Looking Back on the Actor–Critic Architecture

Andrew G. Barto, *Life Fellow, IEEE*, Richard S. Sutton, and Charles W. Anderson, *Senior Member, IEEE*

***Abstract*—This retrospective describes the overall research project that gave rise to the authors' paper "Neuronlike adaptive elements that can solve difficult learning control problems" that was published in the 1983 Neural and Sensory Information Processing special issue of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. This look back explains how this project came about, presents the ideas and previous publications that influenced it, and describes our most closely related subsequent research. It concludes by pointing out some noteworthy aspects of this article that have been eclipsed by its main contributions, followed by commenting on some of the directions and cautions that should inform future research.**

***Index Terms*—Actor–critic, hedonistic neuron, pole blancing, reinforcement learning (RL), temporal-difference learning.**

I. INTRODUCTION

WHEN our paper "Neuronlike adaptive elements that can solve difficult learning control problems" was published in the 1983 Neural and Sensory Information Processing special issue of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS [1], there was no way to predict that it would have a lasting influence, or that reinforcement learning (RL), the class of machine learning methods to which it contributed, would now be one of the most active areas of artificial intelligence. In this retrospective, we describe the overall research project that gave rise to our paper, explain how this project came about, present the ideas and previous publications that influenced it, and describe our most closely related subsequent research. We do not, however, attempt to do justice to earlier related work of which we were unaware in 1983; much of that can be found in [2]. We end by pointing out some noteworthy aspects of this article that have been eclipsed by its main contributions, followed by commenting on some of the directions and cautions that should inform future research.

In the late 1970s and early 1980s, we had the opportunity to participate in a research project aimed at assessing the scientific merit of a hypothesis proposed by physiologist A. Harry Klopf, a senior scientist with the Avionics Directorate of the Air Force Office of Scientific Research (AFOSR). Klopf was dissatisfied with the great importance attributed to equilibrium-seeking processes for explaining natural intelligence and for providing a basis for machine intelligence. These include homeostasis and error-correction learning methods for pattern classification. He argued that systems that try to maximize something (whatever that might be) are qualitatively different from equilibrium-seeking systems, and he further argued that maximizing systems hold the key to understanding important aspects of natural intelligence and for building artificial intelligences. In particular, Klopf hypothesized that neurons, the major components of our brains, are individually "hedonists" that work to maximize a neuron-local analog of pleasure while minimizing a neuron-local analog of pain [3], [4].

A project with the goal of assessing Klopf's ideas to determine if they were novel and if they were worth pursuing, was funded through an AFOSR contract to principal investigators Michael Arbib, William Kilmer, and Nico Spinelli, professors at the University of Massachusetts Amherst and founders of the Cybernetics Center for Systems Neuroscience, a farsighted center focusing on the intersection of neuroscience and artificial intelligence. Andrew Barto, a recent Ph.D. from the University of Michigan, was hired as a post-doc in 1977, shortly joined by graduate students Richard Sutton and Charles Anderson, who later received Ph.D.s under Barto's direction after he became a UMass faculty member. It was our good fortune that the project's funding and its PIs gave us wide latitude to explore the study of learning in artificial intelligence, including its early history, its connections to experimental data and theories of animal learning from psychology, and its connections to data and theories about the neural basis of learning from neuroscience.

II. MINIMIZING OR MAXIMIZING?

Klopf argued that a system that attempts to maximize a quantity is distinctly different from one that attempts to minimize a quantity, such as a system that seeks stability by minimizing the difference between its current state and a desired state. He coined the term "heterostat" to distinguish a maximizing system from a "homeostat", Ashby's term for a system that maintains stability in a changing environment [5].

There is, of course, no mathematical difference between minimizing and maximizing (just change the sign), but one of the first things we realized was that there is a qualitative difference between being directed by a signed error vector, indicating, for example, the difference between a current and a

Manuscript received November 15, 2020; accepted November 24, 2020. Date of publication December 24, 2020; date of current version January 12, 2021. This article was recommended by Associate Editor D. Liu. *(Corresponding author: Andrew G. Barto.)*

Andrew G. Barto is with the College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01003 USA (e-mail: barto@cs.umass.edu).

Richard S. Sutton is with the Department of Computing Science, University of Alberta, Edmonton, AB T6G 2R3, Canada, and also with DeepMind, Edmonton, AB, Canada.

Charles W. Anderson is with the Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA.

Digital Object Identifier 10.1109/TSMC.2020.3041775

desired state, and being directed by scalar evaluations that—in themselves—do not indicate a direction of improvement.

This turns out to be a major distinction between supervised learning, which is fundamentally an equilibrium-seeking process (zeroing out errors) and RL, in which the learner has to do more work to determine how it should change in response to its experiences. An equilibrium-seeking process can stop when equilibrium is attained (zero error), but an evaluation-driven system, not knowing what evaluation is best, has to be incessantly active, continually exploring for directions of improvement. Both error correction and RL are optimization processes, but error correction is a restricted special case where RL is more general. It seemed to us that Klopf's intuition about the relevance of this to intelligence might be on the mark.

III. Initial Progress

From our exploration of earlier research on building artificial learning systems, we came to a rather surprising conclusion. Despite the prominence of evaluation-driven learning in some of the earliest artificial learning systems, this form of learning had been largely overshadowed by error-correcting learning due to supervised learning's ability to learn to recognize patterns by being exposed to training examples. There were clear and important exceptions, but this sparsity of earlier research on what we now call RL confirmed Klopf's contention that something was missing from approaches to machine learning that were then current. This encouraged us as we launched into the subject. An account of this history is beyond our scope here but can be found in [2, Sec. 1.7] and in [6].

There was also an important form of neglect in psychology in that studies of animal learning became unfashionable, supplanted by the "cognitive revolution." In particular, learning from the rewarding and punishing consequences of behavior, studied as instrumental conditioning in psychology, suffered from neglect, despite always being seen as a key principle of learning. This principle was famously stated in Thorndike's "Law of Effect" [7], which says that if an animal's response to a situation is closely followed by the animal's satisfaction, then that response becomes more strongly connected to the situation and is, therefore, more likely to be produced when the animal faces that situation again; conversely, if a response is followed by discomfort, the connection is weakened, making the animal less likely to produce the response when that situation recurs. This "law" has endured to the present, though not without much revision and controversy. It describes the common sense process of learning by trial and error (though it is misleading to equate the word error with the error vectors of supervised learning).

Klopf's idea of hedonistic neurons was that neurons implement a neuron-local version of the law of effect. He hypothesized that the synaptic weights of neurons change with experience according to the following. When a neuron fires an action potential, all the synapses that were active in contributing to the action potential become eligible to undergo changes in their efficacies. If the action potential is followed within an appropriate time period by an increase in reward, the efficacies of all eligible synapses then increase (or decrease in the case of punishment). In this way, synapses change so as to alter the neuron's firing patterns in the service of increasing the neuron's probability of being rewarded, and decreasing its probability of being penalized, by its environment.

In Klopf's hypothesis, reward and punishment were delivered to a neuron via the same inputs that excited or inhibited its electrical activity. He objected to the idea that there is a single specialized reward signal that drives learning. Our algorithms departed from this by using a single specialized input to deliver rewards, but we did not completely discount his objection to it, as we discuss in Section- III-B below.

