classes.

For example, if you see positive examples, $cs \wedge small$, $cs \wedge medium$, and $cs \wedge large$, then just forget about the size attribute and remember cs .

In the following, we show how to use decision trees to implement this sort of generalization.

Example

Consider the following decision tree.

Each nonterminal node specifies an attribute posing a question of the form ‘What is the value of the attribute?’ and each arc specifies an attribute value providing an answer to the posed question.

Each terminal node is associated with a set of examples. All of the examples are in the same class.

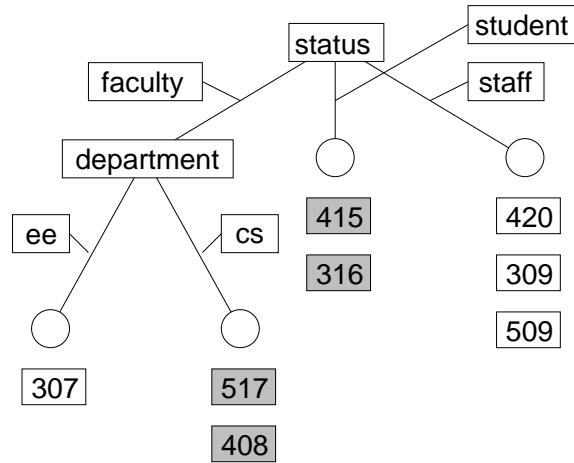
A path from the root to a terminal identifies a sequence of questions and answers that apply to the examples stored at the terminal.

The root implicitly answers the question ‘If an example is consistent with the questions and answers in the path from root to the terminal, what class is it in?’

Training examples for the robot janitor problem

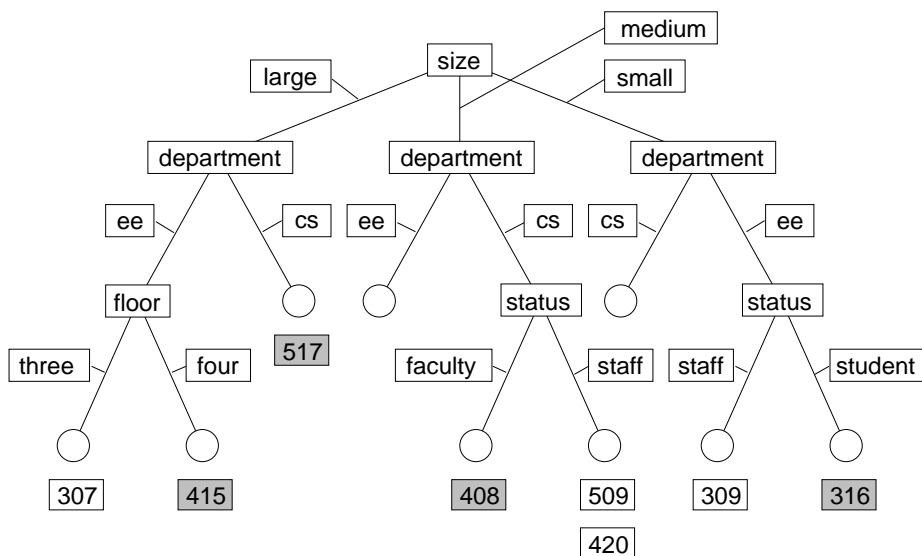
#	status	floor	department	size	recycling bin?
307	faculty	three	ee	large	no
309	staff	three	ee	small	no
408	faculty	four	cs	medium	yes
415	student	four	ee	large	yes
509	staff	five	cs	medium	no
517	faculty	five	cs	large	yes
316	student	three	ee	small	yes
420	staff	four	cs	medium	no

A compact decision tree for the robot janitor problem



The above decision tree represents the concept of ee and cs dpt. offices with recycling bins. How would you represent offices that have recycling bins and belong to students?

A large decision tree for the robot janitor problem



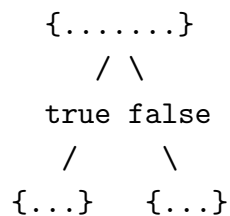
How do you choose an attribute to split the examples?

First we have to ask what does splitting accomplish?

Suppose you are considering a particular attribute to split a node containing a set of examples.

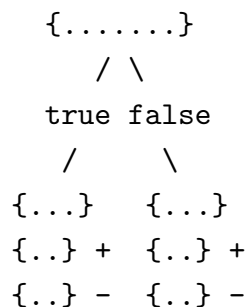
Why would you split the node at all?

