Course Notes

Reinforcement Learning I: Introduction

Richard S. Sutton and Andrew G. Barto

[In which we try to give a basic intuitive sense of what reinforcement learning is and how it differs and relates to other fields, e.g., supervised learning and neural networks, genetic algorithms and artificial life, control theory. Intuitively, RL is trial and error (variation and selection, search) plus learning (association, memory). We argue that RL is the only field that seriously addresses the special features of the problem of learning from interaction to achieve long-term goals.]

1 Learning from Interaction

The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensorimotor connection to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals. This interaction is undoubtedly a major contributor to the infant's developing sense of its environment and of its own role in it. Experience remains a powerful teacher as the infant grows into a child and an adult, although the nature interaction changes significantly over time. Whether we are learning to drive a car or to hold a conversation, we are all acutely aware of how our environment responds to what we do, and we seek to influence its behavior. Learning from interaction is a foundational idea underlying nearly all theories of learning.

Reinforcement learning a computational approach to the study of learning from interaction. The last decade has seen the study of reinforcement learning develop into an unusually multi-disciplinary field; it includes researchers specializing in artificial intelligence, psychology, control engineering, operations research, neuroscience, artificial neural networks, and genetic algorithms. Reinforcement learning has particularly rich roots in the psychology of animal learning, from which it takes its name. A number of impressive applications of reinforcement learning have also been developed. The growing interest in reinforcement learning is fueled in part by the challenge of designing intelligent systems that must operate in dynamic real–world environments. For example, making robots, or robotic "agents", more autonomous (that is, less reliant on carefully controlled conditions) requires decision–making methods that are effective in the presence of uncertainty and that can meet time constraints. Under these conditions, learning seems essential for achieving skilled behavior, and it is under these conditions that reinforcement learning can have significant advantages over other types of learning.

Here we develop reinforcement learning from the point of view of artificial intelligence (AI) and engineering. (Reinforcement learning has also been developed with respect to psychology and neuroscience.) From this perspective, reinforcement learning corresponds to a particular mathematical formulation of the problem of learning from interaction. We examine this problem carefully and then explore some of the many algorithms for solving it that have been proposed in several different disciplines. By presenting these algorithms from a single perspective and using a unified notation, we try to make it easy to see how the different methods relate to one another and how they can can be combined most profitably. Perhaps surprisingly, almost all of these algorithms can be understood as various combinations of a few underlying principles.

Perhaps the most surprising outcome of the modern view of reinforcement learning is the close relationship it reveals between learning and planning, where by planning we mean deciding on a course of action by considering possible future situations before they are actually experienced. The earliest, and simplest, reinforcement learning algorithms make it possible to learn directly from environmental interaction without ever having to consider any situations that are not actually experienced. Using this type of reinforcement learning, a system can achieve highly skilled behavior without having any ability to predict how its environment might behave in response to its actions (that is, without having any sort of *model* of its environment). This is almost the opposite of planning. However, more complex forms of reinforcement learning have been devised that are closely related to computational methods known as dynamic programming, which do take advantage environment models, and these methods are closely related to the state–space planning methods of artificial intelligence. Today it is clear that reinforcement learning, in one or another of its various forms, can be applied to almost any planning problem, sometimes with significant advantages over more conventional planning methods.

2 Examples

A good way to introduce reinforcement learning is to consider some of the examples and possible applications that have guided its development:

- A master chess player makes a move. The choice is informed both by planning—anticipating possible replies and counter–replies—and by immediate, intuitive judgements of the desirability of particular positions and moves.
- An adaptive controller adjusts parameters of a petroleum refinery's operation in real time. The controller optimizes the yield/cost/quality tradeoff based on specified marginal costs without sticking strictly to the set points originally suggested by human engineers.
- A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 30 miles per hour.
- Phil prepares his breakfast. When closely examined, even this apparently mundane activity reveals itself as a complex web of conditional behavior and interlocking goal–subgoal relationships: Walking to the cupboard, opening it, selecting a cereal box, then reaching for, grasping, and retrieving it. Other complex, tuned, interactive sequences of behavior are required to obtain a bowl, spoon, and milk jug. Each step involves a series of eye movements to obtain information and to guide reaching and locomotion. Rapid judgements are continually made about how to carry the objects or whether it is better to ferry some of them to the dining table before obtaining others. Each step is guided by goals, such as grasping a spoon, or getting to the refrigerator, and is in service of other goals, such as having the spoon to eat with once the cereal is prepared and of ultimately obtaining nourishment.
- A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past.