The actor–critic architecture presented in our 1983 paper brought together two lines of research that we had been pursuing from the beginning of the AFOSR project. One line focussed on developing a neuron-like adaptive element following Klopf's idea of a hedonistic neuron. We called this an associative search element, or ASE, later to be known as the actor component of the actor–critic architecture. The other line of research focused on issues of signal timing and prediction. This line led to the neuron-like adaptive element we called the adaptive critic element, or ACE, which became the other main component of the actor–critic architecture. This component was the source of the reward and penalty signals evaluating the actions of the ASE.

We discuss the ASE and ACE before discussing their combination in the actor–critic architecture. Both elements were "neuron like" in the same abstract way that McCulloch-Pitts' formal neurons were [8]. We decided that trying to model real neurons in any detail would distract us from the project's main computational objective.

A. Associative Search Element

The idea of storing information distributed across large areas of a physical structure had gained prominence by the late 1970s for both its computational promise and as a model of how information might be stored in brain (e.g., [9]–[11]). Called associative memories, the simplest were based on correlation matrices, and storing information consisted of presenting "keys" paired with "patterns" to store key-pattern associations. As a learning process, this was supervised learning because the desired pairings of keys and patterns were explicitly provided to the memory systems, though the systems were able to generalize beyond these training pairings as in pattern recognition uses of supervised learning.

We decided that an RL version of an associative memory would be a good way to illustrate the difference between RL and supervised learning. Instead of being given the desired patterns to be associated with the keys, the RL associative memory network had to search for the pattern that maximized an externally supplied reward signal. As this kind of learning proceeded, each key tended to cause the retrieval of better—more rewarding—choices for the pattern to be associated with it. The only part of the system having prior knowledge about what associations were best was the evaluator, or critic, which computed the reward signal.

Details of this network and some simulation results were presented by Barto *et al.* in 1981 [12]. In one example, the network was a single layer of 25 ASEs. Input keys were 8-D vectors delivered to each ASE; output patterns consisted of the binary responses of each ASE, and each ASE received the same reward signal. We called this an associative search network (ASN). The ASEs' learning algorithm generally followed Klopf's hypothesis about how the synaptic weights of neurons might be adjusted by an RL process, but we borrowed from earlier RL algorithms known as stochastic learning automata. These algorithms originated with the work of Tsetlin [13] (reviewed in [14] and [15]) and were not well known in artificial intelligence circles. We also borrowed from Tzanakou and Harth's Alopex method that these authors proposed as a stochastic model of the development of visual receptive fields [16], [17].

Stochastic learning automata and the Alopex method use randomness to search among alternative actions for those that deliver the most reward. Actions are selected according to probabilities that are altered on the basis of reward feedback so as to allocate more probability to higher performing actions. Actions have to be tried out to find out how they perform. This process is *selectional* in the same way that natural evolution is selectional in favoring higher fitness organisms. We added a random number to the activation level of each ASE so that it randomly tried out all of its actions and biased the random selection toward actions yielding more reward. Randomness was essential to provide the variety needed to drive the search, just as animal populations need variety to drive evolution.

ASEs differed from stochastic learning automata, and from the Alopex method, in an important respect. Stochastic learning automata did not normally receive input other than the reward signal, whereas all the ASEs making up the ASN received input vectors coding the associative memory keys in addition to the reward signal. As learning continued, the input keys became associated with better and better output patterns as scored by the reward signal. Where a stochastic learning automaton attempted to find a single best action, an ASE attempted to find the best action for each input key. In this respect, an ASE worked more like the law of effect in forming connections between situations, here the keys, and responses, here the associated patterns.

In more theoretical terms, a stochastic learning automaton faces a multiarmed bandit problem (e.g., [15] and [18]). An ASE, and therefore, the ASN as well, faced what we called an "associative search problem," now commonly referred to as a "contextual bandit problem." This problem involves remembering, in the form of associative links, the results of conducting multiple searches. It is, therefore, closely related what computer science calls "memoization," which is the process of saving results of a calculation in memory so that results that have been calculated previously can be retrieved from memory instead of being calculated again [19], [20]. In RL, the calculation is an ongoing search for higher rewarding actions. Consequently, at its base, RL is a kind of *contextual memorized search*.

It remained for us to decide how good and bad evaluations would be represented for delivery to the ASEs making up the ASN. A natural way to do this was to use *changes* in a scalar reward signal for adjusting each ASE's synaptic weights, instead of using reward signal itself: a reward increase made the element's response more likely in the present context; a reward signal decrease made it less likely. These reward signal changes were the reinforcement signals that directed changes in each ASE's synaptic weights; not the reward signal itself.[1]

It was necessary to include in the ASN a special reward-predictor element that learned to predict the amount of reward an ASE should expect when acting in the current context. This enabled reward signal changes to act correctly as reinforcement. In our simulations the context vectors—the keys—changed randomly at each time step. The network's output influenced the immediate context-dependent reward signal, but it had no influence on what key would be presented next (unlike the situation in our later actor–critic simulations). We had to prevent a change in reward due to a random change in the key from being attributed to the element's action. To do this, the reward change was computed by comparing the current reward with the reward expected when acting in the current context. This foreshadowed the ACE's role in the actor–critic architecture.

B. Adaptive Critic Element

We chose the term *critic* for the ACE component of the actor–critic architecture after Widrow *et al.*'s use of this term to contrast "learning with a critic" from "learning with a teacher," as supervised learning is often called [21]. A teacher provides the learner with desired or correct actions, whereas a critic merely evaluates a learner's actions. Evaluations might be the result of comparing the learner's actions with desired actions (as was the case for our ASN simulations), in which case evaluations are based on how much the learner's actions differ from the desired actions. However, evaluations do not need to be based on any knowledge of what the desired actions should be. In fact, the critic does not even need to have access to the learner's actions; it can base its evaluations on the consequences of those actions on the learner's environment. The pole-balancing control problem of our 1983 paper illustrates this because the ACE evaluates the behavior of the cart–cart system, not the ASE's actions, to which it does not have access.

We included the term *adaptive* in the ACE because it was a learning system itself, capable of learning to make more informative evaluations. In the case of the actor–critic architecture, this meant learning to evaluate the long-term performance of the actor by learning to predict how much reward would be expected to accrue over the future. The predictions themselves then became the reinforcing input to the actor.

The inspiration for the ACE came from animal learning psychology, in particular, from classical, or Pavlovian, conditioning. This form of learning enables an animal to act in

[1]The terms reward and reinforcement are sometimes used interchangeably, but we distinguish between them. An RL system's reward signal sets the learner's objective, which is to maximize the amount of reward received over time. Reinforcement, on the other hand, is the quantity that an RL learning rule uses to adjust the parameters determining its action probabilities.

anticipation of upcoming inputs from its environment, allowing the animal to prepare for, or to avoid, those inputs. The animal effectively learns to predict aspects of its future. The feature of classical conditioning most relevant to the actor–critic architecture is the phenomenon of higher-order conditioning (and the similar phenomenon of secondary reinforcement in instrumental conditioning). This occurs when events that predict the arrival of reward become rewarding themselves. Higher order conditioning and secondary reinforcement remove the need to wait until a final reward or penalty is received in order to learn.

A few years before our 1983 actor–critic paper, we presented the basics of the ACE algorithm as a model of classical conditioning [22], [23], which subsequently evolved into temporal difference (TD) algorithms. Sutton's 1984 Ph.D. dissertation [24] developed TD algorithms further, and Sutton's 1988 paper [25] extended the theoretical treatment, laying the groundwork for the extensive development that followed within the modern RL framework, e.g., [2].

