Principles of Artificial Intelligence
Fall 2005
Handout #6
Learning

Vasant Honavar
Artificial Intelligence Research Laboratory
Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50011

So far we have looked at simple agents that solve problems based on searching (which may involve heuristics). But consider the task of identifying a person. Such tasks are easy to do but tough to program and make a machine do it as lot of this knowledge is implicit.
We know describe agents that can improve their behavior through diligent study of their own experiences

1 Machine Learning

By *Machine Learning* we mean “programs that learn”. The goals of studying machine learning are:

1. To understand what learning is (in algorithmic terms).
2. To build systems that are more flexible and can “program themselves”.

1.1 Definition of Learning

By learning, we mean any process by which a system *improves* its performance on a *set of tasks* as a result of *experience.*

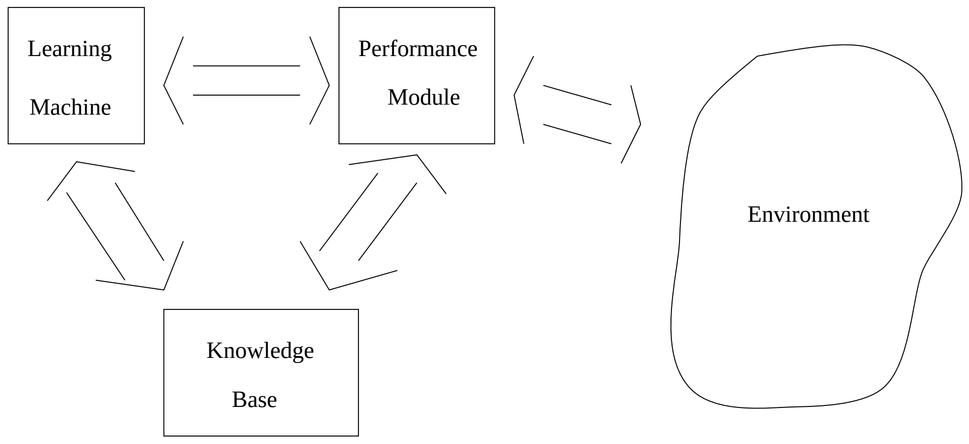


Figure 1: A simple model of a learning system.

1.2 Types of Learning

1. **Rote Learning (memorization)** In this type of learning, almost no inference is involved. Knowledge is stored in a form that is the same as the one is which it was provided.

   Rote Learning is useful if it is more efficient or expedient to retrieve a piece of stored data than it is to recompute it (perhaps using search).

   Examples:

   - Assume an adversarial games where player1 chooses the max h value, and player2 chooses the min h value. The h values at depth 3 are given. For player1's turn, the value of a node at level 2 is the max value of a child at level 3, at level 1 is the min value of the children at level 2, and the value at level 0 will be the max value at level 1.

     Store the values computed, and use the stored value of a node rather than searching again. Storing a position increases the effectiveness of the search i.e., the heuristic becomes more informed, or the effective depth of the search is increased.
   - On the Checkers game, Samuels found that by storing approximately 10^6 configurations and their backed up evaluations the program could attain near expert-level performance. (The size of the search space for checkers is approximately 10^{40}).

2. **Learning by Instruction** In this type of learning, some transformation of knowledge is necessary before it is stored for future use.

   Example: Classroom instruction.

3. **Learning by Induction (inductive inference)** In this type of learning, given a set of examples, the learner has to *induce* a set of general problem solving procedures.

   Example: Syntax of a language.

4. **Learning by Reinforcement (from exploration)**

   Example: A robot.

5. **Learning by Deduction (deductive inference)**

2 Inductive learning (IL)

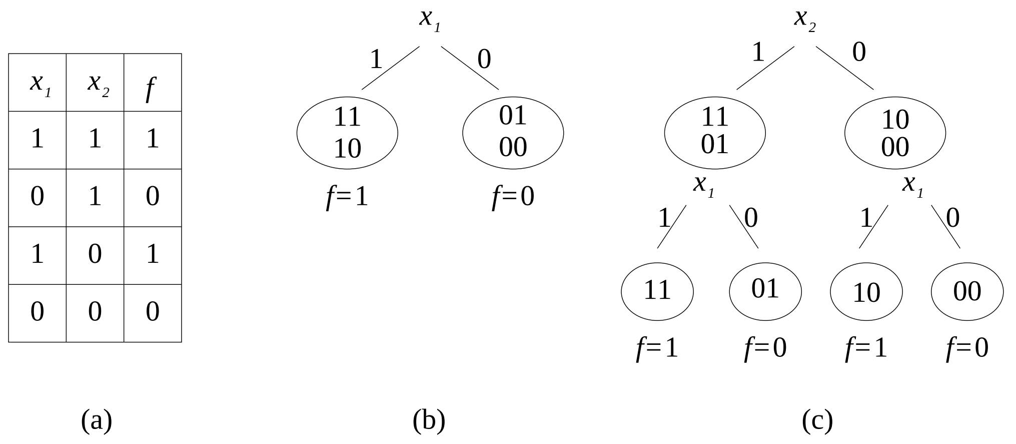
Inductive learning is learning from examples and counterexamples. Vast majority of the learning techniques fall in this category.
An *example* is an ordered pair (X_k, C_k), where X_k is a pattern (e.g. a vector of attribute values) and C_k is its desired classification.

