Ontogenetic and Phylogenetic Reinforcement Learning

Julian Togelius, Tom Schaul, Daan Wierstra, Christian Igel, Faustino Gomez, Jürgen Schmidhuber

Reinforcement learning (RL) problems come in many flavours, as do algorithms for solving them. It is currently not clear which of the commonly used RL benchmarks best measure an algorithm's capacity for solving real-world problems. Similarly, it is not clear which types of RL algorithms are best suited to solve which kinds of RL problems. Here we present some dimensions along the axes of which RL problems and algorithms can be varied to help distinguish them from each other. Based on results and arguments in the literature, we present some conjectures as to what algorithms should work best for particular types of problems, and argue that tunable RL benchmarks are needed in order to further understand the capabilities of RL algorithms.

1 Introduction

As defined by Sutton and Barto, any algorithm that can solve a reinforcement learning (RL) problem, which is defined by a (partially observable) Markov decision process, (PO)MDP, is an RL algorithm [19]. In the last few decades, a wide variety of algorithms have been used to successfully tackle RL problems in different guises. Surprisingly, these algorithms are based on very different principles. This is possibly due to the fact that they have been proposed and are actively studied within separate academic communities (e.g. machine learning, computational intelligence, computational neuroscience and control theory). There has been limited communication between these research fields, leading to insufficient analysis of the differences and similarities between these algorithms. It would therefore be a great boon to RL research to find a unified view, allowing us to understand the relative benefits of algorithms based on different principles.

At the same time, a large variety of RL problems has been defined and studied, varying from real-world continuous control problems to abstract discrete toy benchmarks, where real-world problems can be defined as problems that were not created with the purpose of testing RL algorithms. It is well known that some RL algorithms work well for some problems where other algorithms fail, but in many cases it is not clear what algorithms perform best under what conditions. The principal dimensions along which RL problems can vary are listed below.

The purpose of this short paper is to discuss some distinctions between types of RL problems and RL methods, especially between *ontogenetic* and *phylogenetic* methods, which in our experience is one of the distinctions that most clearly divides the disparate research communities concerned with RL in one form or another. We also make some conjectures about what types of methods would work best on what types of problems, and argue for the need of RL benchmarks that are tunable in important problem dimensions.

2 RL problem dimensions

Reinforcement learning problems may vary along multiple dimensions, for example:

- **Discrete vs. Continuous.** The environment's state, action and observation spaces can each be discrete, continuous or mixed.
- **Size and Dimensionality.** Apart from being continuous or discrete, the state, action and observation spaces may vary in their dimensionality. For instance the state can be represented by a single integer or by a visual scene. The size of discrete dimensions can vary from small (binary) to large (e.g. dictionaries).
- **State Space Structure.** There can be varying degrees of *structure* in the state space. Many benchmarks assume a certain locality (i.e. state transitions reach only a neighborhood of states) or have hierarchical properties. That structure is not necessarily ergodic, which thus allows for 'catastrophic' actions after which the agent cannot return to parts of the state space (e.g. a tabletop robot might fall off the table). There also exist exotic state representations based on, e.g., relations between logical predicates.
- **Stochasticity.** The problem can have varying degrees of stochasticity at different levels:
  - The state *transitions* may be stochastic. For example a robot's wheels might slip, so it does not know how far it will move when trying to move forward.
  - The *reward* function needs not be deterministic, for instance in a randomized game.
  - The *observations* may be stochastic, for example in the case of noisy sensor input.
  - There could be different *start states* drawn according to some random distribution.
- **Observability.** The environment can be *fully observable*, where the underlying state is directly accessible to the agent, or *partially observable*, where the agent can only make indirect, potentially stochastic *observations* of the state. This can impose a memory requirement on the agent as the only way to reliably infer the current state. In addition, observations can have varying degrees of redundancy, which in turn can make learning harder [21].
- **Generalization.** Observation representations may vary in the amount of meaningful structure encapsulating aspects of the transition model and enabling generalization to similar states.