The name TD derives from the algorithm's use of changes, or differences, in predictions over successive time steps to drive the learning process. The prediction at any given time step is updated to bring it closer to the prediction of the same quantity at the next time step. It is a self-supervised learning process that works to reduce errors between current and later predictions, taking intervening incoming data into account. Used in RL, TD algorithms learn to predict a measure of the total amount of reward expected over the future.

In employing a TD algorithm, the ACE of the actor–critic architecture addressed two requirements for successful learning. One requirement was to address the "delayed reward problem," which is when the relevant consequences of an action occur after some nontrivial time interval, making it difficult to assign credit or blame to the appropriate action, or actions, of the learner. The other requirement was to provide an appropriate reinforcement signal to the ASE. Both of these functions were clearly illustrated in the pole-balancing problem tackled in our 1983 paper.

The ACE also went part of the way toward addressing Klopf's rejection of a single unitary reward signal in his hedonistic neuron hypothesis. He argued that whatever generated this signal would have to be so intelligent itself that assuming its existence would beg the question of how intelligence arises. Klopf proposed what he called "generalized reinforcement" as a way to avoid a unitary reward signal. The TD idea is not unrelated. It uses ordinary (nonreward) input to play an important role in rewarding action. The actor is trying to maximize the excitation of the critic as well as maximize the reward, thus a kind of generalized reinforcement. But unlike Klopf's desire to eliminate a reward signal altogether, TD learning is tied ultimately to reward, which is necessary in order to create a well-defined optimization problem.

IV. Pole Balancing

Our 1983 paper featured the problem of learning to balance a pole hinged onto a movable cart. This idea came from our discovery of the 1968 paper by Michie and Chambers entitled "BOXES: An Experiment in Adaptive Control" [26]. BOXES was a true RL system, and the paper's description of how it worked and the ideas underlying it helped shape our thinking about RL and how to explain our research. We thought that the version of the pole-balancing problem tackled by BOXES would provide a vivid illustration of the capabilities of the algorithms we had been working on, would clearly illustrate how RL differed from supervised learning, and would help to establish the utility of RL.

The BOXES pole-balancing task was adapted from the 1964 work of Widrow and Smith [27], who used supervised learning, assuming instruction from a teacher already able to balance the pole. But instead of receiving action-by-action instructions that could be copied, the sole training information available to BOXES was a failure signal when the pole fell past a certain angle or the cart hit the end of its track. This created a difficult delayed-reward problem (or delayed-penalty problem in this case) making the credit (or blame) assignment difficult.

In our earlier work with the ASN, described above, the pattern output of the network was evaluated by the critic, but the output pattern did not influence which key, or context vector, was presented next. The key presented at each step was selected uniformly at random from a finite set of keys. But in a more general setting, the RL system's actions would influence the stream of context inputs in addition to the critic's evaluations. Applying RL to a control problem like pole balancing was a natural way for the RL system's actions to influence the state of the system being controlled, with each state generating context input to the RL system. Furthermore, the critic's evaluations could be based on observing the behavior of the controlled system rather than on observing the RL system's actions themselves, thus illustrating an important property of RL. We therefore decided that a control problem would be an excellent testbed for our RL algorithms, and that Michie and Chambers' pole-balancing task would be a good place to start.

We followed the setup Michie and Chambers used in designing their BOXES system for the pole-balancing task. Like BOXES, our RL controller had no knowledge about the system being controlled, only receiving at each time step a vector describing the controlled system's state or a failure signal if the pole fell past a critical angle or the cart hit the end of the track. We used exactly their state representation, which was to divide the 4-D continuous state space into 225 "boxes" on the basis of the thresholds they selected, and to inform the controller which box the system's state was currently in. Using the picturesque "demon" terminology introduced into AI by Selfridge's 1959 Pandemonium program [28], Michie and Chambers described how BOXES worked like this:

> In order to envision how the $\cdots$ algorithm works it is easiest to imagine each one of the 225 boxes as being occupied by a local demon, with a global demon acting as a supervisor over all the local demons $\cdots$ Each local demon is armed with a left-right switch and a scoreboard. His only job is to set his switch

> from time to time in the light of data which he accumulates on his scoreboard [26, p. 148].

The rule used by the local demons was quite complicated but essentially depended on records for each control decision, left or right, of the number of decisions taken before a run failed.

It was easy to translate this organization into our neural network terms, ending up with the entire system being implemented by a single neuron-like element, in particular, by a single ASE. At each time step, the ASE's input vector would be one of 225 standard-unit-basis vectors: all zeros except a one in the position corresponding to the box occupied by the current state. Then, a local demon would correspond a synapse whose influence on the element's output would correspond to its left–right switch. Its scoreboard, then, would correspond to its synaptic efficacy, or connection weight. The global demon would correspond to the activation rule of the neuron-like element that would convert the active local demon's decision, plus a random number contribution, into the element's binary output by thresholding.

A side benefit of our neuron-like implementation was that it could easily accommodate state representations more complicated than the one used by BOXES. The full function approximation power of neural networks could be enlisted, including multilayer neural networks, as subsequent advances demonstrated. See Section VII below.

Due to the delayed-reward problem presented by the pole-balancing task, which was not present in our earlier work with the ASN, we modified the ASE learning rule by adding *eligibility traces*. Recall that according to Klopf's hedonistic neuron hypothesis, all the synapses that were active in contributing to the neuron firing would become eligible to undergo changes in their efficacies, with the changes happening if reinforcement arrived during the period of eligibility. He envisioned that eligibility would be implemented by the concentration of a synaptically-local chemical that began increasing when the synapse was active in firing the neuron, reached a maximum shortly after this, and thereafter decayed to zero after a time interval long enough to register delayed reinforcement. This concentration would be a trace of past activity called an eligibility trace.

We added eligibility traces to the ASE in the simplest way we could think of. Each eligibility-triggering event added to the ongoing trace at the appropriate synapse; otherwise that trace decayed exponentially with a time constant selected as a parameter of the simulation. The result was not too different from the records kept by the local demons of BOXES. Sutton and Barto [2] extensively discuss eligibility traces in the context of ideas for synaptic tags proposed by neuroscientists.

We conjectured that our ASE algorithm, with eligibility traces that decayed sufficiently slowly, would be superior to the BOXES algorithm. Our main reasoning was that Michie and Chambers did not seriously concern themselves with the necessity for variety in the controller's actions. In other words, BOXES did only very limited exploration. It was purely deterministic except for using pseudorandom numbers to break ties in selecting actions and for selecting the initial state for each learning trial, where a trial lasted from state reset until failure.[2]

An ASE, on the other hand, selected every action randomly, with probabilities adjusted as in stochastic learning automata. We thought that with eligibility traces lasting long enough to deal with the delayed-reward problem, a single ASE could learn faster than BOXES.

In the Summer of 1982, we saw an announcement for the "Neural and Sensory Information Processing" special issue of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS in which our 1983 paper would eventually appear. We thought this would provide an excellent opportunity to publicize our research—if we could make it to the deadline for submission. So we decided to implement our system along with BOXES with the intention of comparing their performances, thinking that our stochastic approach would easily surpass the performance of BOXES.