1. If we want to learn grammars, X_k is a string and $C_k = 1$ indicates that X_k is a valid string in the language, and $C_k = 0$ not valid string. Examples might look like $(ABCDAC, 1)$, $(ACABBB, 0)$.
2. If we want to learn to classify images, X_k may be 2-d digitized image and C_k indicates its classification.

The task of an inductive learner is to learn a "concept description" or a function that

1. correctly classifies all (or most of) the training examples.
2. generalizes well on examples not used in training.

Different models of IL are decision trees, neural networks etc.



x_1	x_2	f
1	1	1
0	1	0
1	0	1
0	0	0

Figure 2: A simple decision tree. (a) The set of examples. (b) Decision tree by testing x_1 first is simpler. Since the instances at the tip of arcs in the same set are identical (most uniform), no further test is needed. (c) Decision tree by testing x_2 first. The instances in the set at the tip of arcs are not uniform, therefore more tests (on x_1) are necessary to completely classify the training set.

2.1 Learning Decision trees

Consider the training set that is shown in the figure reffig2.
In this example, most people would agree that (b) is probably a better hypothesis to choose than (c) for a variety of reasons. I.e. (b) is, in some sense, simpler than (c). We would, in general, prefer the simpler tree while all other things being equal. One might be inclined, in the absence of any other information, to select the simplest explanation or hypothesis that is *consistent* with the data.
This goes by the name Ockham's razor (after William of Ockham). In the case of decision trees, we can generate "simple" trees by picking "the most informative" attributes to rest on. Note however that it is not always a simple matter to decide how to measure the simplicity or complexity of a hypothesis.

3 Digression: Information theory

3.1 Information content of a message

Information content of a message is related to how surprising the message is. Considering a message m whose probability is P, Shannon defined the information content of the message as

$$I(m) = I_p = -\log_2 P\text{bits}.$$

Example: Information provided by a message that announces the outcome of a fair coin toss ("it is heads") is $-\log_2 \frac{1}{2} = 1$ bit (since the probability of "heads" is 1/2). The expected amount of information associated with messages that announce the outcome of a fair coin toss $= -(p_h log_2 p_h + p_t log_2 p_t) = 1$ bit where p_h is the probability of "heads," p_t is probability of "tails."
Example: Suppose there are 8 equally probable messages, information content of a message is 3 bits. If each of the outcomes is not equally likely, some of the messages will require more than 3 bits, some less than 3 bits, but the average will be less than 3 bits.

3.2 Information content of a set

Consider a set S of instances. Let S be partitioned into N disjoint subsets (classes) $C_1, C_2, \cdots, C_N$, as in Figure 3. (We use C_i to denote both the set C_i as well as the class name C_i.)
Suppose we select an instance $s \in S$, and announce that it belongs to class C_j. The probability of this message is $\frac{|C_j|}{|S|}$, so the *information content* of this message is $-\log_2\left(\frac{|C_j|}{|S|}\right)$ bits, and the expected information content

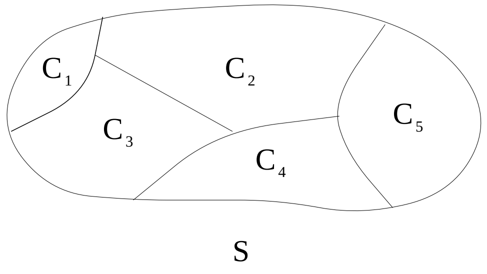


Figure 3: A disjoint set.

of a message announcing the class membership of a random instance $s \in S$ is

$$-\sum_j \frac{|C_j|}{|S|} \cdot \log_2 \left(\frac{|C_j|}{|S|} \right)$$

bits. (On average this much information is passed.)

4 Using Information Theory to Learn Decision Tree Classifiers

In this framework, classifying an instance is tantamount to acquiring the necessary information from the given instances. The task of learner is to find the most effective way (i.e., choose the questions to ask about the instances to be classified) to acquire the necessary information.
In the case of a decision tree learning algorithm, this includes identification of a appropriate sequence of tests to perform on the properties of the instances to be classified.
Example:
Instances are described by 3 attributes.
$A_1 = \{\text{tall, short}\} = \{\text{t, s}\}$
$A_2 = \{\text{dark, blonde, red}\} = \{\text{d, b, r}\}$
$A_3 = \{\text{blue, brown}\} = \{\text{l, w}\}$

Training set:
$I_1 = (\text{t d l})\ +$
$I_2 = (\text{s d l})\ +$
$I_3 = (\text{t b l})\ -$
$I_4 = (\text{t r l})\ -$
$I_5 = (\text{s b l})\ -$
$I_6 = (\text{t b w})\ +$
$I_7 = (\text{t d w})\ +$
$I_8 = (\text{s b w})\ +$

