The option-critic architecture

Pierre-Luc Bacon and Doina Precup
Reasoning and Learning Lab
McGill University
{pbacon, dprecup}@cs.mcgill.ca

Abstract

Temporal abstraction has long been recognized as key to scaling up learning and planning in reinforcement learning. While planning with temporally extended actions in the options framework is well understood, learning such abstractions autonomously is still challenging. We present recent results extending the policy gradient theorem to options, and propose a new *option-critic* architecture.

1 Introduction

Temporal abstraction is the idea of representing abstract knowledge about course of actions at different, extended time scales. In reinforcement learning, the options framework [Sutton *et al.*, 1999; Precup, 2000] allows defining such courses of action, as well as planning with them in seamless fashion. However, the problem of *discovering temporally extended actions autonomously* (also known as skill acquisition) is still an open, despite much research effort [McGovern and Barto, 2001; Stolle and Precup, 2002; Menache *et al.*, 2002; Simsek and Barto, 2004, 2008; Konidaris *et al.*, 2011; Niekum and Barto, 2011]. In particular, this problem is difficult when the state space and action space are continuous. Much of the existing work is focusing on finding *subgoals*, ie. useful states that an agent might reach, then learning policies to achieve these subgoals. This approach has lead to interesting methods, but ones which are very "combinatorial" in flavour. Additionally, learning how to achieve the subgoals becomes a task in itself, which could be expensive in terms of data and computation time.

In this paper we present an alternative view, which does not differentiate between the problem of discovering options and that of learning their policies. Instead, we embrace a gradual process of learning both option policies and their termination conditions simultaneously, in a policy gradient framework. Based on the policy gradient theorem, we derive new results which enable us to blur the lines between *discovering* and *learning* options. Since reinforcement learning makes continual and *online* learning one of its foundational principle, we find important to incorporate this view with that of *discovering options* (which we will refer from now as simply *learning options*).

2 Preliminaries and notation

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathrm{P}, r \rangle$ consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition probability distribution $\mathrm{P} : \mathcal{S} \times \mathcal{A} \to (\mathcal{S} \to [0, 1])$ over the next states and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. A stochastic policy is a conditional distribution over actions given a state $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ while a deterministic policy is a mapping $\pi : \mathcal{S} \to \mathcal{A}$. In this paper, we will consider the problem of maximizing the cumulative discounted return, defined as:

$$\mathbb{E}\left\{ \sum_{t=1}^{\infty} \gamma^{t-1} r_t \;\middle|\; s_0, \pi \right\} \tag{1}$$

The optimal control problem consists in finding a policy which maximizes this objective. A policy is optimal if its underlying value function is also optimal, a statement captured by the so-called Bellman optimality equations:

$$V^*(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} \mathrm{P}\left(s' \mid s, a\right) V^*(s') \right) \tag{2}$$

Equation (2) can be solved with dynammic programming methods such as value iteration or policy iteration [Puterman, 1994], if a model of the system can be acquired, or by temporal Difference (TD) learning [Sutton, 1988] is a model is too expensive to compute. A policy can then be computed based on the estimated value function.

2.1 Policy gradient

One class of methods which has proven very useful in continuous state and action spaces is that of policy gradient, in which the policy π_θ is parameterized using a function approximator. The policy gradient theorem of Sutton *et al.* [2000] addresses the problem of computing the gradient of expected return (1) with respect to the parameters of the policy. In the discounted case, the policy gradient is given by:

$$\frac{\partial}{\partial \theta} V(s_0) = \mathop{\mathbb{E}}_{s \sim \mu} \left\{ \sum_a \pi_\theta \left(a \mid s\right) Q(s,a) \;\middle|\; s_0, \pi_\theta \right\} \tag{3}$$

where μ is the stationary distribution of π_θ and Q denotes the action-value function. Samples from the expectation (3) can be used to update the parameters of the policy using stochastic gradient descent. The algorithmic implementation of the policy gradient theorem is a special case of the *actor-critic architecture* [Sutton, 1984]. As in policy iteration, actor-critic methods decouple policy evaluation from policy improvement. The critic values are used to compute modifications to the parameters defining the police π_θ.

