Reinforcement learning beyond the Bellman equation: Exploring critic objectives using evolution

Abe Leite[1], Madhavun Candadai[1,2] and Eduardo J. Izquierdo[1,2]

[1]Cognitive Science Program, Indiana University Bloomington
[2]Luddy School of Informatics, Computing, and Engineering, Indiana University Bloomington
Corresponding email: abrahamjleite@gmail.com

Abstract

Living organisms learn on multiple time scales: evolutionary as well as individual-lifetime learning. These two learning modes are complementary: the innate phenotypes developed through evolution significantly influence lifetime learning. However, it is still unclear how these two learning methods interact and whether there is a benefit to part of the system being optimized on a different time scale using a population-based approach while the rest of it is trained on a different time-scale using an individualistic learning algorithm. In this work, we study the benefits of such a hybrid approach using an actor-critic framework where the critic part of an agent is optimized over evolutionary time based on its ability to train the actor part of an agent during its lifetime. Typically, critics are optimized on the same time-scale as the actor using the Bellman equation to represent long-term expected reward. We show that evolution can find a variety of different solutions that can still enable an actor to learn to perform a behavior during its lifetime. We also show that although the solutions found by evolution represent different functions, they all provide similar training signals during the lifetime. This suggests that learning on multiple time-scales can effectively simplify the overall optimization process in the actor-critic framework by finding one of many solutions that can still train an actor just as well. Furthermore, analysis of the evolved critics can yield additional possibilities for reinforcement learning beyond the Bellman equation.

Introduction

Animals are not born *tabula rasa*; they do not learn everything from scratch. Conversely, animals are not born equipped with all skills required to perform different behaviors. Instead, our genotypes encode innate knowledge that enables us to learn during our lifetime. This knowledge is expressed via the physical structure of our brains and bodies, as well as through basic, instinctual behaviors and drives such as suckling or hunger. It is optimized over evolutionary time such that it increases our chances of survival. Importantly, unlike lifetime learning that happens through rewards and supervisory signals, this *a priori* knowledge is acquired based on relative fitness of individuals in relation to a population. Thus, natural intelligence is shaped via a wide arsenal of learning approaches including evolutionary learning over generations of populations (Mitchell, 1998), and individual lifetime learning through reinforcement (Sutton and Barto, 2018), supervision (Mitchell et al., 1997) and others (Dickinson, 1980). Each of these methodologies have independently inspired learning algorithms with the aim of developing artificially intelligent systems with the robustness and flexibility of living organisms. However, these approaches tend to get used separately as optimization methodologies for artificial systems and only recently there is a growing interest in developing hybrid methodologies that incorporate evolutionary as well as lifetime learning. In such a situation, it is unclear what the roles of these two learning paradigms are. In this work we set out to explore what is learned when evolutionary approaches and reinforcement learning approaches are combined.

Actor-critic models of learning (Barto et al., 1983) provide a framework to model learning on an individual's timescale. The critic, usually modeled using a neural network, is often interpreted as a model of the world and represents the quality, Q, of taking an action given an environmental state. Such representations are often called Q-tables, Q-networks, or Q-maps. Typically, they are implemented by optimizing to predict the long-term expected reward for a given state and action. Throughout this paper, we use the term Q-maps to refer to the state/action-quality mapping, but we do not enforce the implementation mentioned above. On the other hand, the actor, also modeled as a neural network, generates actions to perform the behavior and thus serve as sensorimotor maps. The actor learns to act in an environment based on training signals from the critic. This model, thus, provides distinct sub-models that separate the representation of the task space (critic) from the performance of the behavior (actor). As such, we utilize these models to explore how evolution of task objective and learning to behave during the lifetime can interact.

In contemporary reinforcement learning strategies, the actor and critic are trained concurrently on the same time-scale (Fig. 1A). The critic is trained using rewards from the environment and the actor is trained to take the actions that would maximize the critic's assessment (Grondman et al.,

