A Vision for Reinforcement Learning and its Implications for Neural Compuation

Sascha Lange and Martin Riedmiller

Albert-Ludwigs-Universität Freiburg - Dept. of Computer Science - Germany

Abstract. New methods in reinforcement learning like policy gradients and batch reinforcement learning as well as a better understanding of the strengths and limits of particular combinations of reinforcement learning and function approximation have allowed for a significant step forward, when it comes to stable and efficient learning on real systems. Due to this progress, old research questions could been answered and completely new arise. With the goal of giving some insights into where the field of reinforcement learning is heading, this paper briefly presents the history of reinforcement learning, continues with a short (subjective) discussion of the present state of the art and finally comes up with ideas for new research directions. A special focus is put on the implications of these developments to neural computation.

1 A Very Brief History of Reinforcement Learning

The early optimism surrounding reinforcement learning (RL) was severely tested, as it became clear, that the transition from learning in small discrete state spaces towards solving real problems on real systems would not be as easy as expected but would cause severe problems concerning the stability and complexity—even in rather simple problem settings. Another drawback was the insight multi-layer perceptrons and other popular function approximators do not guarantee stable convergence with common reinforcement learning update rules [1, 2], but—even worse—may lead to divergence in the general case. This was even a more severe strike as researchers had quite high hopes for these techniques as they had allowed for a number of impressive early successes in large and continuous state spaces (see e.g. [3]).

In the following years the research efforts concentrated on finding stable approximative versions of the known RL methods and on scaling them to continuous learning tasks on real systems. At the main focus were tasks from closed loop control that still were of rather limited complexity. However, in the recent past, there has been made significant progress. Thanks to new methods like policy gradients [4–6] and batch reinforcement learning [7–10] it's now possible to reliably solve many of these real-world problems directly by learning from interacting with the real system [5, 11–15]. The number of necessary interactions for learning good policies was reduced by orders from typically several million interactions to a few thousands or hundreds.

2 Once Again Recalibrating the Research Focus

This situation now allows, or even demands, the recalibration of the research focus, shifting it towards problems that have already been on the agenda in the early days of reinforcement learning, but somehow nearly fell into oblivion during the time of more pressing, more fundamental research on RL's stability and scalability. Among these questions are:

- the question about how and where to explore the environment. For maximizing the learning results, while at the same time reducing the invested effort in form of interactions, it's necessary to actively control the exploration in a goal-directed mannor.
- the question about the state and where its particular representation comes from. The design of the state space often involves important insights into the system and in many cases heavily influences the task's complexity and the final results.
- the question about how an agent could autonomously break down a given task into several smaller sub-tasks, define valid sub-goals and learn to solve them sequentially. This is one of the questions treated in hierarchical reinforcement learning—but a real break-through is still missing.
- the related but more philosophical question about the origins of reward and intrinsic motivations of an agent. How can an agent act and learn self-motivated?

3 A Chance for Rebuilding the Cognitive Agent

We strongly believe future systems have to address these questions in order to achieve further performance enhancements, to allow the application to more complex problems, and to allow an improved autonomy of (reinforcement) learning agents. Hence, it seems to be the right time to think about more complex architectures of learning agents, building upon the basic building blocks of stable and efficient RL methods, and introducing higher levels of modules for monitoring, directing and controlling the process and progress of learning. The goal would be to (re-)introduce reinforcement learning into a broader context within artificial intelligence and the cognitive agent. In this respect, we do not expect much less than a small revolution, at its end having symbolic methods and sub-symbolic methods—to which we count reinforcement learning—no longer opposed to each other, but working together hand-in-hand, solving different sub-tasks on different levels of the architecture. We clearly see reinforcement learning and methods from neural computation at the lower levels of such an architecture, having more deliberative and perhaps symbolic methods at its higher levels. From the area of neural computation, we expect important contributions in two different areas: 1. within the "core" of reinforcement learning, the approximation of value functions and the representation of policies when learning on continuous, real systems and 2. within the outlined cognitive architecture and its supporting modules.

We would like to briefly discuss three examples in the following:

Multi-layer perceptrons for approximating value functions Even in the batch approach to reinforcement learning multi-layer perceptrons and other "non-averagers" [7] do not guarantee a stable learning process in the general case [8]. Nevertheless, when using mulit-layer perceptrons, the batch approach with its separation of dynamic programming and function approximation and its synchronous updates of the approximated value function offers a significant improvement over older "direct" (online) methods with asynchronous updates. This and the benefit of using a global approximator are some of the reasons for Neural Fitted Q-Iteration (NFQ) already having scored several impressive successes on real systems, and for NFQ—besides FQI [9] and LSPI [16]—presently becoming an accepted standard method that people discuss, use and compare against [17–19]. New theoretical approaches [18] moreover promise a better understanding of the conditions and circumstances where particular non-averagers like MLPs might be save to use or better should be avoided.

Winner-take-all networks for adapting grids At the border of 1. and 2. is an application of winner-take-all networks (WTA) to the adaptation of the structure of grid approximators. All kinds of constant and interpolating grid approximators are still widely applied in RL, due to their conceptual simplicity and their provable stability within batch methods [8, 9]. WTA methods allow the automatic adaptation of the structure of such grid approximators to the distribution of the relevant data in the the state space that is actually unknown before starting the learning process [20]. This automatic adaptation is especially helpful, if the data is situated on a low-dimensional mannifold in a high-dimensional state space [20]. At the same time those methods could help bridge the gap towards higher-level modules of the cognitive agent architecture. Methods like vector quantization and especially self-organizing maps and neural gas could help detecting clusters and structure in the data and thus could allow to build a basis for a transition to deliberation and symbolic reasoning about the structure of the problem, for example in order to control exploration or to formulate new hypothesis on valid sub-goals.