2.2 Options framework

Options [Sutton *et al.*, 1999; Precup, 2000] formalize the idea of temporally extended actions, also sometimes called *skills* or *macro-actions*, by endowing agents with the ability to plan at different level of temporal abstraction. More precisely, a Markovian option $\omega \in \Omega$ is a triple $\langle \mathcal{I}_\omega, \pi_\omega, \beta_\omega \rangle$ consisting in an initiation set $\mathcal{I}_\omega \subseteq \mathcal{S}$, a policy π_ω (stochastic or deterministic) and a termination function $\beta_\omega : \mathcal{S} \to [0,1]$.

In the *call-and-return* execution model, an agent initially picks an option according to its policy over options and subsequently uses the policy of the chosen option until it terminates, at which point this procedure is repeated. Planning with options consists in solving the optimal control problem over the set of options and their associated reward and transition models. Reward models express the expected discounted cumulative return that can be obtained by choosing a particular option and executing it to termination. They can be succinctly expressed with the following Bellman-like equations:

$$r(s,\omega) = \sum_a \pi_\omega \left(a \mid s\right) \left[r(s,a) + \gamma \sum_{s'} \mathrm{P}\left(s' \mid s, a\right) \left(1 - \beta_\omega(s') r(s', \omega)\right) \right] \tag{4}$$

Similarly, transition models express the expected discounted probability of transitioning to some given state upon termination.

Because of their Bellman-like equations, reward and transition models can be computed from a model of the MDP, or learned using methods such as intra-option learning [Sutton *et al.*, 1999] or LSTD [Sorg and Singh, 2010]. Given the option models, the control problem of determining an optimal policy over options can be solved by noticing that options induce a Semi-Markov Decision Process over the original MDP (see [Puterman, 1994] for a theoretical treatment of SMDPs). Classical dynamic programming methods such as value iteration and policy iteration [Puterman, 1994] or Dyna planning [Sutton, 1991; Sorg and Singh, 2010] can be used to provide a solution.

3 Learning options

A typical approach for *learning options* is to use pseudo-rewards [Dietterich, 2000; Precup, 2000] or subgoal methods Sutton *et al.* [1999]. Under this approach, the termination function and initiation sets are typically engineered, a "pseudo-reward" function is added to the original rewards of the MDP, and each option is learned separately to optimize the sum of these (typically through SARSA [Rummery and Niranjan, 1994] or Q-learning [Watkins, 1989]). Given the fixed set of learned options, a policy over options can be obtained by learning or planning.

In this work, we try to adopt a more continual perspective on the problem of learning options. At any time, we would like to distill all of the available experience into every component of our system: value function and policy over options, internal option policies and termination functions (as well as their models, if desired). Our main assumption is that option policies and termination functions are stochastic and representable with parametrized differentiable functions approximators. We develop new results for learning options by using as blueprint the policy gradient theorem [Sutton *et al.*, 2000] for iteratively improving the set of options by stochastic gradient descent.

The policy gradient theorem relies on the existence of a stationary distribution over states. As pointed out originally in [Sutton *et al.*, 1999], even in the presence of an MDP structure and Markov options, the induced *flat* process over primitive actions is not Markovian. That is, the current state and choice of primitive action are not sufficient to fully specify the probability distribution over next states. One also needs to remember the current executing option at every step, thus forming an augmented state space in $\tilde{\mathcal{S}} \equiv \mathcal{S} \times \Omega$. As shown in the next sections, the augmented state space will only serve as theoretical construct and fortunately will not appear explicitly in our final results.