As the special issue deadline approached, we struggled to get our system to outperform BOXES, which worked better than we had expected. Very near the deadline, we decided to insert a TD algorithm into our pole-balancer in the form of the ACE. Sutton had been developing the TD idea that was to become a major part of his 1984 Ph.D. dissertation [24]. With the ACE providing reinforcement signals to the ASE, the system was able to learn better than BOXES could. This combination of the ASE and ACE became known as the actor–critic architecture.

But at almost literally the last minute before the special issue deadline, there was another setback: we discovered a bug in our implementation of the cart-pole simulation. Our procedure for updating the pole's angle with respect to the cart accepted as input the current pole angle expressed in radians, but returned the new angle in degrees.[3] Consequently, unbeknownst to us, our system had been learning to control a very different cart-pole system than we had intended, or that BOXES had learned to control in Michie and Chambers' paper. We quickly fixed the bug, but then struggled to tune our controller and simulation to produce effective learning. Finally, after an all-night session, we achieved adequate results and submitted this article.[4]

[2]Michie and Chambers explicitly avoided probabilistic decision making, stating that "such devices cannot be optimal," regarding the two-armed bandit problem as a "famous unsolved problem of mathematics." It was not until the 1970s that a version of the problem was solved with Gittins Indices [29], and we knew that stochastic learning automata could achieve ϵ-optimality with ϵ being arbitrarily small [14].

[3]In 1999, a similar error caused the loss of NASA's 125-million dollar Mars Climate Orbiter due to the use of both English and metric units of acceleration. This put us in good company, though happy that our error was less costly.

[4]This late night struggle explains the unusual values we published for the coefficients of friction of the pole on the cart ($\mu_c = 0.0005$) and the cart on the track ($\mu_t = 0.000002$). Of course, we wanted to experiment more, but it worked with these values, and we had to stop. Our haste also explains an unfortunate error that appeared in this article as published: the sign of gravity given in the Appendix has the wrong sign (although it was correct in our simulations). We learned of this from readers' attempts to duplicate our results. With the published sign of gravity, they found that no learning at all was needed to balance the down-hanging pole. This error did not hinder further research because it was quickly noted and became widely known among RL researchers.

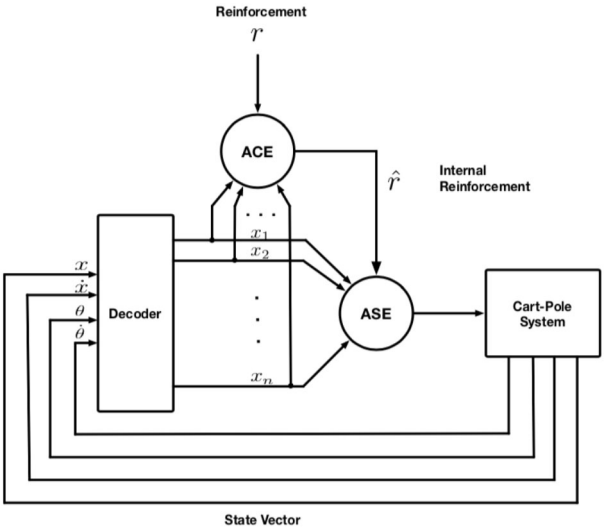


Fig. 1. Actor–critic architecture configured for the pole-balancing task as depicted in our 1983 paper [1].

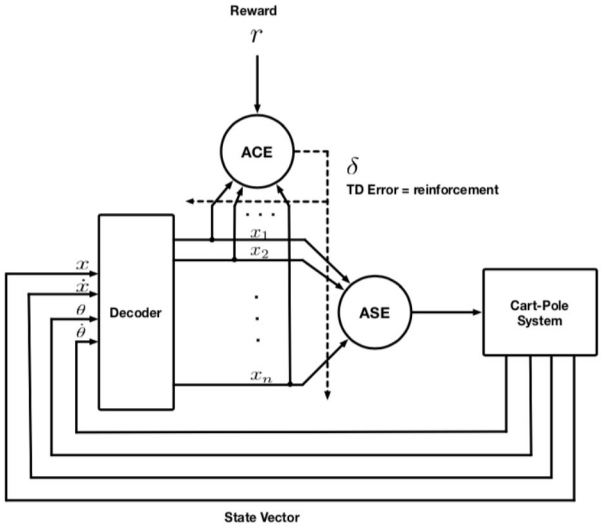


Fig. 2. Updated representation of the actor–critic architecture configured for the pole-balancing task. The input r is now labeled reward, and the dashed lines depict the TD error as the reinforcement signal for adjusting the input connection weights of both the ASE and the ACE.

In 1985, we had the opportunity to work with AI pioneer Oliver Selfridge on some additional experiments with the actor–critic architecture applied to the pole-balancing task [30]. It is more difficult to balance a short pole than it is to balance a longer pole. This suggested the following strategy for learning to balance a short pole: start with an easier-to-balance long pole and shrink its length as learning proceeds, ending up with the skill of balancing the short pole. We tried this using the actor–critic architecture of our 1983 paper. Learning to balance a 1-m pole took on average 67 failures, whereas learning to balance a 2/3-m pole took on average 119 failures. Switching to the short pole after learning to balance the long pole took just six more failures on average. This illustrated how an RL system could benefit from what psychologists call shaping [31], the training strategy that starts with an easy problem and incrementally increases its difficulty as the animal learns. Shaping is indispensable for animal training, and it can benefit learning via RL algorithms as well.

V. Some Details of the Architecture

Fig. 1 follows the figure from our 1983 paper showing the actor–critic architecture configured for the pole-balancing task. The output of the decoder is the standard unit basis vector representation of the 4-D cart-pole state space divided into $n = 225$ boxes. The ACE receives the reward signal r as input, in this case, the failure signal, and produces an "internal reinforcement" signal $\hat{r}$ which it sends to the ASE.

Fig. 2 is an updated representation of the actor–critic architecture based on our better understanding of how the architecture is related to psychology and neuroscience (see Section IX below). We distinguish between a reward signal, which sets the overall objective of the task, and a reinforcement signal, which directs the changes in the learner's parameters, here being the input connection weights of the ASE and ACE. As in Fig. 1, the ACE receives the signal r as input, but Fig. 2 labels it reward instead of reinforcement. The signal the ACE sends to the ASE is the reinforcement signal, which here is the TD error $\delta(t) = r(t)+\gamma p(t)-p(t-1)$, where p is a prediction of the eventual reward and γ is a discount factor between 0 and 1. The TD error is also the reinforcement signal for the ACE. This is shown by the dashed lines that cross the context input lines to each element.

A unique feature of the architecture is that although the TD error $\delta(t)$ is the reinforcement signal for both the ASE and the ACE, these elements learn to perform different functions: the ASE adjusts its parameters (its synaptic weights) in order to move its action probabilities toward higher-rewarding actions, while the ACE adjusts its parameters in order to make more accurate reward predictions.[5] The TD error tells the ASE if the current prediction $p(t)$ of the amount of reward expected over the future has just increased ($\delta > 0$) or just decreased ($\delta < 0$) so that the ASE's action probabilities can be adjusted to make it more, or less, likely to execute the same action when the current context occurs again. On the other hand, the TD error tells the ACE if its prediction of future reward is too low ($\delta > 0$) or too high ($\delta < 0$) so that it can correct its prediction.