Information content of this set is

$$Info(S) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = .954 \text{ bits}$$

Testing of this set is shown in Figure 4.
Apply test A_1, information content of subset of A_1 is

$$E_t = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 \text{ bits}$$

$$E_s = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918 \text{ bits}$$

The expected information content after testing for A_1 (probability following this branch) is

$$E_{A_1} = \frac{5}{8} \times E_t + \frac{3}{8} \times E_s = 0.95 \text{ bits}$$

Following the same procedure, we can calculate the information content for testing A_2 and A_3,

$$E_{A_2} = 0.5 \text{ bits}, \quad E_{A_3} = 0.607 \text{ bits}.$$

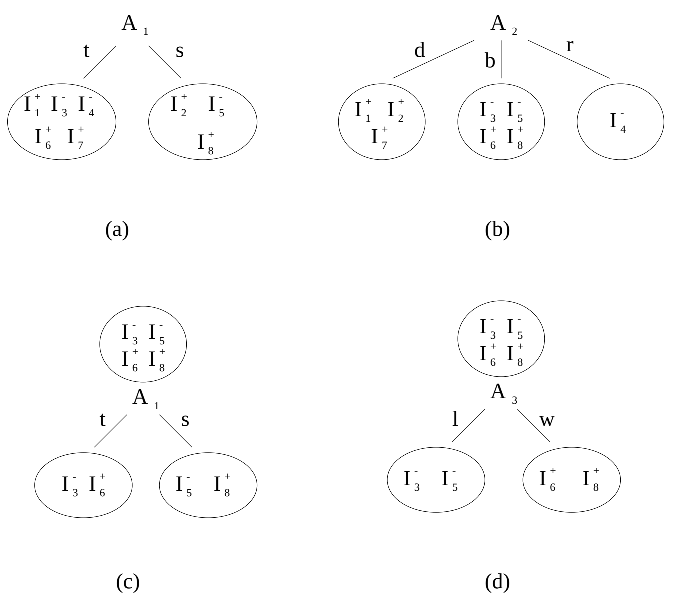


Figure 4: Information content of different tests. (a) Test for A_1 first. (b) Test for A_2 first. (c) Test A_1 for the b-subset after testing A_2. (d) Test A_3 for the b-subset after testing A_2

The lower the E, the less information needed to go on from this point. We see that A_2 is the most informative attribute yield an estimated expected information gain of $0.954 - 0.5 = 0.454$ bits. So we pick A_2 as the test to perform first.
At the next level, we have the choice to test between A_1 and A_3, similarly we can calculate as follows:

$$E_{A_2 A_1} = \frac{4}{8}\left[\frac{2}{4}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) + \frac{2}{4}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)\right] = 0.5 \text{ bits}$$

$$E_{A_2 A_3} = 0 \text{ bits.}$$

4.0.1 Limitation of this method

There are situations that this approach does not work well. For example, If we have a diagnostic procedure based on data of patients and SSN is used as a property, since SSN is unique for everyone, we have a complete classification for any training set. But this obviously is not what we are looking for.
We can use an alternative criterion based on "split information". Consider a set of S of instances which is tested on an attributed A.

$$-\sum_j \frac{|S_j|}{|S|} \cdot \log_2\left(\frac{|S_j|}{|S|}\right)$$

bits. where S_1,S_2,...,S_k correspond to the subset of S that constitute a partition of S on the basis of values of the attribute A.
If A is the SSN, k=$|S|$

$$\text{split-info}(S, A) = -\sum_j \frac{1}{|S|} \cdot \log_2\left(\frac{1}{|S|}\right) = \log_2(|S|)$$

bits.
We can define an information gain ratio,

$$G(S, A) = \frac{\text{info-gain}(S, A)}{\text{split-info}(S, A)}$$

$$\text{info-gain}(S, A) = E - EA$$

where E is the information content of S, and EA the information content of S after testing on A.

4.1 Overfitting

Given a hypothesis space H—in our case, the set of all possible decision trees defined using the given set of attributes—we say that a particular hypothesis h of H (in our case, a particular decision tree) is said to overfit the training data, if there exists a hypothesis h' of H, such that h has less error than that of h' on the training data, but the error of h' is less than that of h over the entire distribution of instances.
To overcome the overfitting problem, we can divide the examples into training set and a validation set. Then use the following training approach:

- Build a decision tree correctly classify the training set. Extract rules from the tree by doing a depth first traversal of the tree to find all the paths from the root to the leaves. Each rule is of form, $(A_1 = a)$ AND $(B_1 = b) \ldots \Rightarrow$ class $= i$
- For each condition in each rule, see if the accuracy of test data can be improved by chopping that condition. If so, modify the rule S accordingly.

During the classification, if a unique label can not be assigned, we guess the classification based on the distribution of instances.
A number of variations on the basic decision tree algorithm (including methods for dealing with continuous valued inputs, alternative splitting criteria, and alternative pruning criteria) are available in the literature.