3.1 Learning policies within options

We need to compute the gradient of the value function $V_\Omega(\tilde{s}) \equiv Q_\Omega(s, w)$ marginalized over the primitive actions for the policy of an option (referred from now on as the *intra-option* policy):

$$\frac{\partial}{\partial\theta} Q_\Omega(s,\omega) = \frac{\partial}{\partial\theta} \sum_a \pi_{\omega,\theta}\left(a \mid s\right) Q_U(s,\omega,a)$$

$$Q_U(s,\omega,a) = r(s,a) + \gamma \sum_{s'} \mathrm{P}\left(s' \mid s,a\right) U(s',\omega) \tag{5}$$

$$U(s,\omega) = (1-\beta_\omega(s))Q_\Omega(s,\omega) + \beta_\omega(s)V_\Omega \tag{6}$$

We will refer to Q_U in (5) as the *option-value function upon arrival* [1] since $r(s,a)$ is the reward accrued upon arrival in s' and where, by the execution model of options, termination of the current option is allowed. Note that the option-value function could be recovered from Q_U by summing over the primitive actions:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega\left(a \mid s\right) Q_U(s,\omega,a) \tag{7}$$

The derivation of the gradient (omitted for lack of space) then relies on simple calculus and on recognizing the k-steps augmented transition probabilities in the expansion. This observation allows us to transform the recursion into an expectation over the stationary distribution in the augmented state space.

Theorem 1 (Intra-option policy gradient theorem)**.** *Given a set of fixed Markov options and a fixed policy over them, in the start-state formulation,*

$$\frac{\partial Q_\Omega(s_0,\omega_0)}{\partial\theta} = \mathop{\mathbb{E}}_{(s,\omega)\sim\mu}\left\{ \sum_a \frac{\partial}{\partial\theta}\pi_{\omega,\theta}\left(a \mid s\right) Q_U(s,\omega,a) \;\middle|\; s_0,\omega_0 \right\} \tag{8}$$

Proof: *Provided in a separate appendix.*

[1] The choice of notation and expression *upon arrival* is in reference to equation 20 of Sutton *et al.* [1999] in the derivation of the *intra-option learning* algorithm.

Intuitively, the presence of Q_U in the gradient stems from the need to account for the change in the expected value over the options given a change at the primitive level. This idea contrasts with the *local* nature of pseudo-reward methods in which option improvement is generally oblivious to its *global* effect.

When dealing with large action spaces, the summation over actions within the expectation in (3.1) can be cumbersome. Fortunately, we can get rid of it by a *change of probability measure* [Rubinstein *et al.*, 2007] and a simple calculus trick. The intra-option policy gradient theorem can then be written as:

$$\frac{\partial Q_\Omega(s_0,\omega_0)}{\partial\theta} = \mathop{\mathbb{E}}_{(s,\omega)\sim\mu}\left\{\mathop{\mathbb{E}}_{a\sim\pi_\omega}\left\{\frac{\partial}{\partial\theta}\log\pi_{\omega,\theta}\left(a \mid s\right)Q_U(s,\omega,a)\;\middle|\; s,\omega\right\}\;\middle|\; s_0,\omega_0\right\}$$

This transformation has often been exploited in a similar context, e.g. by Williams [1992]; Peters *et al.* [2005]; Degris *et al.* [2012]; Silver *et al.* [2014] .

3.2 Learning to terminate

We will now turn our attention to the problem of computing gradients for the option termination functions. To avoid introducing additional symbols, we will overload our notation and write $\beta_{\omega,\theta}$ to stand for the termination function of the Markov option ω pararameterized by θ. In this case, we will think of the intra-option policies as being fixed and having their own, separate set of parameters.

Theorem 2 (Termination gradient theorem)**.** *Given a set of fixed Markov options and a fixed policy over them, in the start-state formulations,*

$$\frac{\partial Q_\Omega(s_0,\omega_0)}{\partial\theta} = \mathop{\mathbb{E}}_{(s,\omega)\sim\mu}\left\{\frac{\partial\beta_{\omega,\theta}(s)}{\partial\theta}\left(V_\Omega(s) - Q_\Omega(s,\omega)\right)\;\middle|\; s_0,\omega_0\right\} \tag{9}$$

Proof: *Provided in a separate appendix.*

This result suggests an interesting interpretation in terms of the advantage function [Baird, 1993]. Defined over options, the advantage function is the difference: $A_\Omega(s,\omega) \equiv Q_\Omega(s,\omega) - V_\Omega(s)$. The termination gradient (9) then becomes the expectation of the product of the gradient of the termination function and the negative of the advantage function over options. When the option choice is suboptimal with respect to the expected value over all options, the negative of the advantage function is positive and drives the gradient corrections up at a particular state. This has the effect of increasing the odds of terminating that particular state, which in turn allow the agent to pick a better option.

