Using Locally Weighted Regression to Enhance Q-learning

Hal Aljibury, A. Antonio Arroyo,
Michael Nechyba
Machine Intelligence Laboratory
Dept. of Electrical Engineering
University of Florida, USA
Tel. (352) 392-6605
Email: luck@mil.ufl.edu, arroyo@mil.ufl.edu,
nechyba@mil.ufl.edu

2002 Florida conference on Recent Advances in Robotics
May 23-24 F.I.U.

Abstract:

This paper describes a two-part method of dealing with the large or continuous state spaces encountered in large reinforcement learning (RL) problems. The problem is first approached by learning the value function over a coarse quantization of states. The value function is then approximated to extend the results to the original large or continuous state space. This second step improves the problem performance by allowing the agent to use all of the originally available state information.

Performance of this method is significantly improved compared to the original performance of the RL algorithm. The nature of this method makes it directly applicable to practical reinforcement learning problems where perceptual aliasing or interminable learning times are a problem. A simulator is used to demonstrate this method on the task of obstacle avoidance in a complicated environment in which a moderate amount of perceptual aliasing is present. Significant performance enhancements are observed, as well as a learned policy that is not dependent on state history.

Introduction:

Oftentimes, the problem faced by researchers applying reinforcement learning to a nontrivial robotics problem is that they run head-on into the curse of dimensionality. This is a particular problem for those researchers using discrete-state algorithms, as the number of states exponentially increases with the complexity of the problem. Various methods of dealing with large state spaces have been proposed and experimentally demonstrated, but none of these methods have effectively dealt with practical situations where the difference between states is sometimes not perceivable by the robot. It can be shown that many useful reinforcement learning algorithms experience theoretical difficulties under these conditions, and that other algorithms which explicitly account for incomplete perception either lack generality, or are too computationally awkward for effective use.

The method introduced here for dealing with this problem is a two-step procedure. First the problems' value function is learned over a small number of states (here accomplished with Q-learning). The second step involves approximating the value function so that it can generalize from those discrete states to a continuous state space, allowing previously discarded state information to be used effectively in the problem solution.

Background and theoretical underpinning:

An abridged version of the problem background is presented here. Readers wishing a more thorough background are encouraged to refer to [1].

Reinforcement learning describes situations where an agent is presented with a problem along with no information about how to solve it other than a set of rewards and possible actions. For example, in the obstacle avoidance problem, the agent knows nothing about the situation other than that it gets rewards for moving and not hitting anything.

Given this, the agent can develop a policy for choosing a path through the obstacles through trial and error. This policy will be based

entirely on the robots current sensory input (i.e. the state S), and a memory of what reinforcements were received upon previous visits to this state. Watkins [2] derived an algorithm, named Q-learning, that iteratively improved the agents estimate of a state's value.

Q-Learning Algorithm:[1]

- Initialize Q(s,a) arbitrarily for all states and actions.
- Set s to some initial start state

Repeat:

- Select action a using the policy derived from current Q(s,a) function.
- Take action a and observe the reward r and next state s'.
- Update Q(s,a) based on the step just taken.
- Set the state s to the new state s'

The quantity $Q^{\pi}(x_i,a)$ is the predicted value of any particular state/action pair. Selecting the $\max_{a \in A} Q(x,a)$ over all states x will result in the ideal value function $V^{\pi}(x)$. Of course, since the method begins with an imperfect estimate of the Q-values of each state, an iterative evaluation and prediction update algorithm must be applied. It is important to note that this equation allows rewards received to propagate backwards upon multiple traversals of the same state, action sequence.

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma[\max_{a'} Q(s',a')] - Q(s,a))$$

The above algorithm has been proven to converge with probability 1 to the true Q-value Q^{π}, given that the problem is a Markov Decision Process, all states are visited infinitely often, and that the Q-values are represented as discrete values in a look-up table

Q-learning is the most computationally tractable bucket-brigade type reinforcement learning algorithm, yet it has a basic problem: it doesn't scale up very well. For example, consider a robot with two infrared sensors each capable of 8-bit precision, plus being able to turn in any of 360 different directions and move at 8 different speeds. In order to apply Q-learning, we need to assume that the inputs will completely distinguish states, and that no hidden states remain in the problem. The set of states for this problem would be every possible combination of sensor values times the number of possible actions, or $2^8 * 2^8 * .7*2^9 * 2^3 = .7*2^{28}$. Obtaining a value function covering every state in this problem's Q-table would be quite impossible. Even if all the state values could be stored (somewhat reasonable, roughly 200 megs), the sheer size of the problem would inhibit convergence, due to long sequences of actions needing many passes to propagate their values back to the start of the sequence. In fact, according to bounds established by Kearns & Singh [3], it would take nearly a trillion steps to approach within 1% of the true Q-values.

