Constructing Temporal Abstractions Autonomously in Reinforcement Learning

Pierre-Luc Bacon, Doina Precup

■ *The idea of temporal abstraction, that is, learning, planning, and representing the world at multiple time scales, has been a constant thread in AI research, spanning subfields from classical planning and search, to control and reinforcement learning. While temporal abstraction is a very natural concept, learning these abstractions without human input has proved quite daunting. In this paper, we present a general architecture called option-critic for learning temporal abstractions end to end from the agent's experience. This approach allows for continual learning and provides interesting qualitative and quantitative results in several tasks.*

Intelligent systems have the ability to adapt and generalize quickly in the presence of change and uncertainty in their environments. Fundamentally, the success of their adaptation and learning strategies hinges on the quality of their representations. Simon (1969, 132), in fact, argued that "[s]olving a problem simply means representing it so as to make the solution transparent."

Building good representations is a challenge of long standing in artificial intelligence. In this article, we examine this problem in the context of reinforcement learning, the learning paradigm in which an agent interacts with its environment by making observations, choosing actions, and receiving feedback in the form of a numerical reward. The goal of the agent is to maximize an expected cumulative measure over the rewards. Since the environment might be enormous (as in the case of the game of Go, for example), and the reward may be sparse, a good representation needs to generalize well not only over observations (or perceptions) but additionally over multiple time scales.

 ISSN 0738-4602

Over time, notable progress has been made in the realm of perceptual generalization. For example, the celebrated TD-Gammon (Tesauro 1995) program achieved unprecedented performance against a human backgammon champion by using a combination of reinforcement learning techniques and a two-layer neural network. Recent advances in deep neural networks have led to even more impressive demonstrations of this ability in tasks such as wagering the daily double (Tesauro et al. 2013), playing Atari (Mnih et al. 2015), and playing Go (Silver et al. 2016). In all these cases, the system is given the responsibility of building its own representation by leveraging data.

The problem of building good generalizations of actions over multiple time scales, otherwise known as temporal abstraction, has also received a steady influx of attention across different subfields of artificial intelligence (Minsky 1961; Fikes, Hart, and Nilsson 1972; Kuipers 1979; Korf 1983; Iba 1989; Drescher 1991; Dayan and Hinton 1992; Kaelbling 1993; Dean and Lin 1995; Sutton, Precup, and Singh 1999). Even in the early stages of AI, Minsky (1961, 10) recognized how "[…] we rarely solve a tricky problem by a steady climb toward success," making a hierarchical approach to problem solving more likely to subtend our intellectual abilities. This organization of knowledge gives a system the ability to choose the right level of abstraction for a problem. As a consequence, progress made at one level may appear as a stroke of insight (Minsky 1961) from the level above.

Systems with insight — the capacity to gain an accurate and deep intuitive understanding of a problem — *(Oxford English Dictionary)* have the flexibility to reason and learn beyond the confines of the knowledge provided to them a priori. Such systems are *habile* (Nilsson 1995). In contrast, *performance systems* (Nilsson 1995) are designed for specific problems. While they may achieve superhuman performance, they lack general autonomy and competency.

We have been pursuing the goal of temporal abstraction for building habile systems, which means that a learner should not just represent its knowledge at given time scales, but also automatically figure out which time scales are interesting for both prediction and control. The problem of knowledge representation at multiple scales can be handled in reinforcement learning systems through the framework of *options* (Sutton, Precup, and Singh 1999). Generally speaking, options encapsulate behaviors that can be initiated and terminated, akin to subroutines in a programming language. Planning with given options, as well as learning options that achieve prespecified subgoals, is well understood. However, the problem of option discovery — figuring out a good set of options fully automatically — has proven very hard to handle so far. A possible reason for this impasse is the attention put on allowing program designers to specify what is interesting problem structure. The resulting programs are akin to Nilsson's performance systems: they do well on certain tasks, but are often too brittle to deploy widely or scale up.

In this article, we describe the option-critic architecture, which is our attempt to take the step towards more habile systems. The idea that a learning system should be in charge of finding the options that are suitable for itself, given its environment, is the core principle of this work. We wish to allow the agent to continually learn and adapt its representation at all abstraction levels, based solely on the data stream it observes, without requiring biasing information from a human designer. Our approach builds on the actor-critic architecture (Sutton 1984), which provides an incremental, online, and model-free approach to learning from a continual stream of experience. Unlike other previous or existing methods, the option-critic architecture requires no subgoals, pseudorewards, decomposition, or demonstrations, and it constructs options fully autonomously while embedded in a control task that has to be solved at the same time.

This article is an overview of the conceptual and technical aspects of our approach. We start with a brief review of reinforcement learning and a formalization of the concept of temporally extended actions. We then describe the options framework, but take a detour along the way to appreciate its roots in constructivism (Drescher 1991). This perspective provides the properties sought in our system, which we achieve using ideas that stem from actor-critic methods. Finally, we demonstrate an option-critic system built over a deep network that is capable of learning to play Atari games and of constructing interesting options at the same time.

Reinforcement Learning

Reinforcement learning (RL) refers to learning from the experience generated by an agent interacting with its environment. Conceptually, the field has been inspired by the work on trial-and-error learning from psychology, but its methods were formalized and analyzed using the theory of Markov decision processes (MDPs). An MDP consists of a set of states (modeling the agent's perceptions) and a set of actions. For each state-action pair, there is a well-defined transition probability distribution from which the next state will be drawn. The reward function specifies, for each state-action pair, an immediate numerical reward that will be received by the agent. While the transition and reward functions are assumed to exist, the agent does not have access to them. Instead, it interacts with the environment by observing states, choosing actions, and observing the resulting rewards and next states. A sequence of states, actions, and rewards generated in this manner is called a *trajectory*.

The agent will typically seek a way of choosing actions, conditioned on states, that is rewarding in the long run. Such a stochastic decision procedure is called a *policy* (denoted by π). Rather than simply maximizing the total reward, which may not be bounded in general, the agent usually attempts to maximize discounted returns. The discount factor γ can be conceptualized as an inflation rate that deprecates rewards at every time step.

In the policy evaluation problem, the goal is to compute the expected discounted return for a given, fixed policy over the distribution of possible trajectories. This information is summarized in a value function:

$$v_\pi(s) = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(S_t, A_t) | S_0 = s\right]$$

In the control problem, the goal is to find a policy that maximizes the expected return.