This result also seems to agree with the original intuition behind *interrupting options* [Sutton *et al.*, 1999] which consists in forcing termination whenever the value of $Q_\Omega(s,\omega)$ for the current option ω is less than $V_\Omega(s)$. Mann *et al.* [2014] recently studied the interruption mechanism as a form of *interrupting Bellman Operator*. We believe that our result could also be be understood under this view, as a gradient-based interrupting Bellman Operator.

4 Option-critic architecture

The algorithmic implementation of theorems 1 and 2 gives rise to the *option-critic* learning architecture (fig. 1), in reference to the gradient-based actor-critic architectures [Sutton, 1984; Peters *et al.*, 2005; Degris *et al.*, 2012]. Although option-critic is conceptually identical to actor-critic, we sought to make a distinction between our holistic approach to learning options and one in which intra-option policies would be learned with regular policy gradient methods in a pseudo-reward context.

Since two types of gradients are needed to learn the options, the *critic* part of the option-critic architecture consists in $Q_U(s,\omega,a)$ or the negative advantage function (or both). In this work, we do not seek to use a critic for learning the policy over options. Note that the problem of learning a parametrized policy over options can be solved readily using the policy gradient theorem (see section 2). Using options has the advantage of reducing a large (potentially continuous) set of primitive actions to a potentially much smaller set of discrete options. In this case, the policy over options can be found using planning methods over the options models.

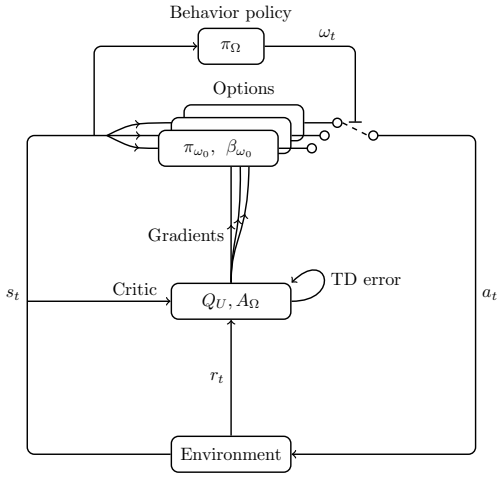


Figure 1: The option-critic architecture consists of a set of options, a policy over them and a critic. Gradients can be derived from the critic for both the intra-option policies and termination functions. The execution model is suggested pictorially by a *switch* $\perp$ over the *contacts* $-\circ$. Switching can only take place when a termination event is encountered.

4.1 Learning values upon arrival

The intra-option policy gradient theorem involves the option-value function upon arrival Q_U which is defined in terms of the the reward and primitive transition functions (assumed generally to be unknown or *unknownable*). Fortunately, we can easily devise an hybrid learning approach of using model-free updates which rely on current estimates of Q_Ω and V_Ω (potentially derived from a planner). A temporal-difference learning approach yields the following TD(0)-like update:

$$\delta_t = \Big(r(s,a) + \gamma U(s',\omega) - \hat{Q}_U(s,\omega,a)\Big)\,\frac{\partial}{\partial\theta}\hat{Q}_U(s,\omega,a) \tag{10}$$

The above equation can be incorporated within a stochastic gradient descent scheme for updating the parameters of $\hat{Q}_U(s,\omega,a)$, represented by either a linear or non-linear differentiable function approximator. Planning or learning methods can be used to obtain Q_Ω and π_Ω, which can then be used to form the U term (6) in (10).

5 Experiments

In order to illustrate our approach, we present some preliminary experiments in the four-rooms domain [Sutton *et al.*, 1999]. We fixed the initial state in the upper left corner and defined a terminal state in the lower right corner. A penalty of -1 is incurred at every step and for every action taken in the direction of a wall (resulting in a non-elastic collision). Primitive actions are defined as the one-step transitions to the next cell in each of the four cardinal directions: north, east, west, south. Any action may fail with probability 0.1, in which case the agent simply remains in the same state.