Clearly, this dimensionality problem discourages the use of Q-learning for practical real-world problems. Another problem arises when an agents sensory inputs are mapped to Q-table states. The problem is that the same sensory input can map to two different situations (perceptual aliasing). Illustrations of this are shown in figures 1-3. Figures 2 and 3 also show the problem of action selection in an environment where two different situations map to the same state.

Reinforcement learning work has focused primarily on solving these problems. Good progress has been made on reducing the dimensionality factor, but the problem with perceptual aliasing has proven to be only marginally tractable. These type of problems are so hairy that when researchers are faced with one, they quite often pretend that their situation is 'close enough' to being fully observable, and use methods developed only for the simpler type problems, hoping that the performance of these algorithms would smoothly degrade as the amount of perceptual aliasing increased. Unfortunately, this approach has been shown to be theoretically unsound, as Gordon [4] and other researchers have demonstrated that even so much as one perceptually aliased state can cause divergence.

The obvious remedy to this difficulty is to increase the number of states enough so that perceptual aliasing is no longer a problem. That returns us to the dimensionality problem, however.

The approaches to reducing the dimensionality hang-up take one of two general strategies. The first strategy updates multiple Q-values at each step, reducing the number of total

---

[1] Definitions:
s – a member of a set of states [S]
a – a member of a set of actions [A]
– discount factor
– learning rate
$Q^*(s,a)$ – the policy which gives the optimal performance.

steps needed [5,6]. The other strategy is to dispense with discrete states altogether and use a continuous state-space. As a solution to the perceptual aliasing problem, the continuous state-space approach has more applicability than the discrete-state approach, but it requires different methods of storing the experiences learned by the algorithm.

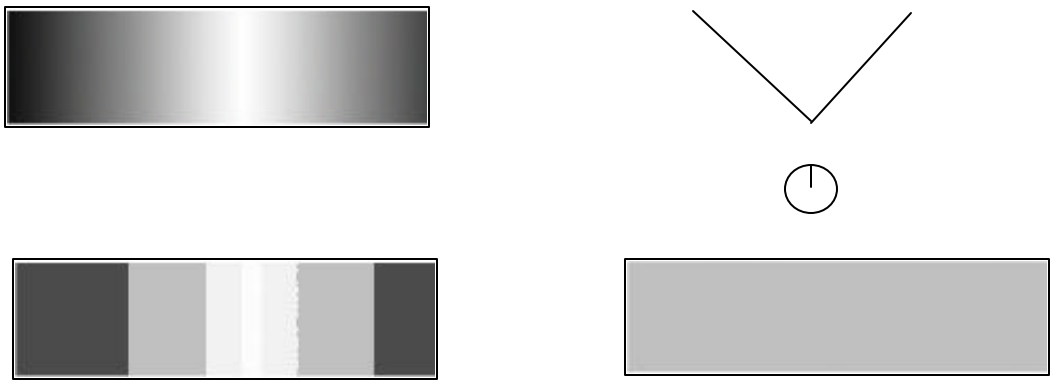

Figure 1. Proceeding counterclockwise from the upper right, we see the situation of a robot approaching a corner. In the diagram at upper left, we see how a robot would perceive the corner with ideal sensors. In the diagram just below that, we see how the robot would observe the corner, when forced to quantize the inputs into 4 levels. Then at the lower right, we have gone one step further and reduced the number of sensors from many to only 2, where each sensor gives the average reading over its detection area. It is now unclear what, if anything, lies ahead of the robot.

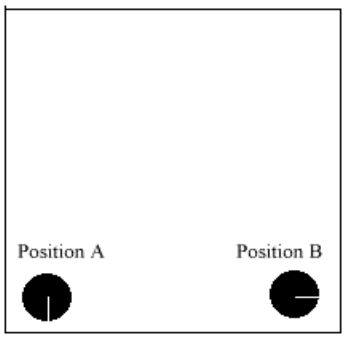

Figure 2. Same inputs, same action

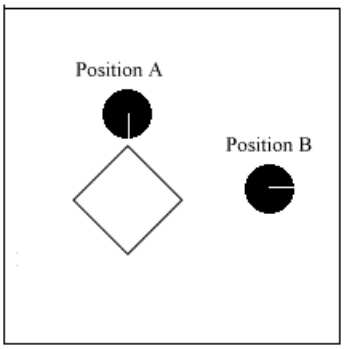

Figure 3. Same inputs, different action

The method used in our approach is to take a discrete-state method and use it to generate a continuous-valued approximation of the value function. Ideally, the approximation would be fed back into the learning process, but there is no theoretical proof that it will work[2].

[2] According to the reasoning in [7], it can't be proven to converge. However, it might work most of the time in practice.

[7]. What we can do however, is to use the discrete-state method to generate a value function, and then, once the learning is complete, use a continuous-valued approximation for control.

