acQuire-macros: An Algorithm for Automatically Learning Macro-actions

Amy McGovern
amy@cs.umass.edu
Computer Science Department
University of Massachusetts, Amherst
Amherst, MA 01003

November 23, 1998

Abstract

We present part of a new algorithm for automatically growing macro-actions online in a reinforcement learning framework. We call this algorithm *acQuire-macros*. We present preliminary empirical results of using acQuire-macros in a simulated dynamical robot task where acQuire-macros enables the robot to discover useful macro-actions during learning.

1 Introduction

Much of the current research in reinforcement learning focuses on temporal abstraction, modularity, and hierarchy in learning. Although learning at the level of the most primitive actions allows the agent to discover the optimal policy, learning can be very slow. Temporal abstraction can enable the agent improve performance more rapidly and to use this solution to reduce the number of trials required to obtain acceptable performance. This is especially useful for time-critical applications such as robotics where the human does not have time to help the robot train over many trials.

Temporal abstraction has been studied extensively in artificial intelligence as well as in reinforcement learning. For example, Korf (1985), Laird et al. (1986), and Iba (1989) all demonstrated that problem solving systems with macro-operators can find solutions to problems that are intractable without the abstraction paradigm. Early research in reinforcement learning and hierarchy by Lin (1993), Kaelbling (1993) Dayan & Hinton (1993), and Singh (1992) demonstrates that using a hierarchical reinforcement learning system can considerably speed learning compared to a non-hierarchical system and can obtain solutions to much harder problems than can be solved using only primitive actions.

More recently, researchers have studied the theory of reinforcement learning with temporally extended actions and some have proposed new methods for learning and planning with such actions (e.g., McGovern, Sutton, & Fagg, 1997; Parr & Russell, 1997; Precup, Sutton, & Singh, 1998; Dietterich, 1998; Hauskrecht et al., 1998; Huber & Grupen, 1997; Digney, 1998; Ryan & Pendrith, 1998).

However, all of these approaches assume that the programmer provides the macro-actions to the system a priori. This paper addresses this issue by introducing part of a new algorithm called *acQuire-macros* for growing macro-actions automatically from interactions with the world. acQuire-macros is composed of several different independent components, and this paper presents the algorithm and results for only one of these components. In the following sections, we describe acQuire-macros, describe the robot simulation that is used for the experiments, and then present preliminary results.

2 Algorithm overview

acQuire-macros is based on two main ideas. The first is the algorithm that Iba (1989) introduced for growing macro-operators automatically in a traditional symbolic AI setting. His algorithm identified peaks in the evaluation function of a best-first search system and constructed macro-operators out of the actions between the peaks. acQuire-macros is based on this idea. However, instead of looking at peaks in the best-first search evaluation function, acQuire-macros looks at peaks in the temporal history of rewards. These peaks are used to form trajectories in a continuing task.

The second main idea behind acQuire-macros is to examine the state visitation frequencies across trajectories. The idea of examining state visitation histograms has also been explored in Dean & Lin (1995). In our robot task, the sensations are real-valued which means that there is no easy way to count state visitations. We solve this by using adaptive k-means clustering (MacQueen, 1967; Moody and Darken, 1989) in the space of sensations. We can then examine visitations to distinct clusters.

High-level pseudocode for acQuire-macros is given in Table 1. Instead of identifying fixed sequences of actions which are executed often, acQuire-macros tries to identify useful subgoals for new macro-actions. These subgoals are used as goal states for new macro-actions. To propose such a subgoal, when acQuire-macros identifies a peak in the history of rewards, it examines the saved clusters for frequent occurrences of a region in sensory space. If it finds such an occurrence, acQuire-macros posits a new macro-action whose subgoal is that of reaching that region. The action-values for this macro-action are updated, added to the Q-function, and learning continues as before using Macro Q-learning (McGovern, Sutton & Fagg, 1997).