These examples share features that are so basic that they are easy to overlook. All involve *interaction* between an active decision–making agent and its environment in which the agent seeks to achieve a *goal* despite *uncertainty* about its the environment. The agent's actions are permitted to affect the future state of the environment (e.g., the next chess position, the level of resevoirs of the refinery, the next location of the robot), thereby affecting the options and opportunities available to the agent at later times. Correct choice requires taking into account indirect, delayed consequences of actions, and thus may require foresight or planning.

At the same time, the effects of actions cannot be fully predicted, and so the agent must frequently monitor its environment and react appropriately. For example, Phil must watch the milk he pours into his cereal bowl to keep it from overflowing. All these examples involve goals that are explicit in the sense that the agent can judge progress toward its goal on the basis of what it can directly sense. The chess player knows whether or not he wins, the refinery controller knows how much petroleum is being produced, the mobile robot knows when its batteries run down, and Phil knows whether or not he is enjoying his breakfast. Moreover, an agent's goals are its *own* goals; not the goals of an outside agent or designer. If we want to use a reinforcement learning system for an engineering application, such as improving the yield of a petroleum refinery, we have to make a reinforcement learning system whose own goals are the same as ours.

In all of these examples, the agent can use its experience to improve its performance over time. The chess player refines the intuition he uses to evaluate positions, thereby improving his play; the gazelle calf improves the efficiency with which it can run; Phil learns to streamline his breakfast making. The level of knowledge the agent brings to the task at the start—either from previous experience with related tasks or from its genetic programming—influences what is useful or easy to learn, but interaction with the environment is essential for adjusting behavior to exploit specific features of each task.

3 Reinforcement Learning

Reinforcement learning is the learning of a mapping from situations to actions so as to maximizes a scalar reward or reinforcement signal. The learner does not need to be directly told which actions totake, 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, an action may affect not only the immediate reward, but also the next situation, and consequently all subsequent rewards. These two characteristics—trial and error search and delayed reward—are the two most important distinguishing characteristics of reinforcement learning.

All reinforcement learning algorithms require a particular synergistic combination of search and memory. Search is required to find good actions, and memory is required to remember what actions worked well in what situations in the past. The synergism arises because search provides the information that must be remembered, and memory makes search easier and faster. Reinforcement learning involves the systematic caching of search results so that future search can be more efficient, or perhaps even eliminated.

Although search and memory are key computational elements of any reinforcement learning algorithm, it is best to define reinforcement learning in terms of a particular class of learning *problems*; not particular algorithms. Any algorithm that is well suited to solving one of these problems we consider to be a reinforcement learning algorithm. In Chapter 2 we present a precise formulation of reinforcement learning problems, drawing heavily on the mathematical definition of a Markov decision process. Although this formulation allows us to take advantage of a great wealth of existing mathematical theory, our primary intent it to provide a fairly straightforward representation of the real problem facing a learning agent interacting with its environment to achieve a goal (or to achieve multiple goals). Clearly, such an agent must be able to sense information pertinent to the state of its environment and must be able to take actions that affect the state. The agent must also have a goal, or goals, defined in terms of how the environment behaves over time under the influence of its actions. These three aspects—sensation, action, and goal—are the building blocks of the theoretical framework that we use throughout this book.

Reinforcement learning is very different from supervised learning, the kind of learning studied in almost all current research in machine learning, statistical pattern recognition, and artificial neural networks. Supervised learning is learning under the tutelage of a knowledgeable supervisor, or "teacher," that explicitly tells the learning agent how it should respond to training inputs. Although this kind of learning can provide an important component of more complete learning systems, it is not by itself adequate for the kind of learning that autonomous agents must accomplish. It is often very costly, or even impossible, to obtain a set of examples of desired behavior that is both correct and representative of the situations in which the agent will have to act. In uncharted territory—where one would expect learning to be most beneficial—an agent must be able to learn from its own experiences rather than from a knowledgeable teacher. Although a reinforcement learning agent might also take advantage of knowledgeable instruction if it is available, the primary source of information and feedback is this interaction with its environment.