- **Reward Regime.** Rewards can vary from a single signal at the end of an episode (e.g. when winning a game) to many informative intermediate rewards (which can correspond to sub-tasks).
- **Episodic vs Life-long.** In case the task breaks down into a sequence of separate finite episodes starting from a (distribution of) start state(s) we speak of *episodic* RL, otherwise we speak of *life-long* RL.
- **Number of Agents.** One can distinguish between typical single-agent RL and multi-agent RL, in which several learning entities interact [4]. Multi-agent RL can further be sub-divided, e.g. depending on whether the agents are co-operative, competitive, or some mixture of both.

3 RL methods

A large number of algorithms have been devised to solve RL problems. There are several ways to categorize them, here we propose a taxonomy and divide them into *ontogenetic* algorithms and *phylogenetic* algorithms.

3.1 Phylogenetic approaches

Phylogenetic algorithms (neuroevolution, for example) are those that *only* use a fitness function $f(\theta)$ to update its policy-defining parameters θ. This fitness function is typically some measure of rewards accrued during one or more episodes, and may be completely unknown to the algorithm. Phylogenetic methods notably do *not* keep track of the particular states visited each episode, and typically (but not always) retain a 'population' of several policies.

Phylogenetic methods treat the RL problem as a (black box) optimization problem, optimizing a policy for maximal accumulated reward over one or several episodes. In principle, any optimization method could be used, including local search methods like hill climbing and simulated annealing. More commonly, however, evolutionary algorithms like evolution strategies and genetic algorithms are used for phylogenetic RL [13]. These algorithms work by maintaining a population of policies. Each policy is assigned a fitness based on the accumulated reward over one or more episodes. Less fit policies are then removed from the population and replaced with new policies, constructed through combining and varying more fit policies. Some algorithms from swarm intelligence, e.g. particle swarm optimization, work according to similar principles and can also be used for phylogenetic RL.

3.2 Ontogenetic approaches

In contrast to phylogenetic approaches, ontogenetic algorithms (Q-learning, for example) can take into account the *full* information on which states were visited and which states yielded which rewards, and typically update a single policy. The algorithms described in the discipline-defining book *Reinforcement Learning* [19] (such as Q-learning [22] and Sarsa [16]) are all ontogenetic. They often build on the concept of a *value function* that maps states (or state-action pairs) to values such as expected future rewards. Then a policy is defined on top of the value function. A greedy policy, for instance, takes at each time step the action that leads to the state with highest value (according to the value function). Then reward is received from the environment, and the value function is updated using temporal difference methods.

A different family of ontogenetic RL algorithms are those based on policy gradient ascent. In the policy gradient framework [25, 2, 14], policies are stochastic and its parameters are updated directly using a gradient in the direction of better expected return. These methods, when applied to function approximators like neural networks [24], constitute an alternative to value-based methods that is more similar to phylogenetic methods in that they typically do not use value functions but represent policies directly, and also because they implicitly represent a distribution of policies (instead of a single greedy policy) because of their inherently stochastic actions [7]. Note that even though policy gradient methods perform updates using gradient ascent, they are nevertheless counted as ontogenetic methods because they use more information than just the fitness: what states were visited and which rewards were obtained at which time steps after what actions.

4 Which algorithms work best? For which problems?

There is no particular RL algorithm that performs better than other algorithms across all finite MDPs (certain universal RL methods can be proven to be optimal, however, these are either incomputable [10] or currently suffer from insurmountable computation overhead that prevent their practical use [17]). Different algorithms have different strengths and weaknesses, many of them currently unknown. However, the literature contains a number of theoretical arguments and empirical results suggesting the superiority of some families of algorithms over others for particular problem classes.