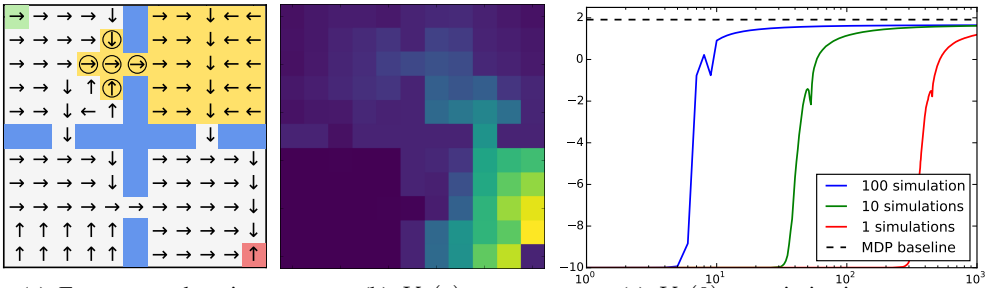
In order to focus on the gradient optimization scheme, we chose not to learn the options models from samples, but to compute them directly from the MDP model. The optimization procedure implemented (cf. Sections 5.1 and 5.2) is the following:

Repeat:

1. Compute the options models
2. Derive π_Ω, Q_Ω by planning over the options models using value iteration
3. Compute the expected values upon arrival Q_U using Q_Ω, π_Ω and the true MDP
4. Sample a given number of trajectories using the current set of options Ω and π_Ω

5. Perform a step of intra-option or termination gradient update for every sample of collected experience

5.1 Fixed subgoals conditions



(a) Four-rooms domain (b) $V_\Omega(s)$ (c) $V_\Omega(0)$ vs optimization steps

Figure 2: Four-rooms domain overlaid with a deterministic optimal MDP policy. The blue color is for *wall* cells, the green cell corresponds to the initial state and the red one is a goal state. The initation set of option 1 is highlighted with the color yellow. The circles represent the subgoal states for option 0

We first studied the behavior of the intra-option policy gradient algorithm when the initiation sets and subgoals are fixed by hand. In this case, options terminate with probability 0.9 in a hallway state and four of its incoming neighboring states (see figure 2a). We found necessary to define the subgoals not just at the hallway state for stability of learning. For example, if the action to go east at the north door would suddenly switch toward the wall, the agent would get stuck in that room. This would block the backups from reaching the initial state and make learning unstable. We fixed the termination probabilities to 0.9. Initiation sets were defined for every room and augmented with the subgoal states of the surrounding options to maintain the ability to chain options. We chose to parametrize the intra-option policies using the softmax distribution:

$$\pi_\omega\left(a \mid s\right) = \frac{\exp^{\theta^\intercal_{\pi,\omega}\varphi(s,a)}}{\sum_{a'}\exp^{\theta^\intercal_{\pi,\omega}\varphi(s,a)}} \qquad \frac{\partial}{\partial\theta}\log\pi_\omega\left(a \mid s\right) = \varphi(s,a) - \mathop{\mathbb{E}}_{a'\sim\pi_\omega}\left\{\varphi(s,a') \mid s\right\}$$

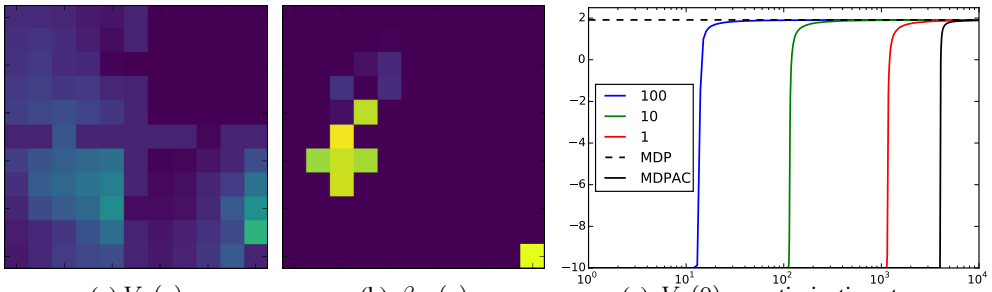
where φ is a state-action basis function. In our experiments, we used a simple a one-hot encoding of state-action pairs as basis functions, but we could also extend such softmax policies to *deep* energy-based ones as demonstrated by Heess *et al.* [2012].