One of the challenges that arises in reinforcement learning, and not in other kinds of learning, has been called 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, while 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. The agent must try a variety of actions *and* progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to reliably estimate its expected reward. The exploration–exploitation dilemma has been intensively studied by mathematicians for many decades. We simply note that the entire issue of balancing exploitation and exploration does not even arise in supervised learning, as it is usually defined.

Another key feature of reinforcement learning is that it explicitly considers the *whole* problem of a goal–directed agent interacting with an uncertain environment. This is in contrast with many approaches that address subproblems without addressing how they might fit into a larger picture. For example, we have mentioned that much of machine learning research is concerned with supervised learning without explicitly specifying how such an ability would finally be useful. Other researchers have developed theories of planning with general goals, but without considering planning's role in real–time decision–making, or the question of where the predictive models necessary for planning would come from. Although these approaches have yielded many useful results, it is clear that their focus on isolated subproblems has become a significant limitation.

Reinforcement learning takes the opposite tack by starting with a complete, interactive, goal–seeking agent. All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. Moreover, it is usually assumed from the beginning that the agent will have to operate despite significant uncertainty about the environment it faces. When reinforcement learning involves planning, it has to address the interplay between planning and real–time action selection, as well as the question of how environmental models are acquired and improved. When reinforcement learning involves supervised learning, it does so for very specific reasons that determine which capabilities are critical, and which are not. For learning research to make progress, important subproblems surely have to be isolated and studied, but they should be subproblems that are motivated by clear roles in complete, interactive, goal–seeking agents, even if all the details of the complete agent cannot yet be filled in.

4 Components of a Reinforcement Learning Agent

A reinforcement learning agent generally consists of four basic components: a *policy*, a *reward function*, a *value function*, and a *model of the environment.*

The *policy* is the decision–making function of the agent, specifying what action it takes in each of the situations that it might encounter. In psychology, this would correspond to the set of stimulus–response rules or associations. This is the core of a reinforcement agent, as suggested by Figure 1, because it alone is sufficient to define a full, behaving agent. The other components serve only to change and improve the policy. The policy itself is the ultimate determinant of behavior and performance. In general it may be stochastic.

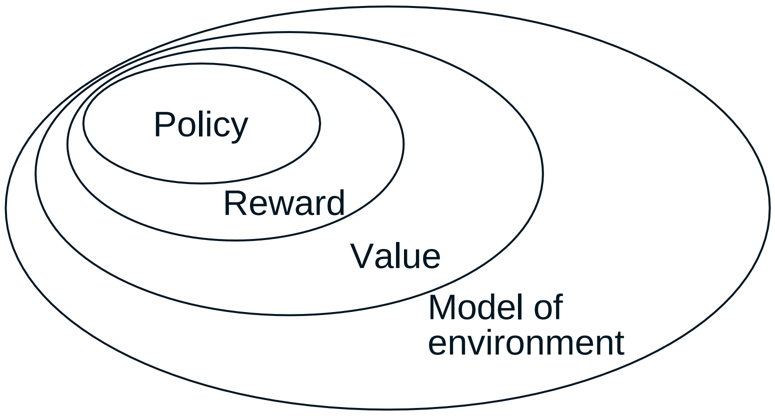

Figure 1: Main Components of a Reinforcement Learning Agent.

The *reward function* defines the goal of the reinforcement learning agent. The agent's objective is to maximize the reward that it receives over the long run. The reward function thus defines what are the good and bad events for the agent. Rewards are the immediate and defining features of the problem faced by the agent. As such, the reward function must necessarily be fixed. It may, however, be used as a basis for changing the policy. For example, if an action selected by the policy is followed by low reward then the policy may be changed to select some other action in that situation in the future.

Whereas reward indicates what is good in an immediate sense, the *value function* specifies what is good in the long run, that is, because it predicts reward. The difference between value and reward is critical to reinforcement learning. For example, when playing chess, checkmating your opponent is associated with high reward, but winning his queen is associated with high value. The former defines the true goal of the task—winning the game—whereas the latter just predicts this true goal. Learning the value of states, or of state–action pairs, is the critical step in the reinforcement learning methods we consider here.