The same reinforcement signal produces these different functions because the learning rules of the ASE and ACE differ in a subtle way: they use different notions of eligibility. Eligibility for the weight of an input connection to the ACE depends solely on input via that connection. That is, eligibility traces associated with the ACE's input connections are traces of past input via those connections. In contrast, eligibility for a weight associated with an input connection to the ASE depends, in addition to input via that connection, on the ASE's

[5]In some presentations (e.g., [2]), the TD error at step t is $\delta(t) = \mathrm{r(t+1)} + \gamma p(t+1) - p(t)$, that is, it depends on the reward and prediction at the next time step $t+1$. The interpretation of this form is that it is the error in the prediction made at step t that becomes available as a signal at $t+1$. The interpretation of the alternate form used in our 1983 paper is that $\delta(t)$ is the error in the prediction made at $t-1$ that becomes available as a signal at time t. Both interpretations appear in the literature.

output. That is, eligibility traces for an ASE's connections are traces of past input via those connections but modulated by the element's output. According to Klopf's notion of eligibility for a neuron, a synapse that transmits an excitatory pulse to the neuron would become eligible only if that input participated in causing the neuron to fire an action potential, and if a synapse transmits an inhibitory pulse, it would become eligible only if the neuron was inhibited from firing an action potential.

We call the ACE's eligibility *noncontingent* because it is not contingent on the element's output, and we call the ASE's eligibility *contingent* because it does depend on the element's output. Output contingency makes the ASE learn about the effect of its actions on the reinforcement signal. Lacking output contingency, the ACE learns to predict reward independently of its output.

The situation is somewhat more complicated than what we have written here. For example, in [1], the ASE's output was always nonzero, being either $+1$ or -1, so that eligibility traces accumulated positive and negative contributions of past activity. Details can be found in [2]. In Section IX, we discuss these issues from the perspective of neuroscience.

VI. Decomposition into Subgames

One of the objectives of Michie and Chambers' BOXES system was to illustrate the benefit of decomposing a large problem into many small problems, arguing (with reference to earlier work on playing Naughts and Crosses, i.e., Tic-tac-toe [32], [33]) that "it may be easier to learn to play many easy games than one difficult one." The organization of BOXES illustrated this by its division of the pole-track state space into 225 boxes, in each of which a local demon was faced with the relatively simple problem of learning how to act just for states falling in that box. BOXES illustrated that success on the large problem could be achieved in this way even if the subproblems, i.e., the problems faced by the local demons, were not independent.

The influence of this aspect of Michie and Chambers' paper is apparent in the direction we took in further developing RL. We maintained the view that favored learning in a state-dependent manner instead of treating the problem of finding an optimal policy as a monolithic, or "black box," optimization problem. In [2], Sutton and Barto referred to monolithic optimization methods as "evolutionary methods," and instead focussed on learning while interacting with an environment in order to take advantage of individual behavioral interactions, which monolithic optimization algorithms do not do.

It is fair to regard this emphasis on learning functions of states, such as value functions and policies, during interaction with the environment partly as a legacy of the influence on us of the BOXES paper. Also contributing to this emphasis was that learning via interaction fits better with our sense of how animals learn during their lifetimes. Evolutionary methods clearly have their place in machine learning where they can be the preferred methods for some problems, but to us there is a clear difference between interaction-based RL and evolutionary, or black-box, methods. Distinguishing the complementary capabilities of each may be a prerequisite to designing algorithms that combine them as effectively as they are combined in the natural world.

VII. Reinforcement Learning With Multilayer Neural Networks

We adopted the BOXES state representation for our pole-balancing experiments so that our system would be as similar to BOXES as possible, differing only in the most critical features. All the while, we were well aware of the much wider set of possible state representations that neural networks could provide, including the very rich representational abilities of multilayer neural networks. In parallel with our work on developing TD algorithms, modeling classical conditioning, and illustrating RL with the pole-balancing example, we were exploring RL methods for training the hidden layers of multilayer neural networks so that state representations could be learned rather than provided from the start.

We published a series of papers showing that multilayer neural networks could learn desired nonlinear mappings if each unit in the network learned via RL and the reward signal was broadcast uniformly to each unit [34]–[37]. In this approach, inspired by the earlier work with "teams" of stochastic learning automate, e.g., [15] and [38], the gradient of the objective function was stochastically estimated rather than backpropagated as in the error backpropagation algorithm that shortly became well known through the 1986 chapter by Rumelhart *et al.* [39]. Although our RL method was much slower than the backpropagation algorithm, we argued that it was simpler and more plausible biologically [40].

Co-author Anderson experimented with the RL approach for training multilayer networks, finding that adding a trainable hidden layer improved performance in the pole-balancing task [41]. He examined the features that the hidden layer learned, noting that they captured essential aspects of the control problem. Later, in his 1986 Ph.D. dissertation [42] (also [43]), Anderson compared a number of algorithms for hidden unit learning as applied to several tasks, including pole balancing and the Tower of Hanoi task. He found that both the RL method and error backpropagation learn the solutions to the tasks much more successfully than earlier methods for training multilayer networks, and he analyzed the features developed by the hidden units in solving these tasks.

Fig. 3 shows the two-layer actor and critic networks Anderson applied to the pole-balancing task [42]. Trained by backpropagation, the two-layer system far outperformed a single-layer system, although both learned more slowly than the system with the hand-crafted boxes state representation used in our 1983 paper. Anderson attributed this to the considerable number of steps that were required for the hidden units to learn the necessary features. Anderson's work is the first instance of which we are aware in which learning algorithms for multilayer neural networks were used in RL tasks, foreshadowing current advances in deep RL.

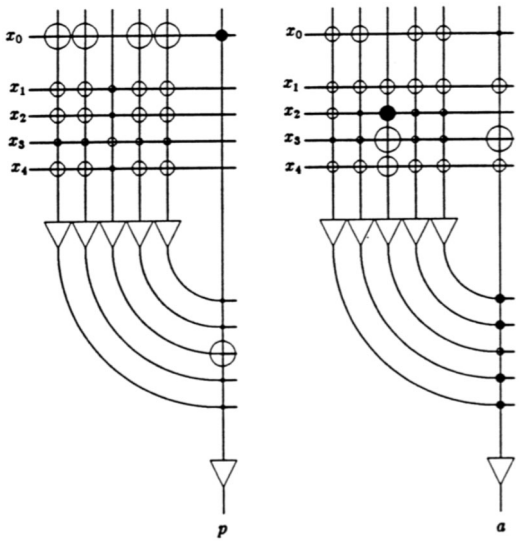


Fig. 3. Two-layer neural network actor–critic architecture applied to the pole-balancing task from Anderson's 1986 Ph.D. dissertation [42]. The left and right networks, respectively, implemented the critic and the actor. In each network, the five input lines connect to five hidden units and to the single output unit, for a total of 35 weights. The relative magnitudes and signs of the learned weights are shown by the circles on the line intersections, with the open and black circles, respectively, indicating positive and negative weights.

VIII. Theory

RL has seen tremendous progress since our 1983 paper. New algorithms, many applications, and vastly improved theoretical understanding emerged as a result of the contributions of numerous researchers. RL has benefitted from being connected to more traditional fields, in particular, to stochastic optimal control and dynamic programming. RL is now regarded as a collection of methods for approximating solutions to Markov decision processes (MDPs), a framework into which all of our earlier work can be placed. Here, we mention a few highlights of this theoretical development that bear directly on the actor–critic architecture, omitting details that can be found in many publications, e.g., [2].

