Making Reinforcement Learning Work on Real Robots

William Smart & Leslie Pack Kaelbling

Artificial Intelligence Laboratory
Massachusetts Institue Of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: Reinforcement-learning (RL) [1] techniques have elegant theoretical foundations and have proven useful in learning a variety of off-line tasks, such as playing Backgammon. However, they are still much too inefficient to be of use in the majority of robot-learning domains. In this project, we seek to make reinforcement learning effective for real robots. We are particularly interested in domains with continuous sensory inputs and continuous actions, and require that learning take place online from a relatively small amount of experience.

Motivation: In order to deploy robots in a wide variety of applications, from household to military, we must find a way to "program" them efficiently. Direct programming by humans is a tedious process, requiring a large amount of trial-and-error debugging on the part of the human. In addition, such hand-built programs are only suited for a single domain, and must be re-engineered for new houses or military situations. Thus, behavior learning must play a major role in the wide deployment of robots.

Previous Work: There has been work on RL for real robots that takes good advantage of particular properties of the application domain. Schaal and Atkeson [3] built a juggling robot that assumed continuous deterministic world dynamics. Moore developed a system that learned to control a factory packaging machine that took advantage of very slow variation in the process [1].

Approach: Our first step will be to use nearest neighbor (and locally weighted regression) [2] as a function approximator, rather than the more traditional choice of sigmoidal neural networks. There are three important reasons for this. Nearest neighbor learns from a single presentation of a data point, so there is no requirement for multiple presentations of the same data. Nearest neighbor learning is local: when the sampling distribution changes over time (as it inevitably does when gathering real experience from a robot), there is no catastrophic forgetting, as there is in typical neural network models. It is possible, with nearest neighbor, to estimate the validity of predictions based on the density of experience in the area; this can keep Q-learning from going out of control due to bad extrapolations. Preliminary experiments show that this change in function approximators is likely to make online RL much more effective.

We expect the methods of the previous paragraph to work well for learning low-level tasks in which actions do not have extremely long-term consequences. As we consider more sophisticated tasks, with longer time horizons, it will surely be the case that Q learning will not be effective in a reasonable amount of time. There are two main problems:

- Undirected exploration will lead to a random walk, which can take unacceptably long to discover interesting parts of the space.
- Model-free learning cannot benefit from hypothetical consideration of actions and requires too much trial-and-error experience in the real world.

Our second task will be to develop, implement, and test an architecture for reinforcement learning that addresses both of these difficulties. The block structure is shown in figure 1; it will work as follows:

- The user provides a policy, π_u, for performing the desired behavior. This policy can be very bad; its crucial role is to bias the robot's initial behavior enough so that the early actions of the robot will lead to interesting experience, giving bootstrapping help to the learning algorithms. This policy is a black box from the perspective of this architecture; it can be a program in any language, or even a human with a joystick.

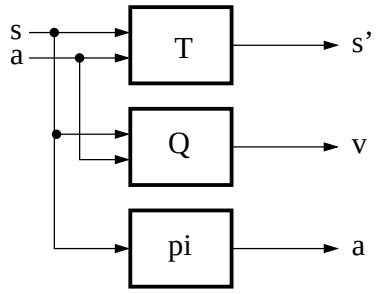

Figure 1: Learning architecture: samples from T are used to train Q; maximal values of $Q(s, a)$ are used to train π.

- In the first phase of the learning, the robot uses supervised learning techniques to train its internal policy π to choose the same actions as π_u, on the grounds that these are the best action choices we have now. Simultaneously, it performs the Q-learning algorithm above to begin estimating the action-value function, *and* uses its experience to train a world model, learning to predict the next state distribution as a function of the current state and action.
- As the world model becomes more accurate, it will be used to generate hypothetical experience, as in Dyna, for training the Q functions.
- As the Q functions become more accurate, they will be used to choose actions, and those action choices will be used to train π, overwriting and improving the originally provided policy.

We believe that the use of an initial human-provided policy will allow the robot to get much more useful experience much more quickly, and that such bad-but-useful policies will not be hard to provide. Furthermore, the deliberative use of a model will allow much more efficient "compilation" of experience into good reactions.

Results of the function approximation work and initial implementation of the learning architecture are documented in a conference paper [4].

Difficulty: There are a number of technical issues to be dealt with in the use of nearest-neighbor methods in RL. Nearest neighbor requires a distance metric on the input/output space, which may need tuning for best performance. Traditional Q learning requires a maximization over a finite set of actions. We have an infinite action space and cannot find the true maximum; we are developing strategies for choosing better actions based on a relatively small number of probes of current Q values. Indexing and editing schemes will have to be employed to keep memory from being overrun. Q learning requires a function approximator that can track a changing function; this means that older points will have to have less weight in nearest-neighbor predictions.

The more complex learning architecture still has to be worked out in all its details. Important questions include when to switch control modes, and exactly what kinds of human-provided constraint can be leveraged by the learning system.

Impact: Currently, there are no fielded robot systems that do a significant amount of online learning. The lack of learning drastically limits the deployment of robots in a wide variety of applications. Solving this problem could result in flexible, practical, adaptable robots for a wide variety of applications.

Future Work: Future work will emphasize the use of learning in very large domains, requiring hierarchical architectures and accepting human advice in a much wider range of forms.

Research Support: This work is supported by DARPA/ITO under contract number DABT 63-99-1-0012.

References:

[1] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[2] Andrew W. Moore, Christopher G. Atkeson, and Stefan A. Schaal. Locally weighted learning for control. *AI Review*, 11:75–113, 1997.

[3] Stefan Schaal and Christopher Atkeson. Robot juggling: An implementation of memory-based learning. *Control Systems Magazine*, 14, 1994.

[4] William D. Smart and Leslie Pack Kaelbling. Practical reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Palo Alto, California, 2000.