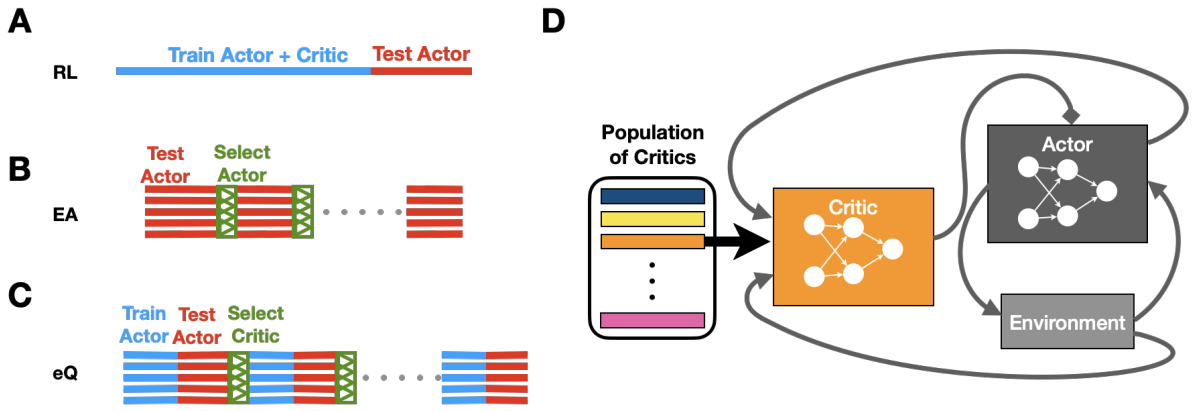

Figure 1: A schematic of the different actor-critic learning approaches [A] Reinforcement learning (RL) involves training both the actor and critic concurrently. [B] Evolutionary algorithms (EA) involve directly selecting an actor from a population over generations. [C] Our evolved Q-maps (eQ) approach involves evolving the critic over generations while training the actor during the lifetime. [D] A schematic of the eQ algorithm within a single generation. Critic, while remaining fixed, uses environmental states and proposed actions to generate training signals to modify the actor. Actor and environment engage in a two-way interaction.

2012). This is an entirely lifetime-learning approach to performing a task. Notably, the critic's objective is given by the Bellman equation that represents the expected long-term reward for taking a particular action in a given environmental state (Bellman, 1952). This approach and its variants have been shown to successfully perform a variety of tasks (Mnih et al., 2013; Silver et al., 2017; Mousavi et al., 2016). On the other hand, purely evolutionary approaches in Artificial Life generally directly evolve the parameters of the actor to perform the behavior without use of a critic (Floreano et al., 2008; Stanley et al., 2019) (Fig. 1B). This approach involves a population of actors that are selected based on their relative fitness. Thus, it is a purely population-based evolutionary approach and has also been shown to successfully perform a variety of tasks (Salimans et al., 2017; Such et al., 2017).

Approaches that combine evolutionary and lifetime learning have been explored in theoretical (Baldwin, 1896; Hinton and Nowlan, 1987; Suzuki and Arita, 2000; Nolfi and Parisi, 1996; Todd et al., 2020) as well in applied contexts within the actor-critic domain Ackley and Littman (1991). Recently, with the advent of deep reinforcement learning there has been a renewed interest in combining the two approaches with promising results (Houthooft et al., 2018; Fernando et al., 2017; Khadka and Tumer, 2018; Pourchot and Sigaud, 2018). While in some cases, the critics in these hybrid approaches have continued to be constrained to the Bellman equation, in other cases the critics have been part of the evolutionary learning. One approach that this work draws from is the Evolved Policy Gradients (EPG) algorithm (Houthooft et al., 2018), where the critic is evolved to generate training signals for an actor.

In this work, we aim to investigate how employing a combination of lifetime learning and evolution can be beneficial. Using the actor-critic framework we explore the different ways evolution can represent task information (critics) beyond the Bellman equation to train actors during their lifetime. Specifically, we devised a simplified evolutionary reinforcement learning approach we call evolved Q-maps or eQ, that utilizes an evolutionary strategy for the critics, which then train the actors during their lifetime (Fig. 1C). We compare the evolved critics with those obtained from an entirely lifetime learning approach, Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015). In order to develop Q-maps that are tractable for analysis, we used the classical inverted pendulum task (Brockman et al., 2016) as our target behavior. Our analysis revealed the following: First, eQ and DDPG are equally good in training actors to perform a task. Second, while Q-maps from eQ were faster to train fresh random actors of the same architecture, Q-maps from DDPG performed slightly better at training actors with different architectures than they were trained using. Third, while the Q-maps obtained from DDPG were all similar to each other, Q-maps from eQ were significantly different from them and also diverse within themselves. Finally, although the Q-maps from the eQ were encoding a diverse set of functions, the training signal they provided to the actors were very similar to each other as well to the training signal provided by the DDPG Q-maps. This suggests that several solutions beyond the Bellman equation exist to generate Q-maps that can train actors. Furthermore, although there may be several approaches to representing a task, the set of possible representation may be constrained by the rather narrow scope of the training signals that can enable learning to perform the behavior.

Methods

The goal of this work is explore the space of possible critics (Q-maps) over an evolutionary time-scale such that they can enable an actor to successfully learn to perform a behavior during the lifetime. This involves developing the actor-critic agent model, the environment or task, the evolutionary training of the critics, and lifetime learning of actors. These components of the model are described in this section. Further, we compare the evolved Q-maps with Q-maps obtained from a purely lifetime learning approach, Deep Deterministic Policy Gradients (DDPG), which is described at the end of this section.