The actor–critic architecture can be understood most clearly as a policy-gradient algorithm. A policy is a function that maps environment states to control actions, often probabilistically. In a policy-gradient algorithm, the policy is represented as a parameterized function with parameters adjusted by the learning algorithm. In an actor–critic algorithm, the actor updates policy parameters by moving the policy parameter vector in the direction of an estimated gradient of a measure of long-term reward, where the gradient is estimated from sample trajectories. The ASE learning rule in our 1983 paper does not exactly perform gradient ascent, but later versions incorporating William's REINFORCE algorithm [44] do achieve stochastic gradient ascent.

The critic of the actor–critic architecture updates parameters of a parameterized state-value function, which is a function that assigns to each state an estimate of the expected long-term return (the expected cumulative reward) when a given policy is followed from that state. The critic uses a TD algorithm to adjust the value-function parameters in order to improve its prediction accuracy based on observed state transitions and rewards.

The interaction of the actor and critic is analogous to the policy-iteration algorithm of dynamic programming. Each iteration of that algorithm alternates between computing a state-value function for a current policy, and then improving the current policy according to the current state-value function. Actor–critic methods effectively perform these two phases simultaneously, interleaving single steps of state-value function estimating with single steps of policy improvement.

The theoretical properties of traditional policy iteration are well known (it converges to an optimal policy for finite MDPs under mild conditions), but the actor–critic analog is more difficult to analyze. The most comprehensive convergence results are due to Bhatnagar *et al.* [45], who prove convergence to a local maximum of the long-run average reward for several versions of the actor–critic algorithm using a two-timescale approach in which the critic learns faster than the actor.

The 1990s saw the development of RL algorithms based on estimating action-value functions instead of state-value functions, the prime example being Q-learning [46]. Action-value functions map state–action pairs to expected return. With these methods, there is no need for an explicit policy representation because actions can be selected simply by consulting the estimated values of the actions for the current state. Action-value functions have been called "action-dependent adaptive critics" [47].

Action-value algorithms came to be preferred over actor–critic algorithms because of their relative ease of implementation, but understanding their convergence properties is challenging, especially when they use function-approximation methods for learning action-value functions. As a result, interest in policy-gradient algorithms, including actor–critic algorithms, has lately increased. Their reliance on explicit parameterized policy representations offers a number of advantages, among which are the following: 1) they provide useful ways to deal with continuous action spaces; 2) they make it possible to select actions with arbitrary probabilities, and yet can converge to deterministic policies; and 3) they offer a good way to introduce prior knowledge into the learning process. Additionally, sometimes a high-performing policy is much simpler than an action-value function [48]. By accounting for this simplicity in selecting a policy parameterization, learning can be faster and lead to a better policy than possible with action-value methods.

IX. Actor–Critic in the Brain

Mounting evidence from neuroscience suggests that the nervous systems of humans and many other animals implement algorithms that correspond in striking ways to RL algorithms. The most remarkable point of contact involves dopamine, a chemical fundamentally involved in reward processing in the brains of mammals. Experiments have shown that neurons (at least many of them) that produce dopamine as a neurotransmitter respond to rewarding events with substantial bursts of activity only if the animal does not expect those events [49]. This finding suggests that many dopamine-producing

neurons are signaling reward prediction errors instead of reward itself.

Experiments have also shown that as an animal learns to predict a rewarding event on the basis of preceding sensory cues, the bursting activity of dopamine-producing neurons shifts to earlier predictive cues while decreasing to later predictive cues. This parallels the backing-up effect of the TD error as the ACE learns to predict reward. It is now widely accepted that bursts of dopamine neuron activity convey reward prediction errors to brain structures where learning and decision making take place, and evidence supports the idea that the prediction errors might be TD errors [50].

Experimental results like these have led to the hypothesis that the brain might implement something like our actor–critic architecture. TD errors conveyed by the activity of dopamine neurons are reinforcement signals that train both the critic's predictions and encourage or discourage the actor's choice of actions [49], [51].

Whether a brain region performs an actor-like or a critic-like function depends on how synaptic efficacies change in each region in response to receipt of dopamine reinforcement. Synapses in a critic-like region would implement noncontingent eligibility traces. In neural terms, noncontingent eligibility means that the eligibility of a synapse is solely a function of presynaptic activity, that is, of activity that reaches the synapse as input from other neurons. Synapses in an actor-like region, on the other hand, would implement contingent eligibility, which is a function of the activity both the pre- and postsynaptic neurons. Neuroscientists would say that critic synapses have a two-factor learning rule (presynaptic activity + dopamine), whereas actor synapses have a three-factor learning rule (presynaptic activity + postsynaptic activity + dopamine). Moreover, if the neuron is to implement a kind of law of effect as Klopf conjectured, the presynaptic activity must have taken part in generating the postsynaptic activity in order for the synapse to become eligible for modification.

Even if neurons behave nothing like the ASE or the ACE of the actor–critic architecture, our 1983 paper suggested that neurons may be capable of very sophisticated processing, vastly more complex than the logic-gate analogy of old would suggest. That nearly the entire BOXES system of Michie and Chambers could be implemented by a *single* neuron-like element is a touchstone for thinking about neural networks as being more analogous to networks of computers than to logic circuits.

Furthermore, understanding how adaptive elements such as the ASE operate requires thinking about them as embedded in closed-loop interactions with their environments. They are metaphorically "swimming" in a medium composed of the rest of the neural net plus the organism's (or robot's) external environment. Their adaptive changes are sensitive to the effects that their actions have on input signals they later receive.

The brain's reward system is undoubtedly much more complicated than current RL algorithms, and the story is still unfolding as more is being learned about the brains's reward system, but the actor–critic architecture, along with other RL algorithms and theory, is proving to be enormously useful in making sense of experimental data, in suggesting new kinds of experiments, and in pointing to factors that may be critical to manipulate and to measure. See [2] for more about RL and neuroscience.

X. Future

Some of the most impressive achievements in AI have been produced by programs that include RL. Notable examples are DeepMind's Go-playing programs [52], [53]. While these programs are vastly more complex than an actor–critic architecture, and the problem they faced is vastly more difficult than pole balancing, they nevertheless carry forward some features of our RL pole balancer, such as learning from interaction with a dynamic environment, caching trial-and-error search results, and learning value predictions to address long-term goals. The future will see the methods used in these game-playing programs, and other successful learning programs, adapted and extended to address a widening range of challenging problems, including pressing real-world problems of scientific and social importance.

RL has the potential to improve the quality, efficiency, and cost effectiveness of processes on which we depend in education, healthcare, transportation, and energy management, among others, but challenges have to be addressed to realize this potential. Many design decisions are involved in applying RL. The architecture has to be designed by selecting appropriate learning algorithms, state and action representations, training procedures, hyperparameter settings, and other design details. An important goal for future research is to make RL algorithms more robust and easier to apply so that new applications can be developed by experts in the application domains instead of by teams of experts in RL and other machine learning methods.

There are ample opportunities to improve and generalize RL algorithms and architectures. Examples include: architectures for learning hierarchical polices to improve efficiency and the ability to transfer learning to new problems; efficient methods for learning with incomplete state information; expanding the role of prediction to enable agents to predict and control many signals from their environments, not just long-term reward; further development of multiagent RL; further development of model-based RL to integrate planning and higher level reasoning; and architectures for open-ended life-long learning. Progress has been made along these and other directions, but much more is possible. Additional improvements are discussed in [2] and [54].