Macro Q-learning is an online Q-learning method for learning macro-action values as well as primitive action-values. Each action-value for primitive actions is updated using Q-learning (Watkins, 1989) while the action-values of macro-actions are updated upon termination of the

acQuire-macros()
do forever
    given s_t, select a_t according to exploratory policy
    transition to s_{t+1}, receive reward r_t
    update Q-values using Macro Q-learning
    save (s_t, a_t, r_t)
    cluster perceptual input
    if (*check-for-peak*(r))
        sg = *find-common-subsequences*(k)
        foreach subgoal $g \in$ sg
            m = *propose-new-subgoal-macro*(g)
            if (*filter-macro*(m))
                create-new-macro(m)

Table 1: Pseudo-code for acQuire-macros

macro-action with the cumulative discounted reward received during execution of the macro-action. More precisely, after a multi-step transition from state s_t to state s_{t+n} using macro-action m, the action value $Q(s_t, m)$ is updated as follows:

$$Q(s_t, m) \quad \leftarrow \quad Q(s_t, m) + \alpha \left[r + \gamma^n \max_a Q(s_{t+n}, a) - Q(s_t, m) \right], \tag{1}$$

where the max is taken over both primitive actions and macro-actions, and

$$r \quad = \quad r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n}.$$

This is a discrete-time version of the *Semi-Markov Decision Process Q-learning* method studied by Bradtke & Duff (1996) and proven to converge by Parr (1998).

3 Experiments

3.1 The robotic simulator

We used a continuous two-dimensional simulated robot foraging task (also used by McGovern & Sutton, 1998, with different parameter settings). The circular robot starts by inhabiting a world with two rooms, one door, and one food object as shown in Figure 1. Each room is 10 feet by 10 feet with a 3 foot wide doorway. The robot is able to discern which room it is in through a sensory bit. The robot has simulated sonars to sense the distance to the nearest wall in each of five fixed directions, three forward and two backwards. The forward sonars are fixed at $0^o, 30^o$, and -30^o from the heading of the robot. The rearward sonars are at -135^o and 135^o from the robot's heading.

Simulated Environment

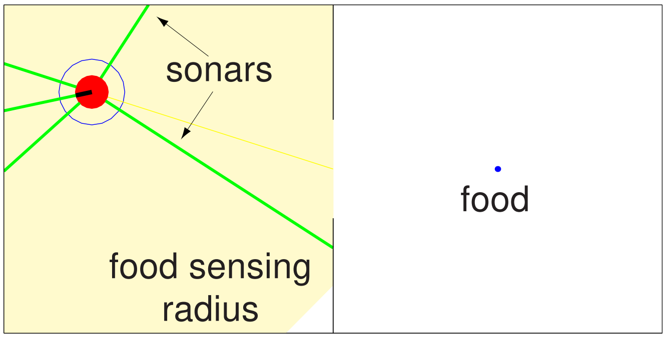

Figure 1: The simulated robotic foraging task.

The robot can also sense the direction and distance to the doorway from any point in the room, and to the food object if it is within 10 feet of the robot. When food comes within a smaller distance of the robot (2 feet), it is consumed and the robot receives a reward of +1 (otherwise the reward is zero). After the food is consumed, it re-appears in the middle of the room that the robot is *not* in. All experiments in this world use a starting position $(x, y) = (5, 1)$. The first piece of food starts in the same room as the robot. Although these experiments are all in the two room world, we plan to expand to more general n-room worlds and to other environments.

The robot uses simple inertial dynamics, with friction and inelastic collisions with the walls. There are 13 possible primitive actions. On each step, the robot can either linearly accelerate in the direction in which it is oriented (to one of 3 positive and 3 negative degrees), apply an angular acceleration (to one of 3 positive and 3 negative degrees), or apply no acceleration at all. The available discrete linear and rotational accelerations are $[-0.15, -0.1, -0.05, 0, 0.05, 0.1, 0.15]$.

The robot's equations of motion are as follows:

$$\begin{aligned} x_{t+1} &= x_t + v_t \cos\theta_t & v_{t+1} &= (1-\mu)v_t + a_t & \dot{\theta}_{t+1} &= (1-\mu)\dot{\theta}_t + \ddot{\theta}_t \\ y_{t+1} &= y_t + v_t \sin\theta_t & \theta_{t+1} &= \theta_t + \dot{\theta}_t \end{aligned}$$

where (x_t, y_t) are the global coordinates of the robot at time t, a_t is the linear acceleration, $\ddot{\theta}_t$ is the rotational acceleration, $\mu = 0.1$ is the coefficient of friction, $\theta_t \in [0, 2\pi]$ is the robot's absolute angle of orientation, v_t is the robot's linear speed, and $\dot{\theta}_t$ is the rotational velocity, or the rate of change of the robot's heading. If the robot chooses a rotational acceleration, a_t is set to zero. Likewise, if the robot accelerates linearly, $\ddot{\theta}_t$ is set to zero. The robot has a maximum linear speed and rotational velocity (0.5 and $\frac{\pi}{8}$ respectively) past which positive accelerations have no effect. The linear speed v is further constrained to be non-negative. The rotational velocity may be either positive (clockwise) or negative (counter-clockwise). The robot's position is constrained only by the walls of the world. Collisions with the walls are inelastic, which means that if a robot's new position (x_{t+1}, y_{t+1}) at time $t+1$ intersects a wall, the robot remains at position (x_t, y_t) and its

