Reinforcement Learning

Richard S. Sutton

January 28, 1999

Reinforcement learning is an approach to artificial intelligence that emphasizes learning by the individual from its interaction with its environment. This contrasts with classical approaches to artificial intelligence and machine learning, which have downplayed learning from interaction, focusing instead on learning from a knowledgeable teacher, or on reasoning from a complete model of the environment. Modern reinforcement learning research is highly interdisciplinary; it includes researchers specializing in operations research, genetic algorithms, neural networks, psychology, and control engineering.

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a scalar reward signal. The learner is not told which action to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward, but also the next situation, and through that all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning.

One of the challenges that arises in reinforcement learning and not in other kinds of learning is the tradeoff between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover which actions these are it has to select actions that it has not tried before. The agent has to *exploit* what it already knows in order to obtain reward, but it also has to *explore* in order to make better action selections in the future. The dilemma is that neither exploitation nor exploration can be pursued exclusively without failing at the task.

Modern reinforcement learning research uses the formal framework of

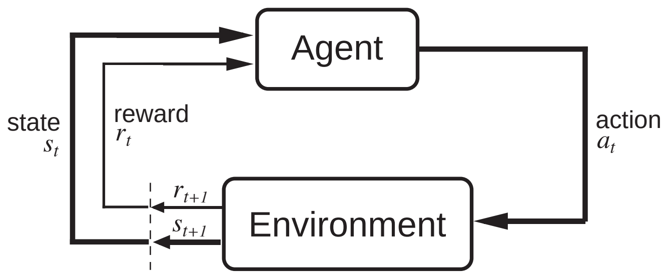

Figure 1: The Reinforcement Learning Framework

Markov decision processes (MDPs). In this framework, the agent and environment interact in a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$ On each step, the agent perceives the environment to be in a state, s_t, and selects an action, a_t. In response, the environment makes a stochastic transition to a new state, s_{t+1}, and stochastically emits a numerical reward, $r_{t+1} \in \Re$. (See Figure 1.) The agent seeks to maximize the reward it receives in the long run. For example, the most common objective is to choose each action a_t so as to maximize the *expected discounted return*:

$$E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots\right\}$$

where γ is a discount-rate parameter, $0 \leq \gamma \leq 1$, akin to an interest rate in economics. This framework is intended to capture in a simple way essential features of the problem of learning from interaction and thus of the overall problem of artificial intelligence. It includes sensation and action, cause and effect, and an explicit goal involving affecting the environment. There is uncertainty both in the environment (because it is stochastic) and about the environment (because the environment's transition probabilities may not be fully known). Simple extensions of this problem include incomplete perception of the state of the environment and computational limitations. Most theoretical results about reinforcement learning apply to the special case in which the state and action spaces are finite, in which case the process is called a *finite MDP*.

Reinforcement learning methods attempt to improve the agent's decision-making policy over time. Formally, a policy is a mapping from states to actions, or to probability distributions over actions. The policy is stored in a relatively explicit fashion so that appropriate responses can be generated quickly in response to unexpected states. The policy is thus what is sometimes called a “universal plan” in artificial intelligence, a “control law” in control theory, or a set of “stimulus-response associations” in psychology. We can define the *value* of being in a state s under policy π as the expected discounted return starting in that state and following policy π. The function mapping all states to their values is called the state-value function for the policy:

$$V\ \ (s) = E\ \ \Big\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s\Big\}.$$

The values of states defines a natural ordering on policies. Policy π is said to be better than or equal to policy π' if and only if $V\ \ (s) \geq V\ \ '(s)$ for all states s. For finite MDPs there are always one or more policies that are better than or equal to all others. These are the optimal policies, all of which share the same value function.

The simplest reinforcement learning algorithms apply directly to the agent’s experience interacting with the environment, changing the policy in real time. For example, *tabular 1-step Q-learning*, one of the simplest reinforcement learning algorithms, uses the experience of each state transition to update one element of a table. This table, denoted Q, has an entry, $Q(s, a)$, for each pair of state, s, and action, a. Upon the transition s_t ; s_{t+1}, having taken action a_t and received reward r_{t+1}, this algorithm performs the update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$$

where α is a positive step-size parameter. Under appropriate conditions (ensuring sufficient exploration and reduction of α over time), this process converges such that the *greedy policy* with respect to Q is optimal. The greedy policy is to select in each state, s, the action, a, for which $Q(s, a)$ is largest. Thus, this algorithm provides a way of finding an optimal policy purely from experience, with no model of the environment’s dynamics.

The algorithm described above is only the simplest of reinforcement learning methods. More sophisticated methods implement Q not as a table, but as a trainable parameterized function such as an artificial neural network. This enables generalization between states, which can greatly reduce learning time and memory requirements. Another common extension, *eligibility traces,* allows credit for a good state transition to spread more quickly to the states that preceded it, again resulting in faster learning. Continuous-time reinforcement learning problems require eligibility traces just as continuous-state or continuous-action problems require parameterized function approximators.

Reinforcement learning is also a promising approach to planning and problem solving. In this case, a model of the environment is used to simulate extensive interaction between it and the agent. This simulated experience is then processed by reinforcement learning methods just as if it had actually occurred. The result is a sort of "anytime planning," in which the agent's policy gradually improves over time and computational effort. Reinforcement planning methods appear most suited to problems that are too large or stochastic to be solved by conventional methods such as heuristic search or dynamic programming. This approach has already proved very effective in applications, having produced the best of all known methods for playing backgammon, dispatching elevators, assigning cellular-radio channels, and scheduling space-shuttle payload processing.

Further Readings

Sutton, R. S. and Barto, A. G. (1998) *Reinforcement Learning: An Ixxxntroduction.* Cambridge, MA: MIT Press.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Bertsekas, D. P., and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA.

Sutton, R. S. (ed.). (1992). Special issue of *Machine Learning* on reinforcement learning, 8(3/4).

Kaelbling, L. P. (ed.). (1996). Special issue of *Machine Learning* on reinforcement learning, 22(1-3).

Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.