As RL moves out into the real world, it is critical to make sure that what is learned conforms to the intentions of the application's designer, and that the learning agent does no harm to itself or to its environment, including any people in it, both during and after learning. This requires adopting risk mitigation and management strategies that exist for other risk-prone technologies and designing new methods especially targeting risks posed by machines that learn through interacting with real-world environments. Unless RL is restricted to always operate in benign environments, like game playing where one can tolerate the worst that can happen,

ensuring the safety of RL applications is a critical challenge that needs careful attention.

The design of an RL system's reward function is of special importance for RL safety. This is the function that assigns reward and penalty magnitudes to states, actions, state–action pairs, and perhaps other aspects of the system. Because RL fundamentally involves optimization, with the reward function being the objective function, it shares with other optimization processes the problem that it can produce unexpected, and sometimes unwanted, possibly catastrophic, results. This possibility has long been recognized. For example, Norbert Wiener, the founder of cybernetics, warned of this problem more than half a century ago by relating the supernatural story of "The Monkey's Paw:" "$\cdots$ it grants what you ask for, not what you should have asked for or what you intend" [55, p. 59]. The problem is featured as "perverse instantiation" in Bostrom's broadside about the dangers of AI [56]. Methods for designing reward functions are needed that go beyond the hand tuning that is common practice today, and sound methodologies are needed to assess the safety of what a reward function enables a system to learn.

Turning to the connections between RL and neuroscience, the future will see continued fruitful interaction between neuroscience and RL. Advances in RL will suggest new ways to think about the brain's decision and reward systems, and advances in neuroscience will inform the further development of RL. As neuroscience uncovers more about how reward processing works in the brain, we may see experimental support for Klopf's hypothesis that neurons—at least some of them—individually implement a kind of law of effect.

XI. Conclusion

Contributing to this 50th Anniversary Issue of IEEE Transactions on Systems, Man, and Cybernetics has given us the opportunity to revisit some of our earliest efforts in what has become the flourishing RL subarea of machine learning. The 1983 "Neural and Sensory Information Processing" special issue of the journal provided the ideal venue for our work at a time when biologically inspired machine learning was not as routine as it is today. We have been gratified and surprised by the influence our paper has had. It is among the most frequently cited of the publications by any of us, its three authors, and pole balancing has served as a testbed for many different learning archiectures.

Exploring Klopf's hedonistic neuron hypothesis led us through some of the early history of AI, to psychology's theories of learning, and eventually to appreciation of stochastic optimal control and dynamic programming. The striking parallels between TD algorithms and the brain's dopamine system revealed strong connections between RL algorithms and reward processing in the brain. Although Klopf's idea of the hedonistic neuron remains a hypothesis, time will tell if it finds compelling neuroscientific support.

There is now a large international community of researchers improving RL algorithms and architectures, combining RL with other technologies, creating new RL applications, and exploring new directions inspired by RL. The actor–critic architecture now is in the company of many other architectures that embody core RL ideas while exploiting the immensely improved computational resources available today. We fully expect that RL can help improve the quality, fairness, and sustainability of life on our planet, provided its risks can be successfully managed.

References

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 835–846, Oct. 1983.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[3] A. H. Klopf, "Brain function and adaptive systems—A heterostatic theory," Air Force Cambridge Res. Lab., Bedford, MA, USA, Rep. AFCRL-72-0164, 1972.

[4] A. H. Klopf, *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Washington, DC, USA: Hemisphere, 1982.

[5] W. R. Ashby, *Design for a Brain: The Origin of Adaptive Behavior*. London, U.K.: Assoc. Book Publ., 1960.

[6] A. G. Barto and R. S. Sutton, "Goal seeking components for adaptive intelligence: An initial assessment," Air Force Wright Aeronaut. Lab./Avionics Lab., Wright-Patterson AFB, OH, USA, Rep. AFWAL-TR-81-1070, 1981.

[7] E. L. Thorndike, *Animal Intelligence*. Darien, CT, USA: Hafner, 1911.

[8] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[9] S.-I. Amari, "Neural theory of association and concept-formation," *Biol. Cybern.*, vol. 26, no. 3, pp. 175–185, 1977.

[10] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model," *Psychol. Rev.*, vol. 84, pp. 413–451, 1977.

[11] T. Kohonen, *Associative Memory: A System Theoretic Approach*. Berlin, Germany: Springer-Verlag, 1977.

[12] A. G. Barto, R. S. Sutton, and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biol. Cybern.*, vol. 40, pp. 201–211, May 1981.

[13] M. L. Tsetlin, *Automaton Theory and Modeling of Biological Systems*. New York, NY, USA: Academic, 1973.

[14] K. S. Narendra and M. A. L. Thathachar, "Learning automata—A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, no. 4, pp. 323–334, Jul. 1974.

[15] K. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[16] E. Harth and E. Tzanakou, "Alopex: A stochastic method for determining visual receptive fields," *Vis. Res.*, vol. 14, pp. 1475–1482, Dec. 1974.

[17] E. Tzanakou, R. Michalak, and E. Harth, "The alopex process: Visual receptive fields by response feedback," *Biol. Cybern.*, vol. 35, no. 3, pp. 161–174, 1979.

[18] D. A. Berry and B. Fristedt, *Bandit Problems*. London, U.K.: Chapman-Hall, 1985.

[19] D. Michie, *Memo Functions: A Language Feature With "Rote-Learning" Properties* (Research Memorandum MIP-R-29), Dept. Mach. Intell. Percept., Univ. Edinburgh, Edinburgh, U.K., 1967.

[20] R. J. Popplestone, *"Memo" Functions and the Pop-2 Language* (Research Memorandum MIP-R-30), Dept. Mach. Intell. Percept., Univ. Edinburgh, Edinburgh, U.K., 1967.

[21] B. Widrow, N. K. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 5, pp. 455–465, Sep. 1973.

[22] R. S. Sutton and A. G. Barto, "Toward a modern theory of adaptive networks: Expectation and prediction," *Psychol. Rev.*, vol. 88, pp. 135–170, Mar. 1981.

[23] A. G. Barto and R. S. Sutton, "Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element," *Behav. Brain Res.*, vol. 4, pp. 221–235, Mar. 1982.

[24] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, Dept. Comput. Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, 1984.

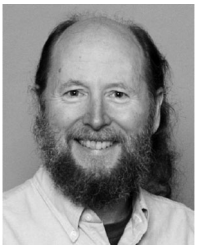
[25] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, Aug. 1988.