# of Tilings	Variables	Size
1	room color, door distance, door angle, forward sonar distances, linear speed, rotational velocity, food eaten in current room	$172,800$
1	all 5 sonar distances, room number, linear speed, rotational velocity, food eaten in current room	$180,000$
8	activation in each $\frac{\pi}{4}$ slice of the robot's food sensors, linear speed, rotational velocity, all 5 sonar distances	$81,000$ each

Table 2: Tile coding for the simulated robot experiments

linear speed and rotational velocity are set to zero.

Although the simulator had complete and perfect knowledge of the robot's state, the robot could only see egocentric information: the sonar readings, the food sensors, the doorway sensors, its linear speed, rotational velocity, and the number of the room it was in. Because the state space is continuous, we used a tile-coding function approximator, also known as a CMAC (Albus, 1981 and Sutton & Barto, 1998). The robot had 10 tilings over different subsets of the available information as summarized in Table 2.

3.2 Experimental results

Given the setup of the world described above, we expected acQuire-macros to posit subgoals around the doorways because each trajectory must pass through the doorway since the food changes rooms each time it is consumed. If acQuire-macros postulated such macro-actions, learning could speed up considerably. To confirm this hypothesis, we ran acQuire-macros in the simulator described above. Each new macro-action had a separate action-value function over a smaller CMAC tiling of the perceptual space as well as action-values in the main action-value function that Macro Q-learning was updating. A new macro-action started with all zero action-values and was rewarded for reaching the desired subgoal. Once the subgoal was achieved, the macro-action also terminated.

We used the square of the Euclidean distance function for adaptive k-means clustering with a distance of 5. The effective learning rate α was 0.05. We ran thirty runs of one million steps each. To examine the spatial locations of the subgoals, we looked at the inverse mapping from the sensory space of each macro-action to the spatial locations in the environment. We then imposed a tiling over (x, y) in the environment and counted how often each tile was part of a subgoal for a macro-action over the 30 runs. The left panel of Figure 2 graphs the results. As the figure demonstrates, acQuire-macros learns that the doorway area is an important location, but it also discovers macro-actions around the edges farther away from the door.

We also examined the cumulative reward over time using acQuire-macros. We partitioned the runs based on the subgoal locations discovered during each run. We partitioned the macro-actions

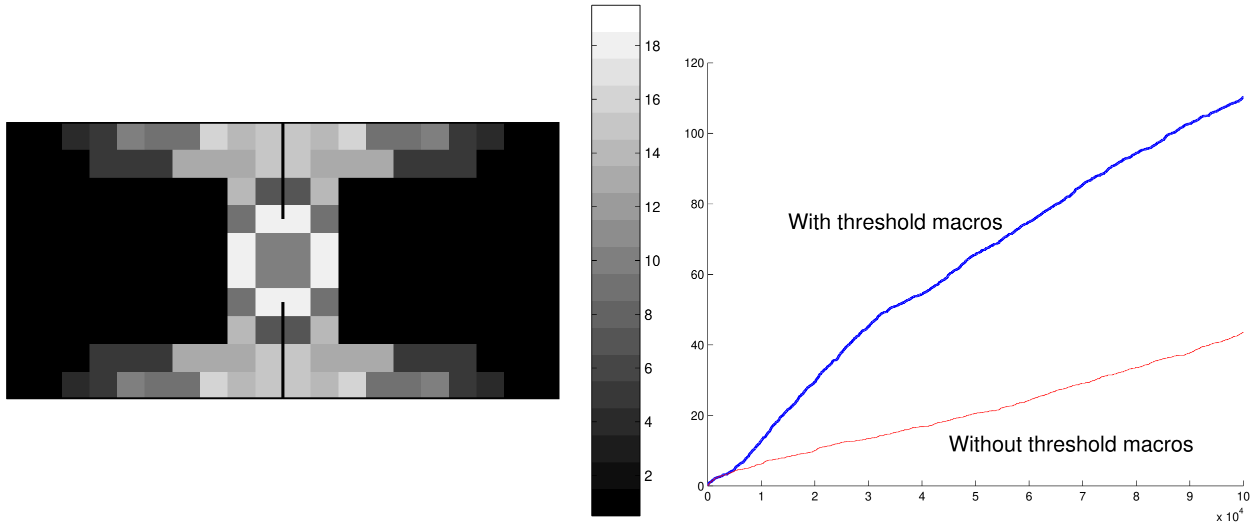

Figure 2: The left panel graphs subgoal locations of the macro-actions that acQuire-macros learns over 30 runs in the robotic simulator. The lighter colored squares were postulated as good subgoal locations more often than the darker colored squares. The right panel shows the performance of acQuire-macros both with and without macro-actions near the doorway. The thicker line represents the average performance with macro-actions near the doorway.