A natural approach to these problems for agents that are continually acting and learning is temporal difference (TD) learning, introduced by Sutton (1984, 1988) in the context of policy evaluation, and later adapted for control through the Q-learning (Watkins 1989) and Sarsa (Rummery and Niranjan 1994) algorithms. For the policy evaluation case, the core idea is that, after learning has completed, the value of a state should be equal, in expectation, to the reward plus the discounted value of the next state. The temporal difference error quantifies how different the estimated value of a state is at the current time step, compared to one time step later (when a new sample transition and reward have been observed). The algorithm uses this error to train an approximation of v_π. In the case of control, this idea is supplemented with a simple strategy for changing the policy π over time: actions that lead to better-than-expected outcomes should be taken more often (that is, reinforced).

Actions with Variable Duration

With the modern foundations of learning through reinforcement established by the end of the 1980s, a number of proposals were made to extend the scope of reinforcement learning methods from actions of fixed duration to actions temporally extended (Watkins 1989; Singh 1992; Dayan and Hinton 1992; Kaelbling 1993; Thrun and Schwartz 1995; Sutton 1995), culminating at the end of the '90s (Parr and Russell 1998; Hauskrecht et al. 1998; Dietterich 1998; Sutton, Precup, and Singh 1999) with several formulations based on semi-Markov decision processes (SMDP) (Howard 1963).

The MDP model makes the assumption that the environment transitions to a new state in a single time step (or, equivalently, in a constant amount of time), while in an SMDP, the transition duration is a random variable. The SMDP framework is therefore a natural fit for representing temporally abstract actions, which can persist over time. More precisely, in an SMDP, the choice of action at a given state induces a joint probability distribution over both the next state and the duration of the transition. Hence, a trajectory in an SMDP includes, in addition to states, actions, and rewards, the duration of each transition. The name *semi-Markov* stems from the fact that the process is only assumed to be Markovian from decision point to decision point, which conveniently allows for existing dynamic programming results to apply at the level of decisions (or action choices). However, the evolution of the system between two decisions may not even be Markovian, and it is also allowed to unfold over continuous time. In fact, when the transition duration is exponentially distributed, this leads to a decision process called *continuous-time Markov decision process* (CTMDP) (Puterman 1994).

Seeing Through the Black Box with Options

Despite adopting the same SMDP formalism, the options framework (Sutton, Precup, and Singh 1999) differs from its contemporaries (Parr and Russell 1998; Dietterich 1998) in its emphasis on exposing and leveraging the structure both within and over temporally extended actions. The evolution of the process between two decision points is no longer a black box, which means it can be both observed and controlled. The ability to seamlessly learn and plan at different levels of abstraction stems from the assumption that there exists a base MDP that is overlaid with temporally extended actions, known as options: the combination is shown to induce an SMDP. With the expression "between MDPs and semi-MDPs" in the title of their paper, Sutton, Precup, and Singh (1999) tried to convey the idea that options provide a lens of variable resolution.

An option is a combination of three components: an initiation set, a policy (sometimes called internal), and a termination condition. The initiation set I_o for an option o is a subset of the states in which the given option could be chosen. In the most common execution model, when an option is executed, its internal policy π_o acts until the probabilistic termination condition β_o is met. More precisely, upon entering the next state S_{t+1}, the system has to toss a coin, which indicates termination with probability $\beta_o(S_{t+1})$ and continuation with $1 - \beta_o(S_{t+1})$. Once an option has terminated, a policy over options μ chooses a new option, and the process is repeated.

Constructivist Influence

To get some perspective on what the options framework is and what it ought to be, it is useful to follow its lineage into the schema mechanism of Drescher (1991). Inspired by Piaget's constructivism (Piaget 1937), Drescher puts forward the idea that all knowl-

edge acquired by an agent is represented in terms of its own experience with the sensation of its actions in the environment. A schema, whose semantics have much in common with options, is a symbolic structure that describes the result of an action given a context. The role of a schema goes beyond the simple specification of actions and can be used to express general knowledge about the environment. For example, a robot might choose to represent the fact that a charger is in front of it based on its own prediction of what would happen if it were to dock (Sutton 2012). It might also choose to represent the presence of humans based on the predicted sensory inputs that would typically follow the playback of its bebeep boop sounds and dance sequence in the morning.

This idea of building representations of the world grounded in predictions about the outcome of temporally abstract actions has informed the development of the options framework. Building on an earlier line of work (Sutton 1995; Precup and Sutton 1997; Precup, Sutton, and Singh 1998), Sutton, Precup, and Singh (1999) showed that predictions about the expected return, the future state, and the duration of an option could be used for planning. Like simple actions, options have associated reward and transition models, which can be used in a set of Bellman equations at the SMDP level and whose solution can be found by dynamic programming methods. Options and their models, then, play a representational role in the sense of Drescher (1991): they are agent-centric and encode meaning over action-conditional predictions.

A Multifaceted Framework

Looking beyond the purely constructivist perspective, thinking about options in isolation from their models has also been of practical interest. Rather than choosing actions after reasoning in a predictive representation, options can interact directly with the environment. This direct interaction leads to what we call the *execution perspective* on options. Here, an option is more procedural in nature and acts as a data structure for expressing action choices. Using computer program execution as an analogy, an option is akin to a function executed within a program in a call-and-return fashion: its instructions are read (policy of an option), they are moved to the CPU (environment), and when the option terminates, the next function is loaded (initiated) from the call stack along with its arguments. Complex control flows can be generated in this manner and a generalization to deeper hierarchies follows naturally.

Fundamentally, it is the need for remembering which option is currently executing that leads to the concept of a stack. The content of the stack is also where we draw a line between Markov options and semi-Markov options. In the simplest case, the stack for Markov options is of constant size because it holds exclusively the identity of the current option: a single integer variable is sufficient for implementation. For example, we can imagine a robot navigation task for which a good Markov option might be: "if there is no obstacle" (initiation set), "move forward" (the policy of that option) "until the charger is reached" (termination condition). However, if we were to also specify that the robot should stop searching for the charger after some time, the corresponding option would be semi-Markovian. In fact, the need to actively keep track of time creates a dependence on the history since initiation (unless timing information is included in the state space). An option is therefore Markovian if its behavior depends only on the current state and not on any measurements of the history since its initiation. The restriction to Markov options leads to the powerful idea of intra-option learning (Sutton, Precup, and Singh 1998), which has no analogue in the semi-Markov case. Both the Markov option property and the intra-option formulation are central to our approach for learning options.

The Bottleneck Concept

Learning and planning with options has been well understood since Sutton, Precup, and Singh (1999). If the options are prespecified, then dynamic programming or temporal difference learning methods can be used to learn about option values and models. However, the problem of discovering useful options automatically is still difficult to tackle. The challenge comes on two fronts: defining what useful or good options mean, and designing algorithms for finding those options.