Task design

For our experiments, we train and evaluate our agents on the classical inverted pendulum task. This is one of the easiest classic control tasks; in it an agent must exert a torque on the fulcrum of a rod pendulum in order to balance the pendulum facing upwards. There are three variables that characterize the behavior of the system: the current angle of the pendulum from the vertical in radians, denoted θ, the current angular velocity of the pendulum in radians counterclockwise per second, denoted ω, and the counterclockwise torque exerted on the pendulum, denoted τ. The angular velocity is limited to 8 radians per second and the torque is limited to 2 Newton-meters. The observation of the system given to the agent comprises the x- and y-coordinates of the tip of the pendulum and the angular velocity of the pendulum. (Coordinates are used rather than the angle itself to preserve continuity.) Each timestep, the agent provides the torque exerted as a single numerical value. Finally, the agent is provided with a reward signal that penalizes the pendulum falling or moving very quickly, or (to a lesser extent) a large torque being exerted. Over a standard 10-second episode with 50 simulated timesteps per second, the ideal reward given perfect initial conditions is 0; the expectation of ideal reward over randomly sampled initial conditions is in the ballpark of -130. We use this task because it is simple, low-dimensional, easy to interpret, and popular. See Figure 2(A) for a graphical depiction of the task and its variables.

An "episode" of the task involves resetting the environment with random initial conditions and stepping through one trial of agent-task interaction until a termination condition has been reached. Time is broken into discrete steps, and processing is traded between the agent and the task. Every timestep, a task provides an agent with an observation (known as S for state), a reward signal (known as R for reward), and a flag for whether or not its trajectory was just reset (known as T for termination). In return the agent provides the task with a pattern of motor activations (known as A for action). Together, all of the information from a single timestep transition can be collected in a tuple of the form $(S_t, A_t, R_{t\to t+1}, T_{t\to t+1}, S_{t+1})$, known colloquially as a SARTS tuple.

Agent model

Our model consists of agents, which include a Q-map critic neural network and an actor neural network. Each agent's critic had a feed-forward neural network connecting the task's 3-unit sensory input to two layers of 20 and 10 hidden units respectively, followed by a single-unit quality assessment output. A 1-unit proposed action input was concatenated to the first layer of hidden units. (Adding action inputs to the first hidden layer is a technique used in the DDPG algorithm to boost the magnitude of gradients.) This critic architecture is much smaller than typical for the DDPG algorithm, and results in somewhat unstable performance in DDPG; however, it facilitates the evolutionary algorithm and produces more compact and analyzable networks. The actor architecture is lifted directly from the paper introducing DDPG: the 3-unit sensory input, followed by two layers of 400 and 300 hidden units, followed by a 1-unit motor output. All units use the rectified linear unit signal function except for the critic's output, which has a linear response, and the actor's output, which uses the hyperbolic tangent signal function. $Q(S, A)$ denotes the Q-map's representation of the quality of an action given a particular state, and $A(S)$ denotes the actor's preferred action given a particular state.

Over the course of simulation, agents' nervous systems may or may not change. Much artificial evolution assumes that an agent's neural network weights are fixed by their genotypes and do not change over the course of each generation. If the organism's nervous system changes over time, then simulation is split into "training" and "evaluation" phases (as may be familiar from reinforcement learning setups). During the evaluation phase, the nervous system is kept fixed to whatever extent possible.

Evolved Q-maps (eQ)

Actor fitting Q-maps are representations of the value of taking certain actions in an environmental state. However, they cannot make decisions on what action an agent should take. Given a Q-map, an actor needs to be fit to the values represented by the Q-map to develop an agent that can act in an environment. The following actor-fitting algorithm is shared by all agents in this work.

The actor neural network is initialized with random parameters. Training episodes begin with the actor performing actions in the environment, with a small amount of correlated action noise added as in the DDPG algorithm. In order to improve the actor over the course of the training period, each agent maintains an "experience buffer" that contains a large number of SARTS tuples accumulated over several episodes. Every timestep, a number of SARTS tuples are sampled at random from the experience buffer, and the weights of the actor network are optimized to maximize the critic's assessment of the quality of the proposed action A'_t given the observation S_t. Practically, this involves taking the

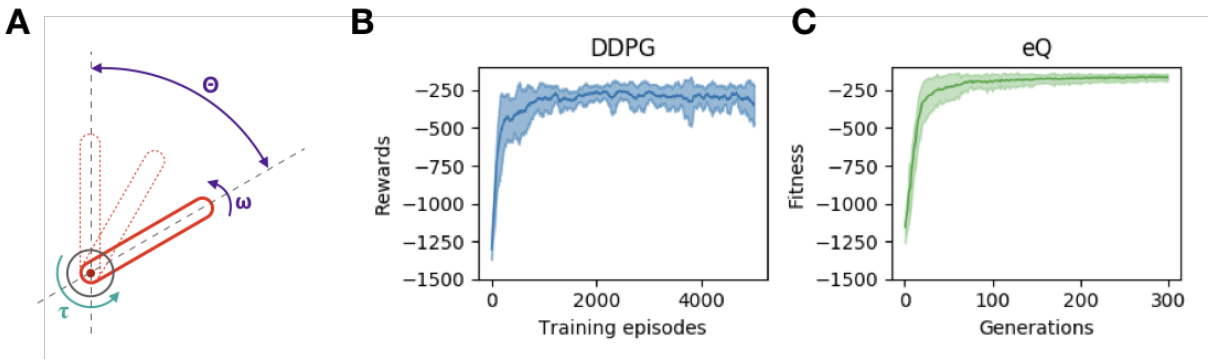

Figure 2: [A] a schematic of the inverted pendulum task, including the internal state variables θ and ω as well as the action variable τ. [B] Optimization curves for 20 runs of the DDPG algorithm. [C] Optimization curves for 20 runs of the eQ algorithm.