into two groups: those ending in subgoal locations chosen fifteen or more times, and those chosen less frequently. Cumulative reward for runs with macro-actions ending in the frequent states are averaged in the upper curve of Figure 2. The rest of the runs are averaged in the lower curve. All the curves with subgoals above the threshold had a greater cumulative reward over time than the other curves. This partitioning also corresponds exactly to runs in which subgoals were posited near the doorway. The next step is to automate this filtering process.

4 Conclusions

We have introduced part of a new algorithm called acQuire-macros which can learn macro-actions automatically. We also presented and discussed preliminary empirical evidence that acQuire-macros learns useful macro-actions. We plan to expand the results and address current limitations of acQuire-macros such as more active filtering of non-useful macro-actions.

Acknowledgments

The author wishes to thank Andrew G. Barto and Andrew H. Fagg for comments and discussions and Doina Precup for helpful comments on earlier drafts of this paper. This work was supported in part by the National Physical Science Consortium, Lockheed Martin, Advanced Technology Labs, NSF grant ECS-9511805, and NSF grant IRI-9503687.

References

Albus, J. S. (1981). *Brain, Behavior, and Robotics.* Byte Books.

Bradtke, S. J. & Duff, M. O. (1995). Reinforcement learning methods for continuous-time markov decision problems. In *Advances in Neural Information Processing Systems 7.* MIT Press.

Dayan, P. & Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5* (pp. 271–278). Morgan Kaufmann.

Dean, T. & Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. Technical Report CS-95-10, Brown University.

Dietterich, T. G. (1998). Hierarchical reinforcement learning with the MAXQ value function decomposition. In *Proceedings of the 15th International Conference on Machine Learning ICML'98.* San Mateo, CA: Morgan Kaufmann.

Digney, B. (1998). Learning hierarchical control structure for multiple tasks and changing environments. In *From animals to animats 5: The fifth conference on the Simulation of Adaptive Behavior: SAB 98.*

Hauskrecht, M., Meuleau, N., Boutilier, C., Kaelbling, L. P. & Dean, T. (1998). Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence.*

Huber, M. & Grupen, R. A. (1997). Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, *3*, 285–317.

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning ICML'93* (pp. 167–173). San Mateo, CA: Morgan Kaufmann.

Korf, R. E. (1985). Macro-operators: A weak method for learning. *Artificial Intelligence*, *26*, 35–77.

Laird, J. E., Rosenbloom, P. S. & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, *1*, 11–46.

Lin, L. J. (1993). *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, School of Computer Science.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In LeCam, L. & Neyman, J. (Eds.), *Proceedings of the 5th Berkeley Symposium on Mathematics, Statistics, and Probability* (p. 281). University of California Press.

McGovern, A. & Sutton, R. S. (1998). Roles of temporally extended actions in accelerating reinforcement learning. Technical Report 98-70, University of Massachusetts, Amherst. in preparation.

McGovern, A., Sutton, R. S. & Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *Proceedings of the 1997 Grace Hopper Celebration of Women in Computing* (pp. 13–18).

Moody, J. & Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*, 281–294.

Parr, R. (1998). *Hierarchical Control and learning for Markov decision processes.* PhD thesis, University of California at Berkeley.

Parr, R. & Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Proceedings of Advances in Neural Information Processing Systems 10.* MIT Press.

Precup, D., Sutton, R. S. & Singh, S. (1998). Theoretical results on reinforcement learning with temporally abstract behaviors. In *Proceedings of the Tenth European Conference on Machine Learning, ECML'98*. Springer-Verlag.

Ryan, M. & Pendrith, M. (1998). Rl-tops: An architecture for modularity and re-use in reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning ICML'98*. San Mateo, CA: Morgan Kaufmann.

Singh, S. P. (1992). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 202–207). Menlo Park, CA: AAAI Press/MIT Press.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning. An Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D. & Singh, S. (1998). Between MDPs and Semi-MDPs: learning, planning, and representing knowledge at multiple temporal scales. Technical Report 98-74, University of Massachusetts, Amherst.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, Cambridge University, Cambridge, England.