Deep auto-encoders for learning feature spaces One important thing for a success of RL is to contruct a suitable state space that carries the necessary information (has the markov property), but is not overly complex. When using basic online-RL methods, this usually had to be done before starting the learning procedure, since changing the state representation afterwards had roughly the same effect as changing the structure of the function approximator (moving a support or basis function): all information learned about the value function so far was lost in the worst case [20]. But batch RL techniques allow for an immediate recalculation of the value functions in the changed state space and thus make "seamlessly" switching the state space during the learning process possible, without increasing the task's complexity in form of necessary interactions with the system [20]. Hence, the agent can deliberate over an appropriate state representation by itself and can adapt it during the learning process as thought necessary. This possibility already has been implemented in practice. The DFQ algorithm [21] uses deep auto-encoder

neural networks [22, 23] for unsupervised learning of low-dimensional feature spaces—these are used as basis for learning a value function—from high-dimensional observations (images), as observed during learning to control real systems based on visual data only [20].

One final thing to note is, from now on, all newly developed solutions have to consider the needs of real systems. If we have learned one thing from the past, it is that we can not develop fancy solutions for the discrete case and then hope these will somehow scale to stochastic problems with continuous state and action spaces. This already has failed once, even for the most simple algorithms, not only for complexity reasons, but also for the fundamental difference of problems when function approximation and stochasticity are involved. Thus, it's now time to once again rebuild the cognitive agent, but this time starting from the bottom, building on practice-proven algorithms, first addressing the fundamental questions and slowly moving towards the top.

4 Conclusion

There are many interesting challenges for neural methods in RL. We see many opportunities for such methods directly at the heart of reinforcement learning as well as in key areas of a more general, cognitive agent architecture. The newly proposed techniques for training deep neural architectures already have led to a notably increased interest in neural methods and are also of great importance in the context of RL. Winner-take-all networks can help to adapt the structure of function approximators to the distribution of the data in a very natural way and may play a role in bridging the gap to symbolic and deliberative modules in the higher levels of a cognitive agent architecture. Eventually, autonomous aerial and ground vehicles and especially autonomous robots will find entrance into our every-day live. In such systems that ultimately should be able to operate in our environment and to interact with us autonomously, modules using reinforcement learning and neural computation will almost certainly realize important aspects.

References

1. L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proc. of the 12th International Conference on Machine Learning*, pages 30–37, 1995.
2. G.J. Gordon. Chattering in SARSA (λ). Technical report, 1996.
3. G. Tesauro and T. Sejnowski. A Parallel Network that Learns to Play Backgammon. *Artificial Intelligence*, 39(3):357–390, 1989.
4. R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, pages 1057–1063, 2000.
5. A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous Inverted Helicopter Flight via Reinforcement Learning. In *Experimental Robotics IX, The 9th International Symposium on Experimental Robotics (ISER)*, pages 363–372, 2004.

6. J. Peters and S. Schaal. Policy Gradient Methods for Robotics. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.
7. G. Gordon. Stable Function Approximation in Dynamic Programming. In *Proc. of the 12th ICML*, pages 261–268, 1995.
8. D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, 2002.
9. D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(1):503–556, 2006.
10. M. Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *ECML 2005*. Springer, 2005.
11. L. Wehenkel, M. Glavic, and D. Ernst. New Developments in the Application of Automatic Learning to Power System Control. In *Proc. of the 15th Power Systems Computation Conference (PSCC05)*, 2005.
12. M. Riedmiller, M. Montemerlo, and H. Dahlkamp. Learning to Drive in 20 Minutes. In *Proc. of the FBIT 2007*, Jeju, Korea, 2007.
13. M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.
14. J. Peters and S. Schaal. Learning to Control in Operational Space. *The International Journal of Robotics Research*, 27(2):197–212, 2008.
15. M. Deisenroth, J. Peters, and C. Rasmussen. Approximate Dynamic Programming with Gaussian Processes. In *Proceedings of the 2008 American Control Conference (ACC 2008)*, pages 4480—4485, Seattle, USA, 2008. IEEE Press.
16. M. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.
17. D. Schneegaß, S. Udluft, and T. Martinetz. Improving optimality of neural rewards regression for data-efficient batch near-optimal policy identification. *Artificial Neural Networks–ICANN 2007*, pages 109–118, 2007.
18. A. Antos, R. Munos, and C. Szepesvari. Fitted Q-iteration in continuous action-space MDPs. *Advances in neural information processing systems*, 20:9–16, 2008.
19. S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 7:917, 2006.
20. S. Lange. Tiefes Reinforcement Lernen auf Basis visueller Wahrnehmungen. Dissertation, Universität Osnabrück, 2010.
21. S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain, 2010.
22. G.E. Hinton and R.R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.
23. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and Q. Montreal. Greedy Layer-Wise Training of Deep Networks. In *Proc. of the NIPS 2006*. MIT Press, 2007.