Because the examples are not all in the same class. This means you have to discriminate further using additional attributes; each attribute serves to partition a set of examples according to the attribute values, say **true** and **false**, for a boolean attribute.



Splitting the examples

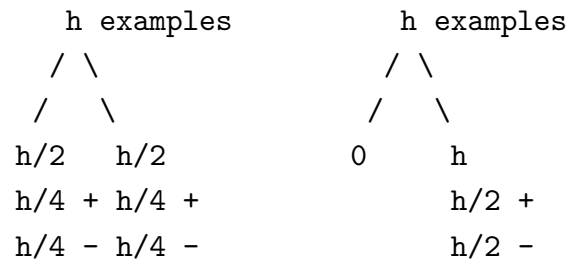
Now, each child has a set of examples that is further partitioned by the classes. Suppose there are just two classes + and -.



In deciding whether and how to split a node, we are interested in the number of examples in the class partitions at each child node.

Bad splits

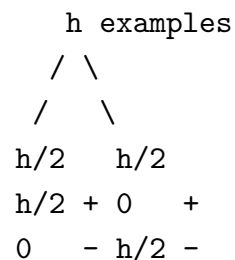
Suppose there are h examples at a given node. Here are some examples of bad splits:



These splits are bad because we have added to the size of the tree without improving its ability to discriminate among the examples.

Good splits

Here is an example of a good split:



This split is good because it results in a decision tree that correctly classifies all of the examples.

Which attribute?

We want to choose an attribute to minimize a measure that has

- i. a maximum of 1 when the examples are equally divided among the classes, and
- ii. a minimum of 0 when the examples all belong to the same class.

There are several measures that have this property; the book describes one particular measure based on information theory.

Using Information Theory

How much would you be willing to pay for an answer to a question?

If you already have a good guess about the answer, then the actual answer is less informative.

One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin.

If the possible answers ν_i have probabilities $P(\nu_i)$, then the information content I of the actual answer is given by

$$I(P(\nu_1), \dots, P(\nu_n)) = \sum_{i=1}^n -P(\nu_i) \log_2 P(\nu_i)$$

This is the average information content of the various events ($-\log_2 P$ terms) weighted by the probabilities of the events.

For the decision tree learning: What is the correct classification? A correct decision tree will answer this question.

Suppose the training set contains p positive examples and n negative examples. The estimate of the information contained in the correct answer is

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Any attribute A divides the training set E into subsets E_1, \dots, E_ν according to their values for A , where A can have ν distinct values.

Each subset E_i has p_i positive examples and n_i negative examples, so if we go along that branch we will need an additional $I(p_i/(p_i + n_i), n_i/(p_i + n_i))$ bits of information to answer the question. A random example has the i th value for the attribute with probability $(p_i + n_i)/(p + n)$, so on average, after testing attribute A , we will need

$$\text{Remainder}(A) = \sum_{i=1}^{\nu} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

bits of information to classify the example.

The **information gain** from the attribute test is defined as the difference between the original requirement and the new requirement:

$$Gain(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - Remainder(A)$$

and the heuristic used in the CHOOSE-ATTRIBUTE function is just to choose the attribute with the largest gain.

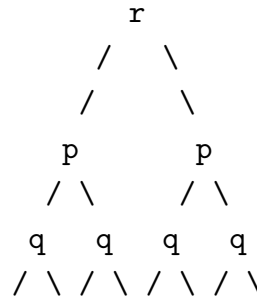
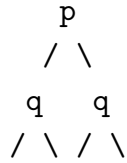
Think about concepts involving exclusive-or formulas. Suppose there are three boolean attributes $\{p, q, r\}$ and the target concept is $((p \wedge \neg q) \vee (\neg p \wedge q))$, i.e., p xor q .

Given the training examples,

- | | | | | | | |
|----|----------|----------|----------|----------|----------|-----|
| 1. | p | \wedge | q | \wedge | r | no |
| 2. | $\neg p$ | \wedge | $\neg q$ | \wedge | $\neg r$ | no |
| 3. | $\neg p$ | \wedge | q | \wedge | r | yes |
| 4. | p | \wedge | $\neg q$ | \wedge | r | yes |

should we split first on p , q , or r ?

With the greedy algorithm described above we would split on r but this is not optimal.



Noise!

Noise can cause examples to be misclassified.

Why is this the case?

How might noise be introduced into learning?

Noisy training examples may mean that you are not able to classify all examples.