The fourth and final major component of a reinforcement learning agent is a *model* of its environment or external world. This is something that mimics the behavior of the environment in some sense. For example, given a situation and an action, the model might predict the resultant next state and next reward. The model is drawn as the largest component in Figure 1 because one might expect it to take up the most storage space. If there are $|\mathcal{S}|$ states and $|\mathcal{A}|$ actions, then a complete model will take up space proportional to size $|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|$, because it maps state–action pairs to probability distributions over states, giving the probability of each possible result state for each action taken in each state. By contrast, the reward and value functions might just map states to real numbers, and thus be of size $|\mathcal{S}|$, while a stochastic policy is at most of size $|\mathcal{S}| \times |\mathcal{A}|$.

Not every reinforcement learning agent uses model of the environment. Methods that never learn or use a model are called *model–free* reinforcement learning methods. Model–free methods are very simple and, perhaps surprisingly, are still generally able to find optimal behavior. Model–based methods just find it faster (with fewer experiences). The most interesting case is that in which the agent does not have a perfect model of the environment a priori, but must use learning methods to align it with reality.

5 Summary

In this chapter we sketched some of the reasons that growing numbers of researchers are paying attention to reinforcement learning. First, reinforcement learning focuses on learning online during normal interaction with a dynamic environment. This contrasts with the focus of much of machine learning, both symbolic and artificial neural network, on systems that learn offline from a pre–specified set of training examples provided by an explicit and knowledgeable "teacher". Although a reinforcement learning system should also be able take advantage of knowledgeable teachers in its environment, if there are any, its real source of information is its own experience. Moreover, most machine learning systems do not learn while they are actually being used. It is more appropriate to call them *learned* systems rather than *learning* systems. Although reinforcement learning is sometimes used in this way, the conceptual bedrock of reinforcement learning is that a learning system should use *all* of its experience, throughout its entire existence, to improve its performance.

Reinforcement learning uses a formal framework defining the interaction between agent and environment in terms of situations (states), actions, and rewards. This framework is intended to be a simple way of representing essential features of the AI problem. These features include a sense of cause and effect, of uncertainty and nondeterminism, and the existence of explicit goals. Most relevant is the formalism of Markov decision processes, which provides a precise, and relatively neutral, way of including the key features. Although we only scratch its surface, this theory allows us to take advantage of related perspectives and methods developed in field of stochastic optimal control.

The concepts of value and value functions are the key features of the kinds of reinforcement learning methods that we consider in this book. We take the position that value functions are essential for efficient search in the space of policies. Their use of value functions distinguish these reinforcement learning methods from conceptually simpler evolutionary methods that search directly in policy space, guided by scalar evaluations of entire policies. In our approach, value functions enable algorithms to take advantage of the details of individual behavioral interactions. Although evolutionary methods may provide useful results for some problems, and value function methods can profitably be used in conjunction with them (much the way learning and evolution work together in nature), we believe that when applied to reinforcement learning problems, evolutionary methods are inherently less efficient than value function methods.

Once one takes the estimation (learning) of value functions as a key computational step, the question becomes how best to do this. In this book we identify three principle classes of methods. *Monte Carlo* methods estimate the value of a state by simply running many trials starting at that state. The actual total rewards received on those trials are then averaged to obtain an estimate of the state's value. *Search* methods such as dynamic programming and heuristic search can be viewed as straightforward model–based ways to estimate a value function. Finally, *temporal–difference* methods, a relatively new development, are based on learning states' values by using the values of the states that follow them in actual trials.

This book is organized around the principle that these three classes of methods for learning value functions are not totally different: they can be viewed as members of one "super family" of methods. Although there are differences between them, one need not pick between them, but can mix and match. They all have at their heart a common operation, called a "backup." Some perform backups based on experience, some based on a model, some backup from a wide set of possible next states, some only from one. The backups are of different sizes, shapes, and sources, but they all share common features and contribute to a common computation.

6 Bibliographical and Historical Remarks

Here we provide a necessarily abridged discussion of the history of the main ideas of reinforcement learning. Although the specific term "reinforcement learning" has never been used by psychologists, the roots of reinforcement learning lie in the learning theories developed by experimental psychologists throughout this century. It would take us too far afield to attempt an overview of the reinforcement theories of psychology, something that is already available in many books (e.g., (Mackintosh, 1983)). We concentrate instead on the best–known of the early explorations of the computational power of reinforcement learning, trying not to obscure the fact that computational and psychological perspectives are sometimes hard to distinguish.