We fixed the learning rate to 0.02 and evaluated our algorithm over 1000 optimization steps. The learning curves shown in 2c were computed for 1, 10 and 100 trajectories (optimization step 4) and a fixed maximum number of steps per trajectory of 100. Figure 2c shows the value estimate of the current set of options from state 0 for the three different conditions. Note that for this environment, the MDP value from the same state is an upper bound on the achievable return with our options. With a discount factor set to 0.9, we computed the MDP solution using value iteration and found the value from the initial state to be 1.91304. The intra-option policy gradient approaches the MDP solution given more optimization steps and sampled trajectories. Figure 2b shows the value function V_Ω after 40 optimization steps with 10 Monte-Carlo simulations per optimization step. We see that most of the value from the goal state had already been propagated backwards to the initial state through option 0 (upper left), 1 (upper right), and 3 (lower right).

5.2 Learning both internal policies and terminations

In this experiment we tried to learn the option policies simultaneously with the termination functions. We set the learning rates for both to 0.05. We used the same softmax parametrization as in the previous experiment but chose to represent the termination functions using the hyperbolic tangent function:

$$\beta_\omega(s) = \tanh(\theta^\intercal_{\beta,\omega}\varphi(s)) \qquad \frac{\partial}{\partial\theta}\beta_\omega(s) = (1-\tanh^2(\theta^\intercal_{\beta,\omega}\varphi(s)))\varphi(s)$$



(a) $V_\Omega(s)$ (b) $\beta_{\omega_0}(s)$ (c) $V_\Omega(0)$ vs optimization steps

Figure 3: Simultaneous learning of option policies and termination functions

where φ was once again defined as a one-hot encoding. We initially experimented with the sigmoid function but found that it would be more sensitive to the weight initialization method. With zero weights, the sigmoid function would initially return termination probabilities of 0.5. This led to more switching events and the optimization process tended to diffuse the termination gradients over more states. With the $\tanh$ funtion on the other hand, the termination probabilities start at 0 with zero-initialized weights. This helps exploration by allowing the agent to commit to its options for a longer time, and eliminates chattering in the final policy over options. Fixing the initiation sets as in the previous experiment also helped to keep the learning away from degenerate solutions. The agent converges to a policy over options taking an option in room 0, followed by a second one through rooms 2 and 3. The termination probabilities for option 0 are shown in 3b and seem to agree with our intuition of hallways as being useful subgoals.

We also compared the option-critic approach to an MDP-based actor-critic one with a single softmax policy over primitive actions. As for the other experiments, we used a tabular representation for the basis functions, computed the critic $Q_{\mathcal{A}}(s, a)$ exactly and we set the learning rate to 0.05. We sampled one trajectory after every step of exact policy evaluation and used it to update the policy parameters. Under these conditions, the algorithm starts converging to the target value after 4000 steps (see "MDPAC" in 3c) compared to the 1000 steps of the options-based.

6 Discussion and related work

Theorems 1 and 2 provide a principled approach to learning options. However, without a biasing mechanism, optimizing just the return objective would lead in the limit to a solution based on primitive actions only. This degenerate solution could take the form of a diverse set of one-step options or one in which a single dominating option decides over the entire state space. While this type of solution is provably optimal for solving a single MDP, it is contrary to our intuitions that options should stay extended in time. This might arise naturally in a setting in which the agent's task changes over time, or might be induced through a reward-like differentiable regularizer. For example, a penalty could be incurred whenever a switching event occurs in order to favor *commitment* during option execution. The idea of regularizing options has recently been studied by Mann *et al.* [2014] in the context of interrupting options [Sutton *et al.*, 1999]. Theorem 2 might allow for some of these results to be generalized to continuous spaces.

As we observed in our first experiment with the fixed subgoals, it is rather crucial to maintain "chainable" or *composable* options. As part of future work, we would also like to express *composability* as part of the optimization criterion as well.