gradient of $Q(S_t, A'_t)$ with respect to A'_t, and then backpropagating that gradient through the actor network to obtain a weight update that we perform according to the Adam optimization algorithm (Kingma and Ba, 2014), a popular gradient descent algorithm that includes optimizations such as momentum in weight updates and a decaying learning rate. Intuitively, the actor tunes its actions so as the maximize the value that the critic provides.

It is important to note that as described here, actor fitting keeps the critic's Q-map fixed and uses it only to update the actor's flexible sensorimotor map. Actor fitting does not make use of reward signals from the environment, or indeed, any information from the environment beyond the distribution of states that occurs in its experience buffer.

Critic evolution We propose a simplified version of Houthooft et al. (2018) for artificially evolving Q-maps based on the fitness of the actions they favor, which we will term eQ for the remainder of the paper. In the eQ algorithm, a population of $N = 160$ critic networks is maintained, along with a single set of initial weights for an actor network. Every generation, N actor networks will be initialized from the set of initial weights, each paired with a critic, and then the actors will be fit to the critics based on the algorithm described above over the course of 50 training episodes. Following that, the fitness of the critics are evaluated by assessing the actors on 100 training episodes using a common, fixed set of initial conditions. To maintain similarity with DDPG, actors use time-correlated action noise as described in the following subsection during fitting and no action noise during assessment. At the end of each generation, an elitist fraction of the top 8 critics are kept as is, and 19 copies of them are made with increasing levels of Gaussian mutation noise added to the parameters of each copy. The standard deviation of the noise ranges from 0.08 to 0.8 units in the parameter space of the neural network. This gives a new population of 160 critics for the next generation.

Deep Deterministic Policy Gradients (DDPG)

Deep Deterministic Policy Gradients is one of the most popular reinforcement learning algorithms. It is used to learn a deterministic policy in which each stimulus corresponds with exactly one action. Although this is not explored in the present work, DDPG is often used to learn a policy that receives raw image data as input and processes this data using a deep convolutional neural network.

In the DDPG algorithm, the actor is fitted to the critic as described above. However, the critic's Q-map is updated concurrently with the actor to fit the Bellman equation, using gradient descent. Under the Bellman equation, the critic's quality assessment of an action given a state is optimized to represent the total expected long-term reward of picking that action from that state: the immediate reward, plus the best available long-term reward given the successor state following this transition, subject to some time discount factor. Given a successor state, the expected long-term reward of the actor's preferred action is used as a proxy for the state's best available long-term reward. In order to stably estimate the best available long-term reward at the successor state, "target networks" that track the critic and actor networks are used to determine this value, rather than determining it directly from the current critic and actor networks.

$$Q(S_t, A_t) \Leftarrow R_{t\to t+1} + \gamma Q_{\text{target}}(S_{t+1}, A'_{\text{target}}(S_{t+1})) \tag{1}$$

DDPG is highly dependent on time-correlated action noise to allow for exploration in action space. In time-correlated noise, such as the Ornstein-Uhlenbeck noise used by most implementations of DDPG, some amount of noise is added to a decaying accumulator each time step, and the accumulated noise is added to the agent's proposed action each time step. This ensures that the action at time t re-

mains predictive of the action at time $t+1$, which is crucial to distinguishing the effects of different actions for the purposes of the Bellman equation. We use Ornstein-Uhlenbeck noise with a variance σ of 0.2 and a decay factor θ of 0.15. Action noise is disabled during the evaluation period after actor fitting. Consult Lillicrap et al. (2015) for more details about the DDPG algorithm.

In order to obtain similarly performing critics from the eQ algorithm and the DDPG algorithm, each run of DDPG included a population of 160 actor/critic pairs. DDPG was carried out for 5000 episodes, at which point the population's performance had stabilized. From each run of each algorithm, the critic whose actor performed the best during run-end assessment was extracted for analysis.

Both DDPG and eQ algorithms share the same basic dynamic: the actor is fitted to the critic's assessments of its proposed actions over recent state transitions, and the critic is influenced by the actor's actions and the environment's dynamics. The key differences between the algorithms are the mechanisms in which and the time scale on which the critic is influenced. In DDPG, the critic's Q-map is fitted to the Bellman equation on recent experiences, with updates every time step of every episode. In eQ, critics that fit actors to do well in the environments provided are deemed more fit and selected for every generation.