In the 1960s one finds the terms "reinforcement" and "reinforcement learning" being in the engineering literature for the first time (Minsky thesis? (Minsky, 1961); Waltz & Fu (Waltz & Fu, 1965), Mendel, 1966; Mendel & McClaren (Mendel & McLaren, 1970)). These terms are used to refer to the general idea of learning from rewards and punishments: trial–and–error learning, where actions followed by good or bad outcomes are respectively strengthened or weakened.

This early notion of reinforcement learning is an exact mirror of a classical psychological principle, Thorndike's (1911) "Law of Effect":

> "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond."

Although this principle has generated considerable controversy in psychology, as well as in other fields, over the years (see ref ???), it remains influential because its general idea is supported by many experiments and it makes such good intuitive sense. It is an elementary, obvious way of combining search and memory: search in the form of trying many actions, and memory in the form of remembering what actions worked best. Dennett (**?**) provides a good account of the continuing attractiveness of the Law of Effect, and Cziko (**?**) provides a very broad account of the utility of methods, like the Law of Effect, that operate using *selectional*, as opposed to *instructional*, principles.

The earliest computational investigations of the Law of Effect that we know of were by Minsky and by Farley and Clark, both published in 1954. In his Ph.D. dissertation, Minsky (Minsky, 1954) describes the construction of an analog machine, the SNARC (Stochastic Neural–Analog Reinforcement Calculator) which was designed to to learn by trial–and–error. Farley and Clark ((Farley & Clark, 1954); Clark and Farley, 1955) describe another neural–network learning machine, but noting its ability to "generalize", discussed it more in terms of supervised learning than of reinforcement learning. This began a pattern of confusion about the relationship between these types of learning. Many researchers seemed to believe that they were studying reinforcement learning, while they were actually studying supervised learning, a confusion that persists to this day. Even modern neural–network textbooks often describe networks that learn from training examples as trial–and–error learning systems because they use error information to update connection weights. While this is an understandable confusion, it substantually misses the selectional character of of learning via the Law of Effect, which is what the term trial–and–error was originally intended to describe.

It is clear that neural–network pioneers such as Rosenblatt (1958, (Rosenblatt, 1961)) and Widrow and Hoff (Widrow & Hoff, 1960), as well as the psychologists Bush and Mosteller (Bush & Mosteller, 1955), were thinking about reinforcement learning—they used the language of rewards and punishments—but the systems they studied became more clearly supervised learning systems, suitable for pattern recognition and perceptual learning, but not for direct interaction with an environment. In the 1960s and 1970s, reinforcement learning was gradually overshadowed and lost as a distinct topic, while supervised learning, particularly in the form of pattern recognition, became widely studied. We discuss some of the fine points of this transition, including exceptions such as learning automata theory and Kiefer–Wolfowitz stochastic approximation methods in Chapter 3.

Other clear exceptions to this trend deserve mention here. In 1963, Andreae described a reinforcement learning machine called STeLLA (?), which included what we would now call an environment model to facilitate learning. Andreae was explicitly concerned with how a machine could learn by interacting with its environment. Later developments included an "internal monologue" to deal with the problem of partial state observablility (?), something that is still an important issue for reinforcement learning. Although later work by Andreae continued to emphasize learning by interaction, it placed more emphasis on the role of a teacher (?). Andreae's pioneering research is not well–known, but it still holds lessons for modern reinforcement learning research.

Also in the 1960s, Donald Michie maintained a clear focus on reinforcement learning. He described a simple reinforcement learning system for learning how to play Tic-Tac-Toe (also known as Noughts and Crosses) called MENACE (for Matchbox Educable Noughts and Crosses Engine) (?; ?). It consisted of a matchbox for each possible game position containing a number of colored beads, a color for each move available from that position. By drawing a bead at random from the matchbox corresponding to the current game position, one could determine MENACE's move. When a game was over, beads were added or removed from the boxes used during play to reinforce or punish MENACE's decisions. We would now regard MENACE as a collection of simple stochastic learning automata (Chapter 3). In 1968, Michie and Chambers (Michie & Chambers, 1968) described a more advanced Tic-Tac-Toe reinforcement learner called GLEE (Game Learning Expectimaxing Engine) which estimated a value function using what they called Expectimaxing. We would now recognize this as closely related to dynamic programming. Michie's Tic-Tac-Toe players served as inspiration for our Tic-Tac-Toe example of this chapter, and their discussion of how decomposing a large problem into a number of mutually independent sub–problems can lead to efficient reinforcement learning influenced our discussion in which we contrasted value–function methods and evolutionary methods. (The latter methods do not decompose problems in this way.)