An important contribution to the discovery problem in the context of classical macroactions came from Iba (1989) and his peak-to-peak heuristic, inspired by the concept of chunking (Mayzner and Gabriel 1963) from psychology. The premise of this work is that pairs of peaks in the evaluation function should provide useful demarcations for where temporally extended actions should start and end. Based on the same intuition, Konidaris and Barto (2009) and later Niekum et al. (2012) framed option discovery as a change-point detection problem from expert demonstrations.

This idea of peaks is also related to bottleneck states (McGovern and Barto 2001; Stolle and Precup 2002), states that occur more frequently on successful trajectories through the environment. Bottleneck states, like peaks, are intuitively associated with breakthroughs in the solution. Consider a goal-directed navigation task in an environment containing rooms and doorways. If the agent is starting in a separate room from the goal, doorways would necessarily be crossed on successful trajectories and should therefore be useful subgoals.

Many graph-theoretic formulations of the bottle-

neck concept have been proposed over the years. For example, Simsek and Barto (2008) chose the notion of betweenness centrality (Freeman 1977), which bases a measure of importance for a vertex on the relative number of shortest paths passing through it. Alternatively, graph-partitioning ideas have often been used to define options around the bottleneck states at the boundary of each partition (Dean and Lin 1995; Menache, Mannor, and Shimkin 2002; Simsek, Wolfe, and Barto 2005; Botvinick, Niv, and Barto 2009; Chaganty, Gaur, and Ravindran 2012; Bouvrie and Maggioni 2012; Bacon 2013; Krishnamurthy et al. 2016; Machado, Bellemare, and Bowling 2017).

The bottleneck concept can be challenging to turn into practical and scalable algorithms. One important reason is the need for vast quantities of data (sometimes expert data, which is hard to obtain). Moreover, in the graph-theoretic formulation, the underlying state connectivity graph of the MDP must be approximated first, before the search for bottlenecks. While some progress has been made recently (Machado, Bellemare, and Bowling 2017), the approximation step often renders the graph perspective incompatible with online implementations over continuous spaces.

Desiderata

In our work on the discovery problem, a liberating decision has been to momentarily give the word *discovery* a break in order to refocus our attention on learning. After all, as reinforcement learning researchers, learning is what gets us up in the morning after coffee … and that seemingly innocuous change from *discovering* options to *learning* options had a significant impact on our understanding of the problem and the kind of properties that our algorithms should have.

The terminology of learning options had also been used in the past, but mostly in the context of subgoal (Sutton, Precup, and Singh 1999) or pseudo-reward (Dietterich 1998) methods, which allow for leveraging such external information in order to learn the policies and termination conditions of options, by treating each option as an MDP on its own. Given that the environment allows for arbitrary resets, the policy of an option would be initialized within its initiation set and executed until its termination condition was met. This approach leads to a process that separates learning the internal information for the options both from learning the policy over options and from learning useful subgoals or pseudo-reward functions.

Our desire to avoid this kind of partitioned learning led us to embrace a more integrative and continual perspective. We asked ourselves, Wouldn't it be possible to learn, at all times, the elements of all options and the policy governing them? This way of thinking also allows us to consider the question of optimality of a set of options. This issue is especially apparent in the subgoal and reward settings, and pertains to the fact that locally optimal options may not lead to optimality of the overall system (Minsky 1961; Watkins 1989; Dietterich 2000). Thus, we wanted to address this mismatch by learning options that were aligned with a well-defined objective for the system as a whole.

The end-to-end perspective not only provides this alignment with a given objective, but also puts learning in the hands of the system. We are thereby strengthening the meaning of options as internal abstractions belonging exclusively to an agent, and not to the environment, as latent variables: that is, options play a subjective role (Tanner et al. 2007; Sutton 2012). In this sense, the study of options is intrinsically phenomenological and as once expressed by Stanislaw Ulam:

> […] what you are describing is not an object, but a function, a role that is inextricably tied to some context. Take away the context, and the meaning also disappears. (Rota 1986, 2)

Pushing the responsibility of learning good options to the agent was also a way to prevent ourselves from biasing towards the kind of options that we thought the system should have. Specifically, we wanted to explore beyond bottleneck options, letting options emerge from learning only if deemed useful by the agent. Araújo and Davids (2011) argue that ascribing behavior externally in terms of personal features rather than within the agent-environment relation causes an organismic asymmetry and a lost sense of private (Sutton 2012) directed purpose within the agent. In order to bring the balance back, a switch must be made from viewing options as external symbolic objects to viewing them in a way that emphasizes their functional relationship within a system and its environment. Option discovery then becomes more a process of attenuation and adaptation (Araújo and Davids 2011) to the key properties of the environment.

An Architecture for Learning Options

The actor-critic architecture (Sutton 1984) lends itself naturally to our continual learning perspective, because it is capable of leveraging the same stream of experience not only to learn about values, but also to update a policy that has its own separate parameterization. Furthermore, actor-critic architectures can be implemented in a fully incremental fashion (Sutton 1984; Sutton et al. 1999), learning at all times. This architecture is also compatible with the average reward formulation (Puterman 1994) in the continuing (nonepisodic) setting (Sutton and Barto 1998).

The actor-critic approach is similar to policy iteration (Puterman 1994) in the sense that they both

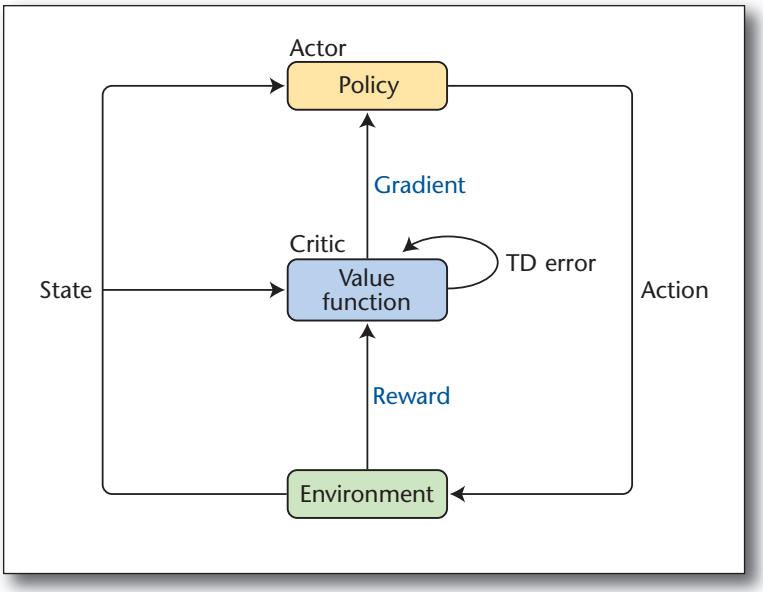

Figure 1. The Original Actor-Critic Architecture.