Results

Our primary goal is to compare the space of Q-maps that produce adaptive behaviors with the space of Q-maps that fit the Bellman equation. First we use the eQ and DDPG algorithms to generate Q-maps in these spaces for the inverted pendulum task. This is one of the simplest continuous control tasks, and thus allows us to interpret Q-map representations of the task. We then compare the eQ and DDPG critics on four levels: their ability to produce an actor that performs a task, their ability to produce a critic that can train any actor to perform a task, the actual map of state-action values that the critic neural networks encode, and finally the training signal that they provide to the actor.

Performance of eQ and DDPG

Given the two approaches to generating Q-maps, we first evaluate their relative performance as methods of producing actors. We generated 20 critic/actor pairs using the eQ algorithm and 20 using the DDPG algorithm. The training curves for these two algorithms show that both eQ and DDPG generate critics that can train actors to perform the inverted pendulum task (Fig. 2).

Robustness of eQ and DDPG critics

While eQ- and DDPG-based optimization of critics were performed with a fixed actor initialization, the optimized critics should theoretically be able to train any random actor. To evaluate this, we performed the actor fitting algorithm on all forty of our obtained critics with 10 fresh actor initializations. Furthermore, we set out to test the robustness of these critics by evaluating their ability to train actors with two other architectures. Altogether, we repeated the actor fitting with three different hidden layer actor architectures: with two hidden layers of 400 and 300 units respectively (the original architecture, denoted by [3, 400, 300, 1]); with two hidden layers of 20 and 10 units respectively (denoted by [3, 20, 10, 1]); and finally with one hidden layer of 5 neurons (denoted by [3, 5, 1]). Since smaller networks exhibit more forgetting problems and less readily fit complex functions, we considered these to be increasing challenges for the critic networks. To obtain smooth training curves that directly measured actor performance, we trained in 25-episode bursts interspersed with 100-episode testing phases using a fixed, standardized set of 100 initial conditions.

Actor	Period	DDPG	eQ
[3,5,1]	50eps	-1129.5 $\pm$ 194.5*	-1347.2 $\pm$ 176.8*
[3,5,1]	5000eps	-287.3 $\pm$ 265.7	-409.7 $\pm$ 290.0
[3,20,10,1]	50eps	-701.6 $\pm$ 342.1*	-961.6 $\pm$ 384.7*
[3,20,10,1]	5000eps	-206.9 $\pm$ 158.8	-238.4 $\pm$ 222.4
[3,400,300,1]	50eps	-338.8 $\pm$ 253.0	-259.7 $\pm$ 188.6
[3,400,300,1]	1000eps	-207.8 $\pm$ 159.5	-238.5 $\pm$ 165.6

Table 1: DDPG (N=20) and eQ (N=20) critic performance training new actors (10 per critic). Includes training period of 50 episodes (as in eQ) and training until convergence. (*Significant to 95% confidence.)

Performance of actors averaged over 100 evaluation episodes after 50 episodes of training and at convergence can be found in Table 1. We found that at convergence, actors trained by DDPG critics did marginally better than those trained by eQ critics on average, irrespective of actor architecture. In addition, [3, 5, 1] and [3, 20, 10, 1] actors trained by DDPG critics did better throughout training (Fig. 3B,C), with statistically significant differences at 50 episodes for both architectures (Welch's t-test, $p < 0.05$). However, for the first 250 episodes, eQ critics train [3, 400, 300, 1] actors better than DDPG critics (Fig. 3A), though the difference at 50 episodes is marginal. The fact that eQ critics perform better than DDPG ones on the conditions they were optimized under but worse on other conditions indicates that eQ critics are more narrowly specialized than DDPG critics.

	DDPG	eQ
DDPG	0.76 $\pm$ 0.12	-0.03 $\pm$ 0.24
eQ	-0.03 $\pm$ 0.24	0.07 $\pm$ 0.53

Table 2: DDPG/eQ Q-map correlations: the average pair of two DDPG Q-maps exhibits a strong positive correlation, but other Q-maps are on average uncorrelated. Refer to Figure 5A.

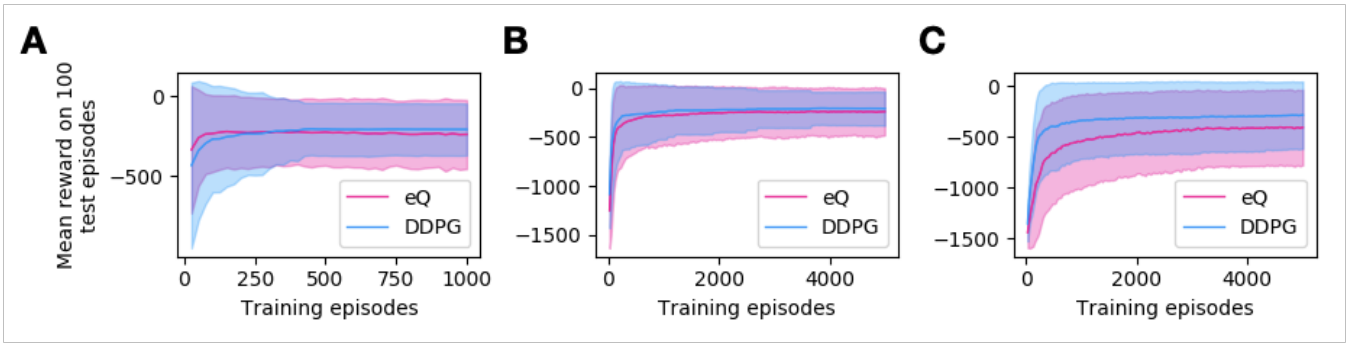

Figure 3: [A] eQ critics train new [3, 400, 300, 1] actors more quickly than DDPG critics. [B] On the other hand, DDPG critics consistently do better than eQ critics training new [3, 20, 10, 1] actors [C] and [3, 5, 1].