Michie and Chambers (Michie & Chambers, 1968) also described a more advanced version of the MENACE approach, implemented in a system called BOXES, which they applied to the problem of learning to balance a pole hinged to a movable cart on the basis of a failure signal occuring only when the pole fell or the cart reached the end of a track. This work was partially inspired by the 1964 pole–balancing system of Widrow and Smith (Widrow & Smith, 1964), which learned via supervised learning from a teacher already able to accomplish the task. (Comparing the pole–balancing system of Widrow and Smith with that of Michie and Chambers is a good very way to come to appreciate the difference between supervised and reinforcement learning.) BOXES, which did not estimate a value function, inspired the pole–balancing system of Barto, Sutton, and Anderson (Barto *et al.*, 1983) which did estimate a value function. (These systems have been followed by other pole–balancing reinforcement learning systems too numerous to mention.)

Although Widrow and colleagues maintained a clear emphasis on supervised learning, he recognized how it differed from reinforcment learning. In 1973, Widrow, Gupta, and Maitra (Widrow *et al.*, 1973) modified the Widrow–Hoff supervised learning rule (often called the Least–Mean–Square, or LMS, rule) to produce a reinforcement learning rule which could learn from success and failure signals instead of from training examples. They called this form of learning "selective bootstrap learning" and used the phrase learning "learning with a critic" instead of "learning with a teacher."

Another researcher who bucked the supervised learning trend was Harry Klopf (Klopf, 1972; Klopf, 1982), who presented a "hedonic", or "heterostatic," theory of neural function and AI. Klopf recognized that essential aspects of adaptive behavior were being lost as learning researchers came to focus almost exclusively on supervised learning. These were the hedonic aspects: the drive to achieve some result from the environment, to control it toward desired ends and away from undesired ends. This is of course an essential element of reinforcement learning. Klopf's work was a especially important to the authors because our assessment of Klopf's ideas (Barto & Sutton, 1981) led to our appreciation of the distinction between supervised and reinforcement learning, as well as to our eventual focus on reinforcement learning.

Important contributions to reinforcement learning were also made by John Holland (Holland, 1975; Holland, 1986). Although best known for his development of genetic algorithms, Holland outlined a very general theory of adaptive systems that stresses interactive learning based on selectional principles. In fact, his classifier system (Holland, 1986) is a reinforcement learning system that updates value functions using what he called the "bucket brigade algorithm," which is closely related to the value–function estimation methods we discuss in this book. Although genetic algorithms are natural candidates for implementing what we called the evolutionary approach to reinforcement learning, which we contrasted with value–function methods, Holland did not suggest this approach. Classifier systems use both genetic algorithms and value functions.

Although the idea of learning by estimating value functions from experience appeared in Minsky's dissertaton (Minsky, 1954), it was most influentially introduced by Samuel (Samuel, 1959) in his program for learning how to play the game of checkers using what we would now call a temporal difference method. We consider Samuel's work to be a seminal influence on the approach to reinforcement learning that we present in this book. His papers (Samuel, 1959; Samuel, 1967) reveal extraordinary insight into nearly all the issues that are still challenging current researchers. These papers make highly worthwhile reading even today, and we have much more to say about Samuel's checkers player in later chapters.

Although Minsky's dissertaion was an early foray into reinforcement learning, much more influential was his 1960 paper "Steps Toward Artificial Intelligence," which, like Samuel's papers, contains cogent discussions of issues that are still relevant to modern reinforcement learning. Of particular note is Minsky's discussion of what he considered to be the major computational problem that complex reinforcement learning systems would have to solve to be successful. He called this the *credit–assignment problem*:

> In applying such methods to complex problems, one encounters a serious difficulty—in distributing credit for success of a complex strategy among the many decisions that were involved.

If the many decisions involved in achieving success are to be reinforced, then their relative contributions to that achievement have to be assessed. Minsky discussed the value–function estimation method used in Samuel's checkers player as an important approach to this problem, pointing out that it is closely related to the phenomenon of conditioned reinforcement that occurs in animal learning. All of the methods we discuss in this book are directed toward making credit–assignment less of a problem for reinforcement learning systems.