Comanici and Precup [2010] also considered the problem of improving the termination function but for the case of semi-Markov options modelled by a logistic distribution over the accumulated features since initiation. The idea of compositionality of options was leveraged by Silver and Ciosek [2012] to dynamically chain options through successively longer temporal abstractions. Levy and Shimkin [2011] adapted the Natural Actor-Critic [Peters *et al.*, 2005] framework to learning options by taking a literal approach to the augmented state space construction. The termination problem was treated by viewing stopping events as additional control actions available to the agent. While

theoretically correct, the explicit construction of the augmented state and action spaces hides some of the useful semantics that we have been able to expose directly with our results.

7 Conclusion

We presented two new results which allow the gradient of the policies within options and the termination functions to be computed so as to optimize the expected discounted return. We can then iteratively improve the set options with these gradients, which allows differentiable, linear or non-linear, stochastic parametrization of options. Our option-critic approach retains the architectural properties of its actor-critic parent. It addresses the need to handle continuous action spaces and reuse knowledge about *values* in a decoupled manner. We believe that the gradients expressions derived in this paper might open the way for the end-to-end training of reinforcement learning agents capable of simultaneously learning a representation of the features [Mnih *et al.*, 2013] and of temporally extended actions for control. Future work should address the problem of designing regularizers to induce certain desirable properties for the options, such as commitment, composability, or memory constraints. The convergence of our optimization approach is also unknown and some analysis along the lines of the two-timescale approach might be needed (see [Konda and Tsitsiklis, 2004] for the linear case). We are confident that our results could easily be carried to deterministic policy gradient [Silver *et al.*, 2014] as well as in to off-policy [Degris *et al.*, 2012] or natural gradient settings [Peters *et al.*, 2005].

References

Leemon C. Baird. Advantage updating. Technical Report WL–TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory, 1993.

Gheorghe Comanici and Doina Precup. Optimal policy switching algorithms for reinforcement learning. In *AAMAS*, pages 709–714, 2010.

Thomas Degris, Martha White, and Richard S. Sutton. Linear off-policy actor-critic. In *ICML*, 2012.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res. (JAIR)*, 13:227–303, 2000.

Nicolas Heess, David Silver, and Yee Whye Teh. Actor-critic reinforcement learning with energy-based policies. In *EWRL*, pages 43–58, 2012.

Vijay R. Konda and John N. Tsitsiklis. Convergence rate of linear two-time-scale stochastic approximation. *The Annals of Applied Probability*, 14(2):796–819, 2004.

George Konidaris, Scott Kuindersma, Roderic A. Grupen, and Andrew G. Barto. Autonomous skill acquisition on a mobile manipulator. In *AAAI*, 2011.

Kfir Y. Levy and Nahum Shimkin. Unified inter and intra options learning using policy gradient methods. In *EWRL*, pages 153–164, 2011.

Timothy Arthur Mann, Daniel J. Mankowitz, and Shie Mannor. Time-regularized interrupting options (TRIO). In *ICML*, pages 1350–1358, 2014.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, pages 361–368, 2001.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *ECML*, pages 295–306, 2002.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Scott Niekum and Andrew G. Barto. Clustering via dirichlet process mixture models for portable skill discovery. In *Lifelong Learning, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 7, 2011*, 2011.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *ECML*, pages 280–291, 2005.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts, Amherst, 2000.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Rubinstein, Reuven Y., and Dirk P. Kroese. *Simulation and the Monte Carlo Method (Wiley Series in Probability and Statistics)*. 2 edition, 2007.

G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

David Silver and Kamil Ciosek. Compositional planning using optimal option models. In *ICML*, 2012.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 2014.

Özgür Simsek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *ICML*, 2004.

Özgür Simsek and Andrew G. Barto. Skill characterization based on betweenness. In *NIPS*, pages 1497–1504, 2008.

Jonathan Sorg and Satinder P. Singh. Linear options. In *AAMAS*, pages 31–38, 2010.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation and Approximation, 5th International Symposium, SARA 2002, Kananaskis, Alberta, Canada, August 2-4, 2002, Proceedings*, pages 212–223, 2002.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063. 2000.

Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, 1984.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44, 1988.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163, 1991.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge, 1989.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.