	DDPG	eQ
DDPG	0.53 ± 0.12	0.45 ± 0.13
eQ	0.45 ± 0.13	0.47 ± 0.13

Table 3: DDPG/eQ training signal correlations are all around 0.5, indicating on average moderate positive correlations between the training signals from any two critics. Refer to Figure 5B.

Comparison of Q-map activations

DDPG critics are optimized to represent the task per the Bellman equation. However, no such constraint is placed on the representation learned by the eQ critics since they are optimized solely based on the performance of their actors. Therefore, it is possible that eQ critics discover solutions beyond the Bellman equation. To study this, we directly compare the representations learned by the critics optimized using eQ and DDPG algorithms by comparing the activation maps over the entire range of state-action inputs. The activation maps are sampled uniformly across the state-action space with 200 angles, 9 angular velocities, and 40 torques spanning the valid range. This yields 72,000 observed activation values for each critic. Because the output neurons of the networks have linear signal functions, there are no strict bounds on the activation values, which range from -422.0 to 321.0 for eQ critics and -424.5 to 2.4 for DDPG critics. An illustrative example of the Q-map of a DDPG critic is shown in Figure 4A,B,C and that of an eQ critic is shown in Figure 4D,E,F. Representations of DDPG critics are interpretable based on the Bellman equation by having higher values for more favorable state-action pairs, and notably for more favorable states. For instance, when the angular velocity, ω, is 0 an angle of 0 has the highest value (pendulum is balanced at the top), but when the angular velocity is -8 an angle of 2.5 has the highest value (swinging upwards). However, eQ critics are not interpretable in the same way suggesting that they do not represent the same objective as the Bellman equation.

To quantitatively compare the representations of the 20 eQ and 20 DDPG critics, we performed a two-part analysis on the activation maps. To prepare, we calculate the correlation of the activation maps of each pair of critics – do the same state-action pairs lead to high or low activity for both critics? Then we performed a clustering analysis, which allows us to visually observe which activation maps appear by correlation to represent similar interpretations of the state-action space. Finally, we calculated the average correlation between different critics within each treatment group and the average correlation between critics across the treatment groups, which lets us easily represent how self-similar each group is and whether the two groups are more dissimilar from each other than they are internally. The clustering analysis revealed that DDPG critics are much more strongly correlated with each other than they are to the eQ critics, forming a single cluster before joining the rest of the critics (Fig. 5A). The average correlation analysis (Table 2) further revealed that while DDPG critics tend to agree quite strongly about which state-action pairs are the most favorable with a strong positive linear correlation, eQ critics have near-zero correlations with either each other or with DDPG critics, indicating that they do not converge to a single shared representation of the full state-action space. The eQ-eQ correlation has a higher standard deviation than the eQ-DDPG correlation, indicating that eQ critics have stronger linear correlations with each other (both positive and negative) than they do with DDPG critics.

Comparison of training signals

Each critic supplies a training signal to its paired actor by the derivative of its Q-map activation with respect to proposed action. Since the representations of the critics are so significantly different between eQ and DDPG, we set out to compare the training signals that they provide to the actors. The training signal at state S and action A is $\delta Q(S, A)/\delta A$. This derivative ranged from -9.9 to 7.4 for eQ critics and -85.8 to 89.7 for the DDPG critics, with mean magnitude 1.3 (eQ) and 4.8 (DDPG). The derivative is sampled across an-