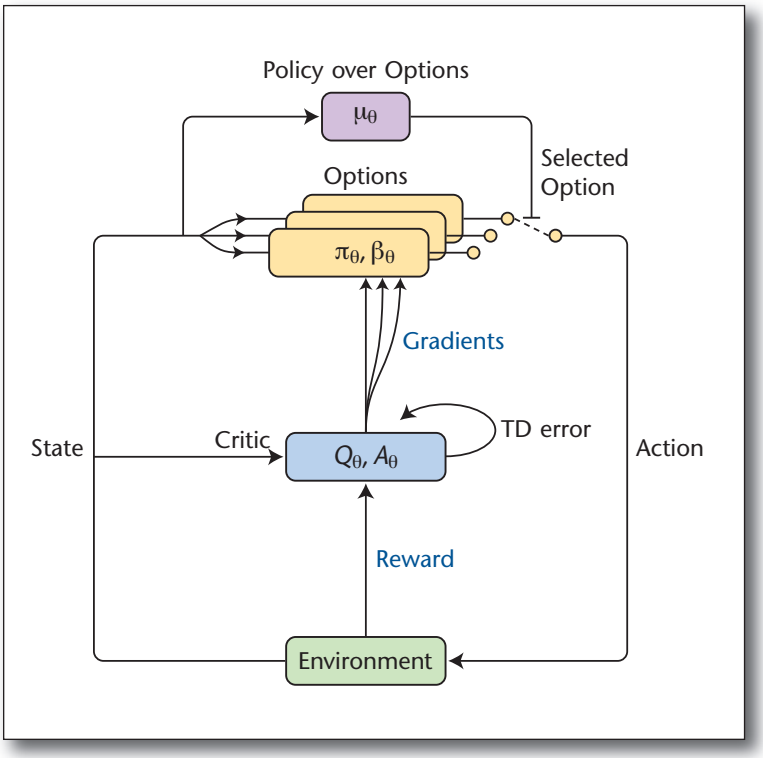

Figure 2. Our Proposed Option-Critic Architecture.

decouple the problem of policy evaluation in the critic from improvement of a policy in the actor. This separation of concerns between the specification of an objective and the means to achieve it is a powerful concept that eases our alignment goal: the ability to set an optimization target and to learn the right solution accordingly. Another benefit of the actor-critic approach is the flexibility in choosing the class of policies represented in the actor. Using a combination of randomized policies and approximate value function, it is possible to seamlessly handle continuous spaces of actions and states.

Given any differentiable parameterization of a randomized policy, the policy gradient theorem (Sutton et al. 1999; Konda and Tsitsiklis 2000) provides an expression for the gradient of either the expected discounted return or the average reward criterion with respect to the parameters of the policy. The main result can be stated rather simply: if an action is good, the policy gradient will update parameters to make that action more likely to be chosen again. The determination of whether an action was good is where the critic intervenes using value estimates. In an actor-critic architecture, the action values are learned in the critic in parallel with the policy updates. Figure 1 shows an actor-critic architecture with policy gradient updates and temporal difference learning in the critic.

The option-critic architecture of figure 2 is our adaptation (Bacon, Harb, and Precup 2017) of the actor-critic architecture for learning Markov options end to end by stochastic gradient ascent. As in regular policy gradient methods, we require the option policies to be represented by differentiable randomized policies. Similarly, the termination conditions need to be randomized, and the chosen parameterization must be differentiable. Therefore, we prefer to refer to termination conditions with parameters as simply *termination functions*.

Theoretical Results

Due to the requirements of our system, we cannot directly apply the policy gradient theorem to learn parameterized options, because working with options brings us to the SMDP framework. Yet working only at the SMDP level prevents us from considering the structure within an option, where the policy of an option is executing. We addressed this problem by using Markov options and by adopting the intra-option learning perspective (Sutton, Precup, and Singh 1999). As a consequence, the Markov property could be recovered both over actions and over state-option pairs.

The first step towards deriving gradient theorems for options was to describe precisely (Bacon, Harb, and Precup 2017) the probabilistic structure of the Markov chain, which takes the memory (stack) into account. We focused on a Markov chain over an augmented space, consisting of states, the option that is executing (in other words, the content of the stack), and the primitive action choice. It then sufficed to apply standard calculus tools in this chain, in order to derive gradients for the policy of an option and its termination function.

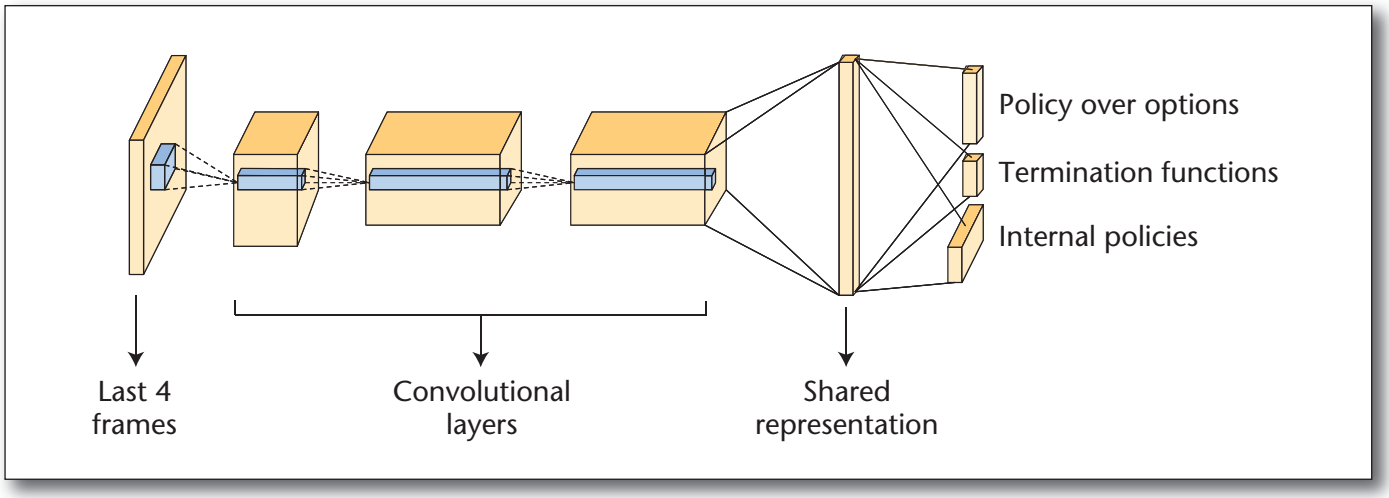

Figure 3. Network Architecture for Option-Critic in the ALE Environment.