[26] D. Michie and R. A. Chambers, "BOXES: An experiment in adaptive control," in *Machine Intelligence 2*, E. Dale and D. Michie, Eds. Edinburgh, U.K.: Oliver Boyd, 1968, pp. 137–152.
[27] B. Widrow and F. W. Smith, "Pattern-recognizing control systems," in *Computer and Information Sciences*, J. T. Tou and R. H. Wilcox, Eds. Washington, DC, USA: Spartan, 1964, pp. 288–317.
[28] O. J. Selfridge, "Pandemonium: A paradigm for learning," in *Proc. Symp. Mech. Thought Process.*, 1959, pp. 511–529.
[29] J. C. Gittins and D. M. Jones, "A dynamic allocation index for the discounted multiarmed bandit problem," *Biometrika*, vol. 66, no. 3, pp. 561–565, 1979.
[30] O. J. Selfridge, R. S. Sutton, and A. G. Barto, "Training and tracking in robotics," in *Proc. 9th Int. Joint Conf. Artif. Intell.*, 1985, pp. 670–672.
[31] B. F. Skinner, "Reinforcement today," *Amer. Psychol.*, vol. 13, no. 3, p. 94, 1958.
[32] D. Michie, "Trial and error," in *Science Survey, Part 2*, S. A. Barnett and A. McLaren, Eds. Harmondsworth, U.K.: Penguin, 1961, pp. 129–145.
[33] D. Michie, "Experiments on the mechanization of game-learning part I. Characterization of the model and its parameters," *Comput. J.*, vol. 6, no. 3, pp. 232–263, 1963.
[34] A. G. Barto, "Game-theoretic cooperativity in networks of self-interested units," in *Neural Networks for Computing*, J. S. Denker, Ed. New York, NY, USA: Amer. Inst. Phys., 1986, pp. 41–46.
[35] C. W. Anderson, "Strategy learning with multilayer connectionist representations," in *Proc. 4th Int. Workshop Mach. Learn.*, 1987, pp. 103–114.
[36] A. G. Barto, "Learning by statistical cooperation of self-interested neuron-like computing elements," *Hum. Neurobiol.*, vol. 4, no. 4, pp. 229–256, 1985.
[37] A. G. Barto, P. Anandan, and C. W. Anderson, "Cooperativity in networks of pattern recognizing stochastic learning automata," in *Adaptive and Learning Systems: Theory and Applications*, K. S. Narendra, Ed. New York, NY, USA: Plenum, 1986.
[38] K. S. Narendra and R. M. Wheeler, "An *n*-player sequential stochastic game with identical payoffs," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 6, pp. 1154–1158, Nov./Dec. 1983.
[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1: Foundations*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA, USA: MIT Press, 1986.
[40] A. G. Barto and M. I. Jordan, "Gradient following without back-propagation in layered networks," in *Proc. IEEE 1st Annu. Conf. Neural Netw.*, San Diego, CA, USA, 1987, pp. II629–II636.
[41] C. W. Anderson, "Feature generation and selection by a layered network of reinforcement learning elements: Some initial experiments," Dept. Comput, Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, Rep. COINS 82–12, 1982.
[42] C. W. Anderson, "Learning and problem solving with multilayer connectionist systems," Ph.D. dissertation, Dept. Comput. Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, 1986.
[43] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Control Syst. Mag.*, vol. 9, no. 3, pp. 31–37, Apr. 1989.
[44] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, May 1992.
[45] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.
[46] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.
[47] D. Liu, X. Xiong, and Y. Zhang, "Action-dependent adaptive critic designs," in *Proc. Int. Joint Conf. Neural Netw. (Cat. No. 01CH37222)*, vol. 2. Washington, DC, USA, 2001, pp. 990–995.
[48] C. W. Anderson, "Approximating a policy can be easier than approximating a value function," Dept. Comput. Sci., Colorado State Univ., Fort Collins, CO, USA, Rep. CS-00-101, 2000.
[49] W. Schultz, "Predictive reward signal of dopamine neurons," *J. Neurophysiol.*, vol. 80, pp. 1–27, Jul. 1998.
[50] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, pp. 1593–1598, Mar. 1997.
[51] A. G. Barto, "Adaptive critics and the basal ganglia," in *Models of Information Processing in the Basal Ganglia*, J. C. Houk, J. L. Davis, and D. G. Beiser, Eds. Cambridge, MA, USA: MIT Press, 1995, pp. 215–232.
[52] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[53] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[54] M. Wiering and M. V. Otterlo, *Reinforcement Learning: State-of-the-Art*. Berlin, Germany: Springer-Verlag, 2012.
[55] N. Wiener, *God & Golem, Inc.* Cambridge, MA, USA: MIT Press, 1964.
[56] N. Bostrom, *Superintelligence: Paths, Dangers, Strategies*. Oxford, U.K.: Oxford Univ. Press, 2014.

Andrew G. Barto (Life Fellow, IEEE) received the B.S. (with distinction) degree in mathematics and the Ph.D. degree in computer science from the University of Michigan, Ann Arbor, MI, USA, in 1970 and in 1975, respectively.

He Co-Directed the Autonomous Learning Laboratory, University of Massachusetts Amherst, Amherst, MA, USA, where he is a Professor Emeritus of Computer Science having retired in 2012. He is currently an Associate Member of the Neuroscience and Behavior Program, University of Massachusetts Amherst. He has published over one hundred papers or chapters in journals, books, and conference and workshop proceedings. He has coauthored textbook *Reinforcement Learning: An Introduction*, (MIT Press, 1998, 2018).

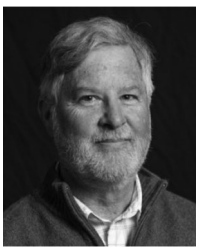
Dr. Barto received the 2004 IEEE Neural Network Society Pioneer Award, the IJCAI-17 Award for Research Excellence, and the University of Massachusetts Neurosciences Lifetime Achievement Award in 2019. He serves as an Associate Editor of Neural Computation, as a Member of the Advisory Board of the *Journal of Machine Learning Research*, and as a Member of the Editorial Board of *Adaptive Behavior*. He is a Fellow of the American Association for the Advancement of Science.



Richard S. Sutton received the B.A. degree in psychology from Stanford University, Stanford, CA, USA, in 1978, and the Ph.D. degree in computer science from the University of Massachusetts Amherst, Amherst, MA, USA, in 1984.

He is a Distinguished Research Scientist with DeepMind, Edmonton, AB, Canada, and a Professor with the Department of Computing Science, University of Alberta, Edmonton. Prior to joining DeepMind in 2017, and the University of Alberta in 2003, he worked in industry at AT&T and GTE Labs, and in academia at the University of Massachusetts Amherst. He has coauthored textbook *Reinforcement Learning: An Introduction* from (MIT Press). His research interests center on the learning problems facing a decision-maker interacting with its environment, which he sees as central to intelligence. He has additional interests in animal learning psychology, in connectionist networks, and generally in systems that continually improve their representations and models of the world. His scientific publications have been cited more than 70 000 times. He is also a libertarian, a chess player, and a cancer survivor.

Prof. Sutton is also a Fellow of the Royal Society of Canada, the Association for the Advancement of Artificial Intelligence, the Alberta Machine Intelligence Institute, and CIFAR.



Charles W. Anderson (Senior Member, IEEE) received the B.S. degree in computer science from the University of Nebraska-Lincoln, Lincoln, NE, USA, in 1978, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, MA, USA, in 1986.

From 1986 to 1990, he was a Senior Member of Technical Staff, GTE Laboratories, Waltham, MA, USA. He is currently a Professor with the Department of Computer Science, Colorado State University, Fort Collins, CO, USA, with joint appointments with the School of Biomedical Engineering, the Molecular, Cellular and Integrative Neuroscience Program, the Graduate Degree Program in Ecology, and the Systems Engineering Program. He is the founder of Pattern Exploration, a small company that develops explainable AI solutions to prediction, classification, and control problems. His research interests are in practical algorithms for training neural networks and applying them to real-world problems in climate, environment, health, and energy.