Learning

In providing expert systems, knowledge engineering is often overwhelming. Also, there are many situations where one would want a program to learn the necessary techniques. Machine learning can do generalization, aid humans and avoid brittleness. One standard problem is the categorization or classification problem. But there are many others. We saw earlier a discussion in the chapter on information theory of how much can one learn by asking one question.

3.1 Types of Learning

One can say there are three types of learning:

- ***supervised***: training sets with example input and output
- completely ***unsupervised*** learning: must discover/create the knowledge
- ***reinforcement*** is a type of supervised learning in which only feedback (positive or negative reward) is given at end of a sequence of steps. It requires assigning reward to steps by solving the credit assignment problem—which steps should receive credit or blame for a final result?

> EXAMPLE 3.1. *An example of pure feedback learning is a matchbox player of tictactoe. Start with a matchbox for each possible position in tictactoe, with the position drawn on the side. Then place inside each matchbox a few beans. The player plays as follows: for each move, choose arbitrary successor state (which contains a bean). If the machine loses, then take away a bean from the last matchbox used. Discard empty matchboxes. If it reaches a position with no successor matchbox, then resign. The machine soon plays a perfect player (never losing). But the matchbox knows nothing about the goals, and only needs a generator of legal moves.*

At one end of the spectrum is knowledge-rich learning such as ***deductive learning***: this uses only knowledge, not examples. At the other end is knowledge-poor learning such as ***neural networks***, which use only examples. In between are many areas such as explanation-based learning, case-based reasoning and inductive learning.

Of course, one wants some confidence in what is produced. Hence validation (on unseen training sets, for example) and statistical tests.

3.2 Classification

We saw earlier that one way to approach a classification problem, is to create a decision tree using information theory and the induction algorithm. Another very successful technique is ***clustering*** (unsupervised learning): try to make sense of a collection of items. The idea is to find how they cluster. One approach is to try to find the best split: maybe a splitting plane. Or an iterative approach: add new object to nearest cluster. For all this, one needs decision metrics. Clustering is an important field in statistics.

3.3 Other Areas of Learning

- The standard ***induction*** learning problem can be viewed as the following: given a collection of examples of f, return a function that approximates f. Of course there are many; so one uses ***Occam's razor***: choose the simplest explanation.

- ***Neural networks*** are good at function approximation. They simulate the internal structure of the brain. The theory is that the brain consists of nerve cells or neurons which are connected. Some neuron inputs excite, some inhibit. So in an artificial neural network, each node fires if the input exceeds a certain threshold. It is the threshold that is trained through the feedback. Many networks use backpropagation: the network has cycles. The results from neural networks will always be approximate, and inaccessible: one cannot know what they have learned.

- ***rote learning***: One-to-one mapping from inputs to stored representation. "Learning by memorization." Of course, even in other situations every time a value is computed it can be remembered.

- ***case-based reasoning***: The program has a database of examples: for example, a collection of block-world tasks and the corresponding action. If there is an exact match, then the program just returns the stored action. But if not, the program finds the closest match, and tries to make the necessary changes. For example, the input might differ from a stored position in only one piece, so the program substitutes and adapts the action accordingly. Of course, this might not work if the action needed a specific property of the original piece (e.g. being flat). This approach easily handles explanations and missing data.

- ***deductive learning***: automated theorem proving (in some sense this is not new knowledge, just logical consequence of what we already know). Same techniques used in planning.

- ***model construction***: Use specific examples to reach general conclusions: might be model learning.
- ***explanation-based learning***: have only a few examples, but the programmer tries to explain why they work (and so is generalizable).
- A huge area is ***statistical learning***. Two of the commonest approaches go by the names of maximum likelihood and maximum a priori learning. Use a lot of Bayes and Gauss.

Computational learning theory shows that under certain conditions the result is probably approximately correct.