In the gradient theorem for option policies, the interpretation for the original policy gradient theorem is maintained: if an action chosen within an option is useful, the gradient update will make it more likely to be picked again in the same state. This result also provides the global optimality property that we initially required. In fact, the effect of considering the choice of option as part of the state space results in a critic that provides estimates of the expected discounted return for the system as a whole, given that an action is taken in a certain state and under a certain option. Therefore, the gradient for option policies takes into account how a local change in the choice of actions would impact performance of the entire system.

The gradient theorem for termination functions also has a clear interpretation, but involves a different critic feedback than for option policies. The termination gradient makes an option more likely to terminate if there is no longer an advantage in following it. Conversely, if committing to an option is deemed advantageous by the critic, its probability of terminating should be decreased, so as to lengthen that option. The expression *advantageous*, loosely used up to now, is defined precisely in terms of the advantage function (Baird 1993): the difference between the value of a given option at a state and the expected value over all options. Interestingly, the termination gradient theorem for options can be seen as another instantiation of the interruption execution model (Sutton, Precup, and Singh 1999), whereby the policy over options commits to an option unless a better one can be taken.

Deep Options

In addition to options, a state representation can also be learned end to end using the option-critic architecture. With the Arcade Learning Environment (ALE) (Bellemare et al. 2013) in mind, we designed a parameterization based on the deep network architecture of the DQN algorithm (Mnih et al. 2015). Because in this environment the agent observes images, the first few layers of the network (figure 3) apply convolutions to a concatenation of the last four frames. In the penultimate layer, the high-level visual features extracted are combined in a shared representation across all options, termination functions, and value outputs.

While we could have chosen to also parameterize the policy over options, we decided to use an epsilon-greedy (Sutton and Barto 1998) policy over options derived from the value outputs. Therefore, the stream of computation going from input to value output, and epsilon-greedy policy, mirror the same design as DQN. However, the second path of computation ending in the option policies and termination functions necessitates randomization, according to the gradient theorems for options. Because the action space is discrete, we chose softmax (Sutton et al. 1999) for the option policies and sigmoids for the termination functions.

Different kinds of parameter updates are also necessary in each of the two streams. For the value updates and control over options, we used the idea of a target network from DQN, but in combination with intra-option Q-learning (Sutton, Precup, and Singh 1999) instead of Q-learning (Watkins 1989). By freezing the network for a fixed interval, the target for the value updates becomes more stationary and learning is more stable. We computed both kinds of updates at every step with samples coming from two different sources: from an experience replay buffer (Lin 1992) for learning values, and from fresh online samples for the options updates. The reason for not using replayed samples with option gradients (or policy gradients in general) was to ensure that our gradient estimates would truly come from the distribution of

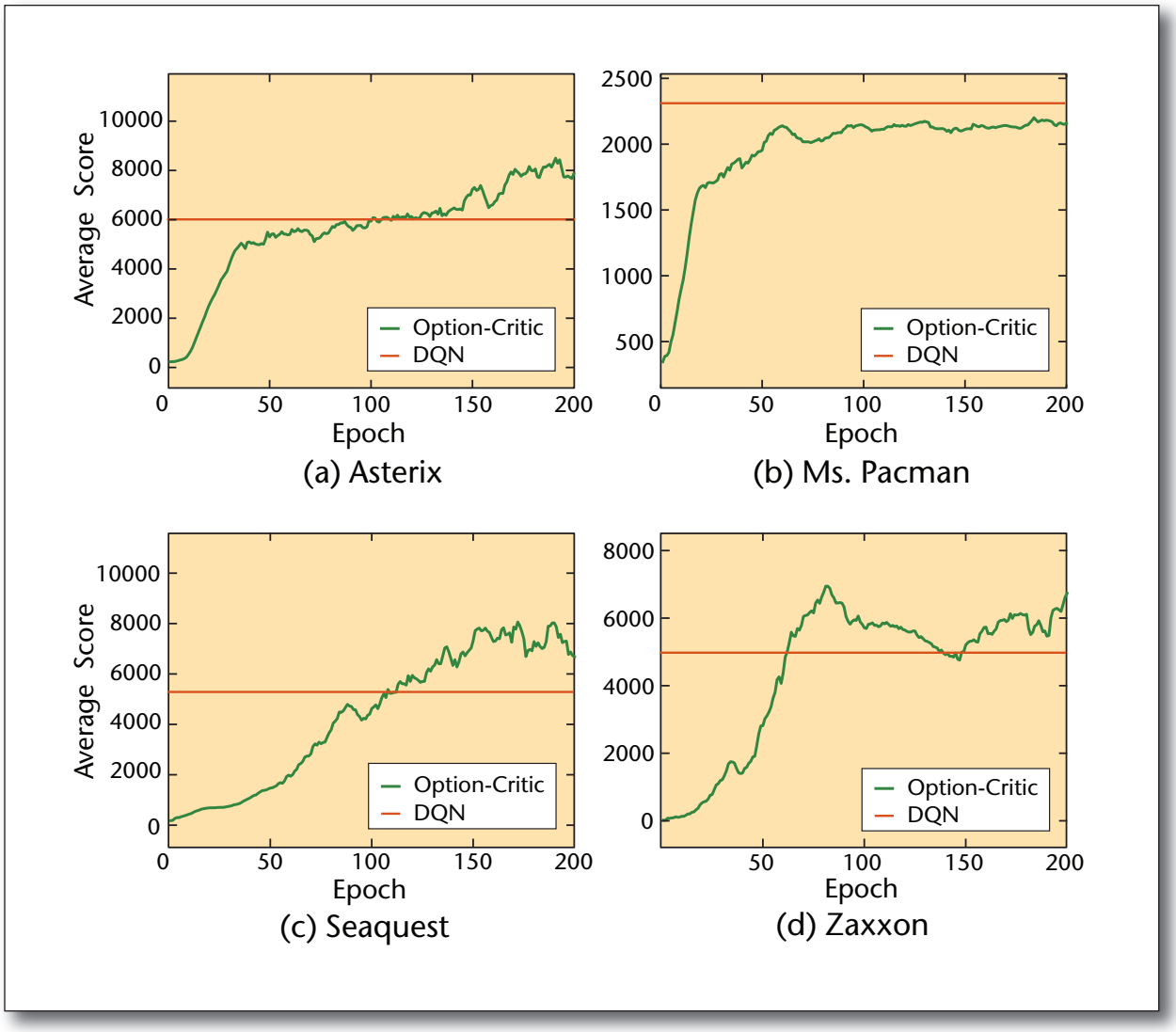

Figure 4. Learning Results with Option-critic in the Arcade Learning Environment.

interest: the stationary distribution of the online process.