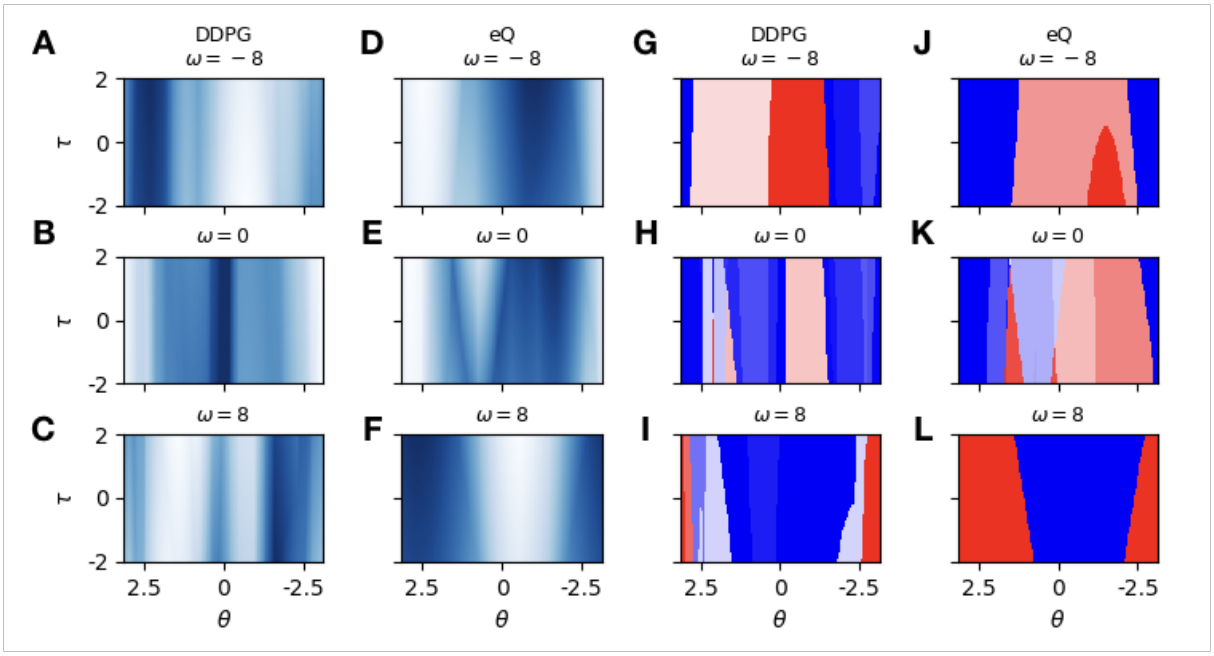

Figure 4: Illustrative example of a DDPG Q-map (A-C), an eQ Q-map (D-F); DDPG training signals (G-I) and eQ training signals (J-L). In Q-maps, deeper blue indicates a lower value $Q(S, A)$ relative to Q-values elsewhere in the state-action space; for the DDPG critic, more favorable states and actions are generally lighter than less favorable states and actions, but for the eQ critic there is no discernible pattern of activations by state. In training signal maps, blue indicates that a more clockwise (negative) torque is favorable and red indicates that a more counterclockwise (positive) torque is favorable.

gle, angular velocity, and torque as described in the previous subsection. An illustrative example of the training signals of a DDPG critic is shown in Figure 4G,H,I and that of an eQ critic is shown in Figure 4J,K,L. Both represent qualitatively similar decision boundaries.

The same quantitative analyses are performed on training signal maps as were performed for Q-maps. The clustering analysis (Fig. 5B) found that 11 of the 20 DDPG critics initially clustered together but that the remaining 9 DDPG critics were evenly intermingled with the eQ critics. This means that critics were intermingled overall, but that several DDPG critics exhibited a similar training pattern. Unlike in the raw activation case, the cluster of DDPG critics does not show a greater internal correlation than other – intermingled – clusters of critics. Consistent with these results, the average correlation analysis (Table 3) reveals that the training signals of DDPG-DDPG critic pairs exhibit a slightly higher correlation than that of eQ-DDPG or eQ-eQ critic pairs. All of these correlations are around 0.5, a moderate positive correlation. This suggests that although the two groups of critics encode different representation of the task, they ultimately provide similar training signals to the actor.

Discussion

In summary, using our simplified actor-critic model of evolutionary reinforcement learning, we demonstrate that there is a degenerate mapping between the evolved representation of the state-action space and the training signal that enables lifetime learning. Specifically, by evolving critics to train actors and comparing the evolved critics to the lifetime-learned critics, we demonstrate the following: First, an evolutionary reinforcement learning approach, eQ, is able to generate actors that are just as good as those generated by DDPG in performing the Inverted Pendulum task. Second, evolved Q-maps train fresh actors better than DDPG Q-maps under the conditions they were optimized for, but waiting for actors' convergence or varying the actor's architecture reverses this trend. This indicates that while both strategies produce functional critics, eQ produces more specialized Q-maps than DDPG. Third, most Q-maps generated by DDPG correlate strongly with each other but on average, Q-maps generated with eQ are correlated neither with each other nor with those generated by DDPG. Finally, any two Q-maps' training signals will have a moderate positive correlation across the state-action space, regardless of which way they were generated. The evolutionary reinforcement learning approach allowed us to explore other objectives for the critic beyond the Bellman equation. As a result, we found a diverse set of Q-maps that were just as good in their ability to train actors, broadening the definition of a useful critic beyond those that fit the Bellman equation. This opens up avenues of future research to characterize the functions that the evolved critics