The theoretical results for ontogenetic methods are more advanced than for phylogenetic methods, in particular, convergence rates can be derived for some ontogenetic algorithms [3]. Phylogenetic methods also suffer more from the *credit assignment problem*, especially in stochastic or large domains, since only one single fitness measure is attributed to an entire episode roll-out (or several roll-outs). If there is stochasticity or noise inherent in the environment, many reruns of the same policy may be required to reliably estimate its performance. On the one hand, ontogenetic methods suffer less from this limitation as they can (at least in the fully observable case) attribute rewards and value to exactly those states that have actually been visited during an episode. On the other hand, many ontogenetic methods are directly based on ranking policies and do not require to estimate correct performance values or to estimate reliably performance gradients. This makes them more robust, because estimating a sufficiently good ranking under uncertainty or noise is much easier than estimating accurate values or even gradients [8]. Another limitation of phylogenetic approaches is that the difficulty grows prohibitively with the state space. Whereas ontogenetic methods can, in theory, exploit all information acquired during learning, phylogenetic methods cannot – they must rely solely on fitness values, hence suffering more from the credit assignment problem than do ontogenetic methods. This might lead one to suspect that, in large fully observable discrete state spaces that have inherent stochasticity, ontogenetic approaches would also scale better than phylogenetic ones. However, as of yet, neither formal proof nor systematical empirical evidence is available to support this. Establishing the relative scalability of ontogenetic and phylogenetic methods would be an important research contribution. Many temporal difference (TD) ontogenetic approaches, however, seem to have difficulty using function approximators (required for partially observable or continuous-state environments), especially with neural networks [1]. Recent advances in function approximation such as neural fitted Q iteration [15] have somewhat lessened this problem. Nevertheless, using ontogenetic RL in continuous environments remains tricky in practice. For example, an ontogenetic method such as Sarsa learns much faster than evolution on the continuous *simplerace* benchmark, but is less reliable, harder to tune, and evolution always eventually reaches higher fitness [12]. Phylogenetic approaches such as neuroevolution have been fairly successful and robust in practice on various domains [6, 18, 6, 5, 7, 8, 9], and generally do not suffer as much from function approximator fine-tuning problems as TD-based ontogenetic approaches do [23]. The reason presumably is that it is easier to find a good policy than an approximately correct value function. Note that even approximated value functions that differ only slightly from the true value function of an optimal policy may still yield very inappropriate policies, and many continuous problems have simple policies but complex value functions.

As soon as partial observability comes into play, the ontogenetic methods' difficulties get even more aggravated. Solving POMDPs using ontogenetic methods seems to be exceedingly hard and requires significant fine-tuning skill, especially when using memory-capable function approximators (e.g. recurrent neural networks).

4.1 Some conjectures

Based on the arguments advanced in this paper we conjecture the following:

- Phylogenetic methods, such as neuroevolution, generally outperform ontogenetic methods on problems with continuous state spaces and partial observability.
- Ontogenetic methods with value functions, such as Q-learning and Sarsa, are unbeatable on problems with small, discrete state spaces and full observability. For example, evolutionary methods perform worse than Q-learning and Sarsa in large discrete state spaces with full observability.
- Since ontogenetic methods, unlike phylogenetic methods, can use all experiential information obtained during interaction with the environment, ontogenetic methods outperform phylogenetic algorithms in applications where it is helpful to exploit intermediate rewards, especially if episodes are long.
- Policy gradients scale better than value-based ontogenetic learning for continuous state spaces where function approximation is necessary.
- Rank-based phylogenetic algorithms can better deal with uncertainty or noise than algorithms based on estimating true performance values or performance gradients.

One may disagree with these conjectures. If so, this underscores the need for empirical corroboration or falsification.

5 Conclusion: the need for parameterizable benchmarks

Many attempts have been made to compare different RL methods on benchmark problems. These include open competitions [20, 11] and efforts to standardize simple benchmarks through source code sharing. However, each of these efforts typically only compares a few algorithms on a single problem, leading to contradictory results regarding the merits of different RL methods.