From Zero to Options: Results in ALE

Could we learn from scratch, and within a single task, a set of options and the corresponding state representation and policy over options within a single task? We set out to answer this question in four representative tasks of the ALE domain : Asterix, Ms. Pacman, Seaquest, and Zaxxon. Even for simple grid environments, discovering options with complete autonomy used to require excessively large amounts of data and computation, or some form of prior experience in related tasks. Hence, learning options in ALE without any prespecification other than the goal of maximizing the discounted return would be a formidable challenge.

Despite the complexity of this endeavour, the combination of option-critic and our deep architecture outperformed the best reported DQN performance (figure 4) for the same total number of frames in the games Asterix, Ms. Pacman, and Seaquest. It is important to remember that all learning took place entirely within the same task at a rate and computational cost comparable to DQN. Apart from the option parameterization, the only parameter that we had to provide to our system was the number of desired options.

With the end-to-end approach underpinning the option-critic architecture, the options the system discovered were those it found useful for maximizing its expected discounted return. When learning two options in the game of Seaquest, the option-critic found a particularly vivid solution structure. At the SMDP level, typical trajectories show (figure 5) both options being used in an alternating fashion for an extended period each time. The partition of action sequences between the options is revealing. One

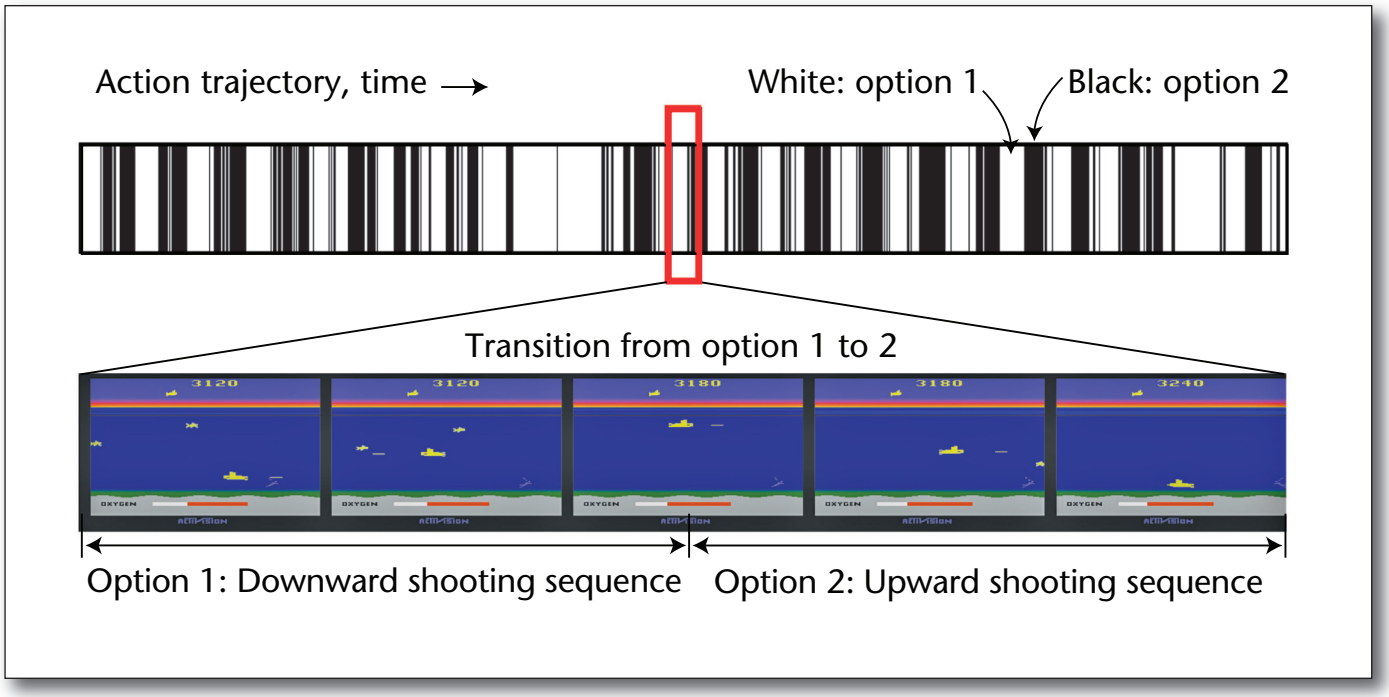

Figure 5. Interpretable and Specialized Options Found in the Game of Seaquest.

option specialized in action sequences going upwards on the way to replenish the oxygen, while the other executed only when descending below the water surface. Because nothing is specified a priori about options beyond the control objective, there is no mechanistic explanation for how these specific options came to be. However, we postulate that the elementary memory structure of options might represent aspects of the game dynamics having to do with oxygen management. When nearing low levels of oxygen, the agent must resist the urge to replenish the tank long enough to reach the surface. This would be more difficult to represent in a purely reactive fashion or without having recourse to temporal abstraction.

Conclusion and Future Work

The option-critic architecture is based on the general idea that the responsibility of learning should go to the learner (Drescher 1991). Instead of requiring an expert to make guesses about what aspects of the task and environment might be useful for building options, we let our system learn the right kind of options for the task at hand directly from its stream of experience. Building from the blueprints of policy gradient methods, we provide gradient theorems for options that allow their internal policies and termination conditions to be adjusted continually and simultaneous in order to actually solve the task. If desired, regularizers can also be added to this objective to make the system easily informable (Nilsson 1995). The option-critic architecture can then be instantiated through different implementations of the stochastic gradient ascent procedure associated with these gradients.

In spite of the success of the option-critic approach in Atari games, many questions have yet to be answered. For example, a common problem observed in practice is that, as the system becomes more proficient, the average duration of its options also tends to decrease. Considering the fact that the option-critic learns options for maximizing the expected return, this phenomenon is hardly surprising. From a pure optimization perspective, options are indeed useless for achieving optimal control: their optimal value function cannot be greater than the optimal value function of the MDP. As we also know, the optimal value function in a discounted MDP is always attainable by a greedy policy using only primitive actions (Puterman 1994). Hence, in a dynamic programming setting, having long temporally extended actions provides no benefit over primitive actions if optimal control is the only goal.

To prevent options from collapsing to primitive actions, we devised simple regularization strategies that could be incorporated readily to the objective without altering the learning architecture. For instance, the approach used in the Atari games consisted of adding a scalar margin to the advantage function used in the termination gradient. Intuitively, the effect of this margin term was to set a baseline of advantageousness in favor of maintaining the same option. We can also think of the margin as a

cost for switching options or for deliberating too long (Bacon and Precup 2015) — an interpretation that finds its roots in the bounded rationality framework (Simon 1957).