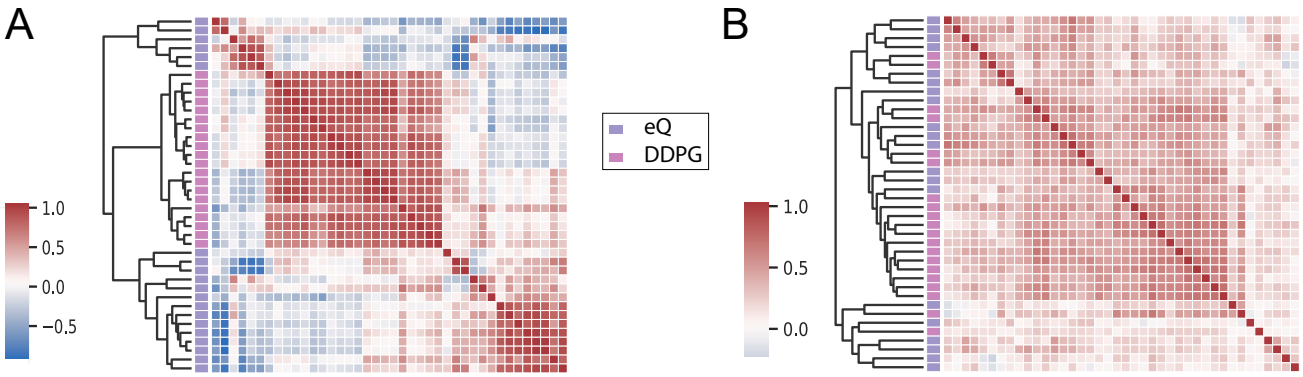

Figure 5: In general, while Q-maps from DDPG and eQ are very different, the gradients they supply to train actors are very similar. [A] Q-map activations from DDPG cluster together first before forming a cluster with eQ maps. [B] Hierarchical clustering of gradients from Q-maps reveals little discernible clustering of gradients from DDPG and eQ.

represent to potentially discover new reinforcement learning training methods.

Our work on whether training signals from eQ critics are meaningfully different from training signals from DDPG critics contributes to the literature of evaluating actor-critic models. From our results showing the variability of solutions that an algorithm can produce, we argue that it is advisable to report both within- and between-algorithm variability when comparing two algorithms. In related work, Houthooft et al. (2018) find that their evolved policy gradients algorithm produces training signals with a Spearman correlation of 0.5 to those of a standard reinforcement learning algorithm, and use this fact to argue that the two algorithms generate training signals "related to, but different from" each other. In our work, we found that multiple runs of DDPG produced critics whose training signals are only correlated to this level. Given the wide range of reinforcement learning algorithms under study, each of which might result in a different level of variability between runs, reporting within-algorithm variability will help to provide a richer picture of the differences between different algorithms.

This work provides evidence in support of the ongoing movement towards algorithms generating algorithms (Clune, 2019). In this work, we used a specific evolutionary/lifetime learning algorithm, replacing a machine learning objective with an unconstrained fitness-maximizing objective. We believe that further analysis of models obtained from such approaches has the potential to yield several new algorithms that would be impractical to discover through manual incremental development of existing algorithms. While this approach has focused on the actor-critic framework, there have been other non-actor-critic approaches to lifetime reinforcement learning (Lapan, 2018) that can be explored in a similar fashion. Altogether, hybrid approaches combining evolutionary and lifetime learning have demonstrated great promise towards the aim of developing new algorithms for developing artificial systems with the robustness and flexibility of natural systems.

Data availability

The simulation code and data files are publicly available in our research group's GitHub account: github.iu.edu/EASy/LeiteALife2020.

Acknowledgements

We are thankful for the reviewers' comments. This material is based upon work supported by the National Science Foundation under Grant No. 1845322. AL is supported by the Hutton Honors College of Indiana University, and the computation was supported by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

References

Ackley, D. and Littman, M. (1991). Interactions between learning and evolution. *Artificial life II*, 10:487–509.

Baldwin, J. M. (1896). A new factor in evolution. *The american naturalist*, 30(354):441–451.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Clune, J. (2019). Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*.

Dickinson, A. (1980). *Contemporary animal learning theory*, volume 1. CUP Archive.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. (2017). Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1):47–62.

Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex systems*, 1(3):495–502.

Houthooft, R., Chen, Y., Isola, P., Stadie, B., Wolski, F., Ho, O. J., and Abbeel, P. (2018). Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pages 5400–5409.

Khadka, S. and Tumer, K. (2018). Evolutionary reinforcement learning. *arXiv preprint arXiv:1805.07917*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lapan, M. (2018). *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitchell, T. M. et al. (1997). Machine learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mousavi, S. S., Schukat, M., and Howley, E. (2016). Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference*, pages 426–440. Springer.

Nolfi, S. and Parisi, D. (1996). Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior*, 5(1):75–98.

Pourchot, A. and Sigaud, O. (2018). Cem-rl: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222*.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Suzuki, R. and Arita, T. (2000). Interaction between evolution and learning in a population of globally or locally interacting agents. In *Proceedings of the Seventh International Conference on Neural Information Processing*, volume 738, page 743.

Todd, G., Candadai, M., and Izquierdo, E. J. (2020). Interaction between evolution and learning in nk fitness landscapes.