*The Knowledge Engineering Review,* Vol. 14:4, 1999, 383–385
*Printed in the United Kingdom*


# Book reviews

**Reinforcement learning** by AG Barto and RS Sutton, MIT Press, Cambridge, MA 1998, ISBN 0-262-19398-1

Can a machine learn how to play chess or backgammon? Can it discover optimal strategies for dispatching elevators in skyscrapers and for channel allocation in cellular telephone systems? And can it do so without instruction, just capitalizing on experience accrued in the course of interactions with the environment and/or with appropriate models of environment? In living nature, such interactions are undoubtedly a major source of knowledge. An infant waves his hands so as to master proper coordination of the involved motor signals, a piano adept grinds five finger études, and a chess player studies typical board positions to discern patterns underlying strategic decisions. Deeper understanding of these learning phenomena is indispensable not only for advances in artificial intellingence, but also for many real-world computer applications, ranging from job-shop scheduling through computer games through the support of command and control in battlefield theaters.

Years ago, when I embarked on studies of machine learning, a colleague of mine did not even try to disguise his disrespect. In his view, young researchers found this field attractive only because of its apparent simplicity. Back then, most of the relevant research focused on induction of concept descriptions compatible with preclassified examples. Decision trees were a novelty and neural networks have just begun to stir scientific imagination. The trouble started when it came to applications in really difficult tasks. In spite of many success stories (Michalski, Bratko and Kubat, 1998), the discipline never turned into major industry, and the reason certainly was not weak motivation of researchers seeking easy topics for their PhD disertations.

The problem is that this world does not come to us in neat sets of pre-classified training examples. The tasks mentioned above can hardly be attacked by classical concept-learning algorithms. When an agent takes an action, the environment does not return precise information as to what the correct decision should have been. Rather, the agent receives a signal that can be interpreted as a reward or punishment. The agent then modifies internal values of the respective actions with the objective to increase the chance of high rewards and rare punishments.

Computational approaches to learning from interaction are called *reinforcement learning*. In this framework, reality is interpreted as a cycle through states, actions and rewards. At each given moment, the environment finds itself in one out of many *states*. At each state, usually two or more different *actions* can be selected, and the agent wants to learn which choices maximize the system's benefit. In response to an action, the environment transforms from one state to another and the agent can – but does not have to – be *rewarded* (punishment is just a reward with negative sign).

A popular illustration is the *n*-armed-bandit task. Experimenter *X* is facing a slot machine with *n* levers, and has to decide which lever to pull. Each lever returns certain amount of money (usually zero) chosen from a probability distribution that is different for each lever. As to the characteristics of these distributions, *X* has no prior knowledge. Suppose, for simplicity, that the distributions do not change in time.

In principle, the solution might seem to be trivial. *X* will simply pull each lever one thousand times and keep a detailed tally of the returns. The lever with the highest average return is obviously the best and, from now on, *X* will stick to it. However, experimentation of this kind can become fairly expensive because each single trial costs a coin and *X* has to pull many times at levers with very rare returns. Systematic experimentation is most likely to give the correct answer in the end of a long "training" session, but not before considerable losses that can perhaps be avoided.

What is needed is a compromise that can be achieved by a more opportunistic approach. After

just a few pulls at each lever, *X* will develop some initial idea as to which lever is likely to be good and which can rather be supposed to be bad. This initial idea will be loaded with considerable uncertainty. The lever that looked good at the beginning can later prove to be less then optimal, the encouraging initial returns being due to mere statistical fluctuations. But even if *X* does not win as much as he might do with complete knowledge, the chances are that the rewards will still be higher than in random behavior or during the costly systematic experimentation.

Knowing that the initial opinion is far from reliable, *X* will alternate between two stages that are typical of reinforcement learning. During *exploitation*, *X* will stick to the lever that has so far yielded the highest profit; during *exploration*, *X* will try elsewhere. A simple look-up table keeps track of the average rewards at each lever. In the long run, *X* is likely to find the best choice, while not sacrificing too much in the process of learning. Algorithms emulating this behavior are investigated in Part I of the book. As it turns out, there are several different ways how to decide where to place the wager, and various possibilities how to update the look-up table. Each of them is carefully explained and illustrated by lucid examples.

With this, the first part sets stage for more advanced (and more realistic) applications where the task is complicated by two important features. First, the environment can find itself in many different states, each state calling for different actions.[1] A look-up table containing average rewards for all state-action pairs can contain millions of rows, a size that can easily become unmanageable. Secondly, a reward does not necessarily follow right after the action that deserves credit for it.