When departing from perfect rationality, boundedly rational systems are naturally pressured into making use of the regularities of their environment. When such systems are learning representations, only the essential elements can be captured because the resources — time, energy, computation, favorable opportunities — are scarce. Evaluation platforms that suitably reflect these conditions are not yet available in reinforcement learning. However, we are hoping to extend our experiments to a more naturalistic scenario by learning in a continuing fashion rather than in a single task.

From the bounded rationality perspective, providing "good enough" behavior at all times in an efficient manner might be the raison d'être for options. For example, consider a problem setting where the world does not wait for a carefully thought out best next action: maybe a rhinoceros is suddenly charging — no time to waste, acting is all that matters. There is an inherent cost in nature, but also in artificial systems, for carrying out excessive computation. Having options that are sufficiently temporally extended seems to provide a balance between fast decision making and high-level deliberative reasoning.

Initiation sets also provide a mechanism for managing computation. We avoided working with them in our option-critic approach, however, by making the assumption that options are available everywhere. By their very nature, initiation sets are not parameterized functions, so it is difficult to use our usual optimization toolkit to learn them end to end, as we did with termination functions. This problem needs to be addressed by first redefining the concept of initiation sets to initiation functions. Then, to derive a policy gradient–like theorem for initiation functions, we should also be capable of representing termination functions using a randomized and differentiable parameterization. In this case, the meaning of randomized initiation functions would have to be clarified in relation to the call-and-return execution model. Finally, we might need to enforce compositional properties of options so as to avoid an option terminating in a region of the state space where no other options can be taken. It is not clear at this point how this property could be tractably enforced in our optimization objective.

Acknowledgements

We gratefully acknowledge the funding received for this work from the Canadian National Science and Engineering Research Council (NSERC) and the Fonds de Recherche Quebecois – Nature et Technologie (FRQNT). We are very grateful to Jean Harb for the experimental results presented here, to Genevieve Fried for her feedback on this article, and to Rich Sutton for many inspiring conversations on options.

References

Araújo, D., and Davids, K. 2011. What Exactly Is Acquired During Skill Acquisition? *Journal of Consciousness Studies* 18(3–4): 7–23.

Bacon, P.-L. 2013. On the Bottleneck Concept for Options Discovery: Theoretical Underpinnings and Extension in Continuous State Spaces. Master's thesis, Dept. of Computer Science, McGill University.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The Option-Critic Architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1726–1734. Palo Alto, CA: AAAI Press.

Bacon, P.-L., and Precup, D. 2015. Learning with Options: Just Deliberate and Relax. Paper presented at the NIPS Bounded Optimality and Rational Metareasoning Workshop, Montréal, Québec, Canada, December 11.

Baird, L. C. 1993. Advantage Updating. Technical Report WL–TR-93-1146, Wright Laboratory, Wright-Patterson Air Force Base, OH.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47(1): 253–279.

Botvinick, M. M.; Niv, Y.; and Barto, A. C. 2009. Hierarchically Organized Behavior and Its Neural Foundations: A Reinforcement Learning Perspective. *Cognition* 113(3): 262–280. doi.org/10.1016/j.cognition.2008.08.011

Bouvrie, J. V., and Maggioni, M. 2012. Efficient Solution of Markov Decision Problems with Multiscale Representations. In *50th Annual Allerton Conference on Communication, Control, and Computing,* 474–481. Piscataway, NJ: Institute for Electrical and Electronics Engineers. doi.org/10.1109/Allerton.2012.6483256

Chaganty, A. T.; Gaur, P.; and Ravindran, B. 2012. Learning in a Small World. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS'12), volume 1, 391–397. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Dayan, P., and Hinton, G. E. 1992. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems 5,* 271–278. San Francisco: Morgan Kaufmann.

Dean, T., and Lin, S.-H. 1995. Decomposition Techniques for Planning in Stochastic Domains. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (IJCAI'95), volume 2, 1121–1127. San Francisco: Morgan Kaufmann.

Dietterich, T. G. 1998. The MAXQ Method for Hierarchical Reinforcement Learning. In *Proceedings of the Fifteenth International Conference on Machine Learning* (ICML'98), 118–126. San Francisco: Morgan Kaufmann Publishers.

Dietterich, T. G. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Artificial Intelligence Research 13: 227–303.

Drescher, G. L. 1991. Made-Up Minds: *A Constructivist Approach to Artificial Intelligence*. Cambridge, MA: The MIT Press.

Fikes, R.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and

Executing Generalized Robot Plans. *Artificial Intelligence* 3(1–3): 251–288. doi.org/10.1016/0004-3702(72)90051-3

Freeman, L. C. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40(1): 35–41. doi.org/10.2307/3033543

Hauskrecht, M.; Meuleau, N.; Kaelbling, L. P.; Dean, T. L.; and Boutilier, C. 1998. Hierarchical Solution of Markov Decision Processes Using Macro-Actions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (UAI'98), 220–229. San Francisco: Morgan Kaufmann Publishers.

Howard, R. A. 1963. Semi-Markovian Decision Processes. Paper presented at the 34th Session of the International Statistical Institute, Ottawa, Ontario, Canada, August 21–29.

Iba, G. A. 1989. A Heuristic Approach to the Discovery of Macro-Operators. *Machine Learning* 3: 285–317. doi.org/10.1007/BF00116836

Kaelbling, L. P. 1993. Hierarchical Learning in Stochastic Domains: Preliminary Results. In *Machine Learning, Proceedings of the Tenth International Conference,* 167–173. San Francisco: Morgan Kaufmann Publishers.

Konda, V. R., and Tsitsiklis, J. N. 2000. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems 12,* 1008–1014. Cambridge, MA: The MIT Press.

Konidaris, G., and Barto, A. G. 2009. Skill Discovery in Continuous Reinforcement Learning Domains Using Skill Chaining. In *Neural Information Processing Systems 22,* 1015–1023. Red Hook, NY: Curran Associates.

Korf, R. E. 1983. Learning to Solve Problems by Searching for Macro-Operators. PhD dissertation, Carnegie Mellon University, Pittsburgh, PA.

Krishnamurthy, R.; Lakshminarayanan, A. S.; Kumar, P.; and Ravindran, B. 2016. Hierarchical Reinforcement Learning Using Spatio-Temporal Abstractions and Deep Neural Networks. Unpublished MS. Computing Research Repository, May 2016 CoRR Abs/1605.05359. New York: Association for Computing Machinery.

Kuipers, B. 1979. Commonsense Knowledge of Space: Learning from Experience. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (IJCAI'79), volume 1, 499–501. San Francisco: Morgan Kaufmann Publishers Inc.

Lin, L.-J. 1992. Reinforcement Learning for Robots Using Neural Networks. PhD dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. H. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning* (ICML'17), 2295–2304.