It is clear from the brief account above that the main ideas of reinforcement learning have been present in AI since its earliest days. However, it is only relatively recently that they have been attracting widespread attention. One of the reasons that reinforcement ideas have historically had little influence in AI is their association with behaviorist views of learning and intelligence. AI research in 1960s followed the allied areas of psychology in shifting from approaches based in animal behavior toward more cognitive approaches, leaving little room for reinforcement theories; in fact, leaving little room for learning theories of any kind. Although we agree with the critics who have argued that one cannot understand or generate *all* intelligent behavior on the basis of reinforcement principles alone, we believe that AI systems and cognitive theories that steer clear of these basic learning principles are handicapped as well. Indeed, the climate has grown considerably warmer toward classical learning principles, including reinforcement learning, because researchers are using them in systems that owe as much to cognitive perspectives as they do to earlier theories of animal behavior.

A related factor that limited the influence of reinforcement learning principles in AI is the reputation that they are too computationally weak to be of much use. However, there is now ample evidence that reinforcement learning can be very powerful. Some of the most impressive accomplishments of artificial learning systems have been achieved using reinforcement learning.

More is needed on history and bibliography: Watkins, Campbell and Craik and Dennett (models), Witten!, Werbos!, our own early stuff. Heuristic Search, Booker, Hampson. Bellman and Howard. Also add Minsky & Selfridge (1963) quote: "in a novel situation, try methods like those that have worked best in similar situations"

References

Barto, AG & Sutton, RS (1981). Goal seeking components for adaptive intelligence: An initial assessment. Technical Report AFWAL-TR-81-1070 Air Force Wright Aeronautical Laboratories/Avionics Laboratory Wright-Patterson AFB, OH.

Barto, AG, Sutton, RS, & Anderson, CW (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics,* **13**, 835–846. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, MA, 1988.

Bush, RR & Mosteller, F (1955). *Stochastic Models for Learning.* New York: Wiley.

Farley, BG & Clark, WA (1954). Simulation of self-organizing systems by digital computer. *IRE Transactions on Information Theory,* **4**, 76–84.

Holland, JH (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Holland, JH (1986). Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In: *Machine Learning: An Artificial Intelligence Approach, Volume II,* (RS Michalski, JG Carbonell, & TM Mitchell, eds) pp. 593–623. San Mateo, CA: Morgan Kaufmann.

Klopf, AH (1972). Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164 Air Force Cambridge Research Laboratories Bedford, MA.

Klopf, AH (1982). *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence.* Washington, D.C.: Hemishere.

Mackintosh, NJ (1983). *Conditioning and Associative Learning.* New York: Oxford University Press.

Mendel, JM & McLaren, RW (1970). Reinforcement learning control and pattern recognition systems. In: *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, (JM Mendel & KS Fu, eds) pp. 287–318. New York: Academic Press.

Michie, D & Chambers, RA (1968). BOXES: An experiment in adaptive control. In: *Machine Intelligence 2*, (E Dale & D Michie, eds) pp. 137–152. Oliver and Boyd.

Minsky, ML (1954). *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem.* PhD thesis Princeton University.

Minsky, ML (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers,* **49**, 8–30. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought.* McGraw-Hill, New York, 406–450, 1963.

Rosenblatt, F (1961). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* 6411 Chillum Place N.W., Washington, D.C.: Spartan Books.

Samuel, AL (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development,* 210–229. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, McGraw-Hill, New York, 1963.

Samuel, AL (1967). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal on Research and Development,* 601–617.

Waltz, MD & Fu, KS (1965). A heuristic approach to reinforcment learning control systems. *IEEE Transactions on Automatic Control,* **10**, 390–398.

Widrow, B, Gupta, NK, & Maitra, S (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Transactions on Systems, Man, and Cybernetics,* **5**, 455–465.

Widrow, B & Hoff, ME (1960). Adaptive switching circuits. In: *1960 WESCON Convention Record Part IV* pp. 96–104,.

Widrow, B & Smith, FW (1964). Pattern-recognizing control systems. In: *Computer and Information Sciences (COINS) Proceedings* , Washington, D.C.: Spartan.