Both issues can be illustrated on a simplified version of the popular child's game of tic-tac-toe. Two players take turns on a three-by-three board. One player plays **X**s and the other **O**s until one of them manages to place three marks in a line, as illustrated by Figure 1. Each different combination of crosses and circles on the board represents a state. At each state, the player has to chose from one out of several possibilities where to place the mark (these are actions). And the reward will come only when the game is over: 1 for a win, –1 for a loss, and 0 for a drawn game. None of the individual moves can be picked as being exclusively responsible for the final outcome but each of them can be allowed to share part of the overall credit. Also, notice the probabilistic nature of the environment. The opponent can react to your moves with diverse responses that cannot be predicted with certainty – each opponent may follow a different strategy.

Part II of the book explains three fundamental solutions to the problem just outlined. The first approach, *dynamic programming*, assumes that a detailed model of the environment is available. The agent learns the values of the state-action pairs in the course of experimentation with a model, without incurring any real losses. By contrast, the second solution, the *Monte-Carlo* method, assumes that the agent does not have any *a priori* knowledge that would make it possible to build the model, and therefore has to resort to the costly experimentation with the environment, much as in the case of the *n*-armed-bandit task. The third strategy, *temporal-difference learning* (TD), can be viewed as a combination of the ideas behind the former two. Like Monte-Carlo methods, TD learns directly from raw experience (without the model of the environment's dynamics). Like dynamic programming, TD methods update their estimates of the state-action values using, to a certain extent, the estimates obtained before.

The art of any branch of Artificial Intelligence rests on proper understanding of the subtleties and idiosyncracies of the many available algorithms, and in obtaining synergy from their ingenious combination in powerful systems. The same applies to the three fundamental techniques used in reinforcement learning. For this reason, Part III of the book explaines advanced approaches that are in common use in real-world applications.

As a first step, Barto and Sutton explain the algorithm TD($\lambda$) that they present as a sophisticated amalgamation of Monte Carlo methods with temporal-difference learning. Then, they address another problem encountered in realistic applications: the requisite look-up table is often prohibitively large and therefore has to be replaced with function approximators that generalize across many states and actions. Finally, the authors show how to develop approximate models of

[1] The *n*-armed-bandit problem knows only a single state: 10 levers to be chosen from.

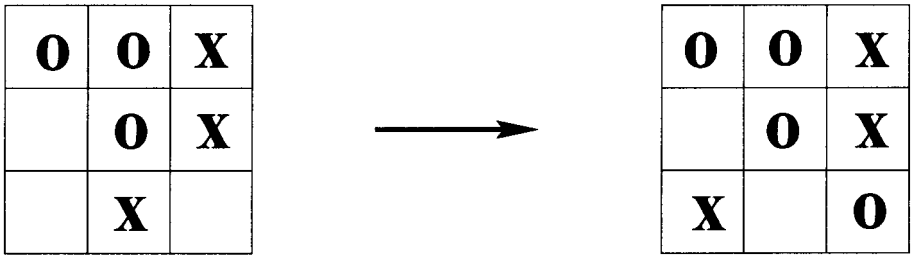


**Figure 1** The player of tic-tac-toe wins by placing three marks in a line.

the environment so as to marry the strengths of dynamic programming with those of heuristic search. All these approaches can be used synergistically as parts of joint methods.

Six case studies, concentrated in a separate chapter, illustrate the practical use of the explained methods and algorithms. They range from strong backgammon-playing programs to applications in job-shop scheduling, and have been carefully selected to show, convincingly, the advantages of reinforcement learning in realistic domains, while still being understandable for an average reader.

It is not easy to make predictions about the future of machine learning. The discipline is today known to be anything but easy, and its main contribution seems to be the lesson that the essence of learning in living nature is still a mystery. But if you want to know where one future direction of its computer implementation can be, read *Reinforcement Learning* by Barto and Sutton.

The book is self-contained and the text is accessible to any graduate or undergraduate student of computer science or engineering. All methods are explained with exceptional care and lucidity, using many illustrative examples. Essential ideas are illuminated by well-designed pictures. Unlike many other textbooks, this one takes special care in explaining the intuition behind the basic learning formulae. The material is logically structured and all algorithms are summarized in easy-to-follow tables. Those who are interested in the genesis of the individual ideas will appreciate that most chapters contain a section entitled "Bibliographical and Historical Remarks", where ample information can be found. I myself am using the book in my *Machine Learning Seminar*. AG Barto and RS Sutton have done excellent job and I am glad to recommend this book to the attention of anyone who is interested in this advanced approach to machine learning.

## Reference

Michalski, RS, Bratko, I and Kubat, M, 1998, *Machine Learning and Data Mining: Methods and Applications* Wiley.

Reviewed by Miroslav Kubat, Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA, USA (email: mkubat@cacs.usl.edu)