Mayzner, M. S., and Gabriel, R. F. 1963. Information Chunking and Short-Term Retention. *The Journal of Psychology* 56(1): 161–164. doi.org/10.1080/00223980.1963.9923710

McGovern, A., and Barto, A. G. 2001. Automatic Discovery of Subgoals in Reinforcement Learning Using Diverse Density. In *Proceedings of the Eighteenth International Conference on Machine Learning* (ICML'01), 361–368. San Francisco: Morgan Kaufmann Publishers Inc.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-Cut — Dynamic Discovery of Sub-Goals in Reinforcement Learning. In *Machine Learning: ECML 2002, Proceedings of the 13th European Conference on Machine Learning*. Lecture Notes in Computer Science 2430, 295–306. Berlin: Springer.

Minsky, M. 1961. Steps Toward Artificial Intelligence. *Proceedings of the IRE* 49(1): 8–30. doi.org/10.1109/JRPROC.1961.287775

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, a. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-Level Control Through Deep Reinforcement Learning. *Nature* 518(7540): 529–533. doi.org/10.1038/nature14236

Niekum, S.; Osentoski, S.; Konidaris, G.; and Barto, A. G. 2012. Learning and Generalization of Complex Tasks from Unstructured Demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5239–5246. Piscataway, NJ: Institute for Electrical and Electronics Engineers. doi.org/10.1109/IROS.2012.6386006

Nilsson, N. 1995. Eye on the Prize. *AI Magazine* 16(2): 9–17.

Parr, R., and Russell, S. J. 1998. Reinforcement Learning with Hierarchies of Machines. In Jordan, M. I.; Kearns, M. J.; and Solla, S. A., eds., *Advances in Neural Information Processing Systems 10,* 1043–1049. Cambridge, MA: The MIT Press.

Piaget, J. 1937. *La Construction du Réel chez L'Enfant.* Paris: Editions Delachaux and Niestlé S.A.

Precup, D., and Sutton, R. S. 1997. Multi-Time Models for Temporally Abstract Planning. In *Advances in Neural Information Processing Systems 10,* 1050–1056. Cambridge, MA: The MIT Press. doi.org/10.1007/BFb0026709

Precup, D.; Sutton, R. S.; and Singh, S. P. 1998. Theoretical Results on Reinforcement Learning with Temporally Abstract Options. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning*. Lecture Notes in Computer Science 1398, 382–393. Berlin: Springer.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley and Sons. doi.org/10.1002/9780470316887

Rota, G.-C. 1986. In Memoriam of Stan Ulam — The Barrier of Meaning. *Physica D: Nonlinear Phenomena* 22(1): 1–3. doi.org/10.1016/0167-2789(86)90228-9

Rummery, G. A., and Niranjan, M. 1994. On-Line Q-Learning Using Connectionist Systems. Technical Report 166, Engineering Department, Cambridge University, Cambridge, UK.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529(7587): 484–489. doi.org/10.1038/nature16961

Simon, H. A. 1957. *Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in Society Setting*. New York: Wiley.

Simon, H. A. 1969. The Sciences of the Artificial. Cambridge, MA: The MIT Press.

Simsek, Ö., and Barto, A. G. 2008. Skill Characterization Based on Betweenness. In *Advances in Neural Information Processing 21: Proceedings of the 22nd Annual Conference,* 1497–1504. Red Hook, NY: Curran Associates. doi.org/10.1145/1102351.1102454

Simsek, O.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying Useful Subgoals in Reinforcement Learning by Local Graph

Partitioning. In *Proceedings of the 22nd International Conference on Machine Learning* (ICML'05), 816–823. New York: Association for Computing Machinery.

Singh, S. P. 1992. Reinforcement Learning with a Hierarchy of Abstract Models. In *Proceedings of the 10th National Conference on Artificial Intelligence*, 202–207. Menlo Park, CA: AAAI Press.

Stolle, M., and Precup, D. 2002. Learning Options in Reinforcement Learning. In *Abstraction, Reformulation and Approximation, 5th International Symposium* (SARA 2002). Lecture Notes in Computer Science 2371, 212–223. Berlin: Springer. doi.org/10.1007/3-540-45622-8_16

Sutton, R. S. 1984. Temporal Credit Assignment in Reinforcement Learning. Ph.D. dissertation, University of Massachusetts Amherst.

Sutton, R. S. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3(1):9–44. doi.org/10.1007/BF00115009

Sutton, R. S. 1995. TD Models: Modeling the World at a Mixture of Time Scales. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning*, 531–539. Amsterdam, The Netherlands: Elsevier

Sutton, R. S. 2012. Beyond Reward: The Problem of Knowledge and Data. In *Inductive Logic Programming: 21st International Conference*, 2–6. Berlin: Springer.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning.* Cambridge, MA: The MIT Press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12,* 1057–1063. Cambridge, MA: The MIT Press.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1998. Intra-Option Learning About Temporally Abstract Actions. In *Proceedings of the Fifteenth International Conference on Machine Learning* (ICML'98), 556–564. San Francisco: Morgan Kaufmann.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1999. Between MDPS and Semi-MDPS: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence* 112(1-2):181–211. doi.org/10.1016/S0004-3702(99)00052-1

Tanner, B.; Bulitko, V.; Koop, A.; and Paduraru, C. 2007. Grounding Abstractions in Predictive State Representations. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (IJCAI'07), 1077–1082. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Tesauro, G. 1995. Temporal Difference Learning and TD-Gammon. *Communications of the ACM* 38(3):58–68.doi.org/10.1145/203330.203343

Tesauro, G.; Gondek, D.; Lenchner, J.; Fan, J.; and Prager, J. M. 2013. Analysis of Watson's Strategies for Playing Jeopardy! *Journal of Artificial Intelligence Research* 47: 205–251.

Thrun, S., and Schwartz, A. 1995. Finding Structure in Reinforcement Learning. In *Advances in Neural Information Processing Systems 7*. Cambridge, MA: The MIT Press.

Watkins, C. 1989. Learning from Delayed Rewards. PhD dissertation, King's College, Cambridge, UK.

Pierre-Luc Bacon is a PhD candidate in the School of Computer Science at McGill University, whose research interests are in reinforcement learning. He received the Outstanding Student Paper Award at AAAI'2017 for his work on the option-critic architecture.

Doina Precup shares her time between McGill University, where she codirects the Reasoning and Learning Lab, and DeepMind Montreal. Her research interests are in machine learning, especially reinforcement learning, reasoning and planning under uncertainty, and applications of these methods. She became a senior member of AAAI in 2015, a Canada research chair in 2016, and a senior fellow of CIFAR in 2017.