An approach to more exhaustive reliable characterization of methods would be to create benchmarks that can be varied along as many as possible of the problem dimensions listed in section 2. Through tuning benchmark parameters, we could then corroborate, falsify or qualify hypotheses about relative method performance such as those in section 4.1. Evolutionary computation could conceivably also be used to find problem parameters that order algorithms according to a desired rank, illuminating the relative strengths of these algorithms.

References

[1] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *In Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 30–37. Morgan Kaufmann, 1995.

[2] J. Baxter and P. L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *In Proceedings 17th International Conference on Machine Learning (ICML)*, pages 41–48. Morgan Kaufmann, 2000.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.

[4] L. Buşoniu, R. Babuška, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[5] R. De Nardi, J. Togelius, O. Holland, and S. M. Lucas. Evolution of neural networks for helicopter control: Why modularity matters. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006.

[6] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(May):937–965, 2008.

[7] V. Heidrich-Meisner and C. Igel. Similarities and differences between policy gradient methods and evolution strategies. In M. Verleysen, editor, *16th European Symposium on Artificial Neural Networks (ESANN 2008)*, pages 149–154. Evere, Belgien: d-side publications, 2008.

[8] V. Heidrich-Meisner and C. Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

[9] V. Heidrich-Meisner and C. Igel. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 2009.

[10] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, 2005.

[11] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez. The WCCI 2008 simulated car racing competition. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.

[12] S. M. Lucas and J. Togelius. Point-to-point car racing: an initial study of evolution versus temporal difference learning. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 260–267, 2007.

[13] D. E. Moriarty, A. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[14] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[15] M. Riedmiller. Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, editors, *European Conference on Machine Learning (ECML)*, volume 3720 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2005.

[16] 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.

[17] J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*, pages 119–226, 2006.

[18] K. O. Stanley. *Efficient evolution of neural networks through complexification*. PhD thesis, Department of Computer Sciences, University of Texas, Austin, TX, 2004.

[19] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, 1998.

[20] The RL-competition team. RL-competition. http://www.rl-competition.org.

[21] J. Togelius, T. Schaul, J. Schmidhuber, and F. Gomez. Countering poisonous inputs with memetic neuroevolution. In *Parallel Problem Solving From Nature 10 (PPSN X)*, volume 5199 of *Lecture Notes in Computer Science*, pages 610–619. Springer, 2008.

[22] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[23] S. Whiteson, M. E. Taylor, and P. Stone. Empirical studies in action selection with reinforcement learning. *Adaptive Behavior*, 15:33–50, 2007.

[24] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *Proceedings of the 17th International Conference on Artificial Neural Networks (ICANN)*, volume 4668 of *Lecture Notes in Computer Science*, pages 697–706. Springer, 2007.

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

Contact
IDSIA
Galleria 2
6928 Manno-Lugano
Switzerland
Tel.: +41 58 666 6660
Fax: +41 58 666 6661
{julian, tom, daan, tino, juergen}@idsia.ch
christian.igel@neuroinformatik.rub.de

Julian Togelius is a researcher at IDSIA. He specializes in evolutionary reinforcement learning and computational intelligence in games. His recent research involves using neuroevolution to measure learnability of game rules in order to aid game design.

Tom Schaul is a PhD student at IDSIA, working on black-box optimization, evolution of large-scale recurrent neural networks, as well as artificial curiosity.

Daan Wierstra is a PhD student at IDSIA, working on black-box optimization and on the combination of policy gradient learning with recurrent neural networks.

Christian Igel is Juniorprofessor for Optimization of Adaptive Systems at the Institut für Neuroinformatik at the Ruhr-Universität Bochum, Germany.

Faustino Gomez is a senior researcher at IDSIA. He works primarily on evolutionary reinforcement learning using cooperative coevolution of neural networks.

Jürgen Schmidhuber is co-director of IDSIA, and professor at TU Munich and University of Lugano. His research interests include machine learning, mathematically optimal universal AI, recurrent neural networks, adaptive robotics, complexity theory, digital physics and the fine arts.