The method of approximation used, locally weighted regression (LWR), is well suited for generating a smoothed line from data points having a non-uniform distribution [8,9,10,11]. Other methods, and neural networks in particular, are not expected to function well when approximating a Q-function which varies so widely with comparatively few data points.

LWR has been successfully used in a variety of RL applications [11,12,13,14]. In general, LWR is used to create a local model of a function around a query point. The particular advantage of using only a local model is that

only simple linear functions are required to approximate the global function in the neighborhood of the query point. This avoids the often difficult task of generating a global model of a nonlinear function.

Experimentation:

A speedy robot simulation [1] was used to evaluate whether using LWR to approximate the value function would provide any improvement in the task of obstacle avoidance. The simulated robot arena can be seen in figures 4 and 5.

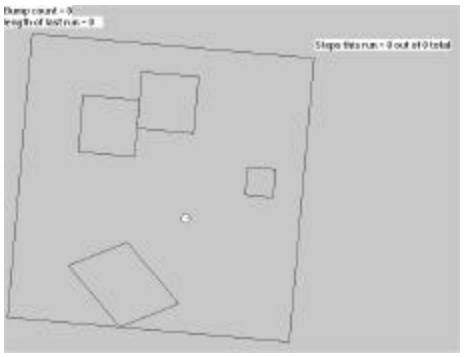

Figure 4. The main robot arena.

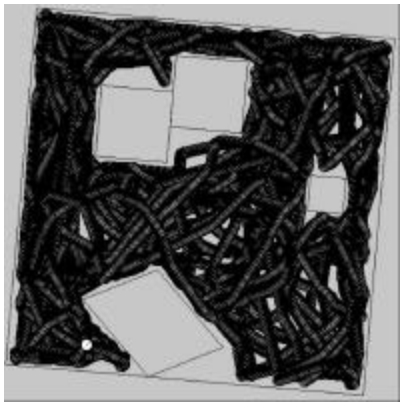

Figure 5. The area covered by the simulated robot over 300,000 steps. Note the unpopular areas near the upper left and bottom center.

The first half of the procedure was to obtain a Q-table generating an acceptable behavior. This behavior would have the robot covering all of the reachable area in the arena, while colliding as little as possible (e.g. the behavior seen in figure 5). This specification rules out otherwise ideal collision-avoidance behaviors such as spinning in place, advance-retreat, etc. Determining the required reinforcement schedule involved tweaking the rewards until the desired behavior emerged. The reward values used were –20 for a collision, +5 for a forward step, +2 for a turn, and +1 for backing up. The other Q-learning factors were learning rate alpha = 0.4, and discount factor gamma = 0.7. Also, to ensure sufficient exploration of states, a random move was picked 10% of the time, while the greedy (maximum value) selection was the default selection.

Once a reinforcement schedule was found that generated an acceptable behavior, the simulator was run for as long as reasonably possible to generate a Q-table that had approached convergence. Over a 10-day run, the simulator performed 650 million updates (on a PII 266), saving the Q-table every 50,000 steps. When the graph of the absolute value sum of the Q-table states was plotted, it shows that the values had come reasonably close to convergence after only 100 million steps.

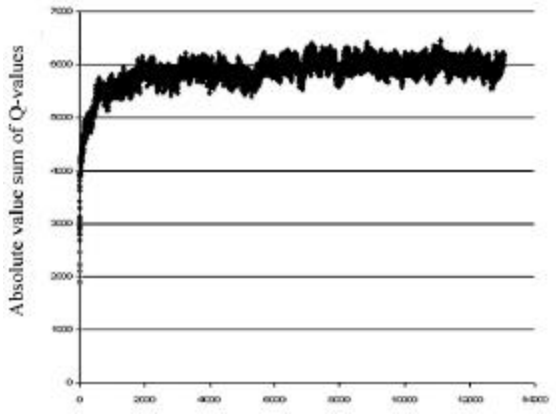

Figure 6. Graph of Q-value convergence, showing the absolute Q-value sum over all states.

Because of the chattering of the Q-values, the Q-tables of the 10 last saves were averaged together to obtain a representative Q-table.[3]

Once the Q-table had been generated, last task was to create the value function approximation and use it for controlling the robot. Because of the paucity of the sample points and the roughness of the value function it was unnecessary to include values distant from the query point in the regression calculation. Instead, only the nearest 10 sample points were used, and various distance weighting formulas were tested for performance, showing that the best weighting formulas were centered around 1/n^2.

[3] For a complete description of chattering, the reader is referred to [13]

The experimental runs were evaluated on a 'time until task failure' basis, i.e., by measuring the number of steps to the first collision. The weighting functions were all of the form 1/distance ^2, where n could be any positive real number. Values of n greater than 3 increasingly reproduced the quantized Q-values of the original table, and thus the identical action selection. Values of n less than one resulted in rapid performance decrease, due to oversmoothing.

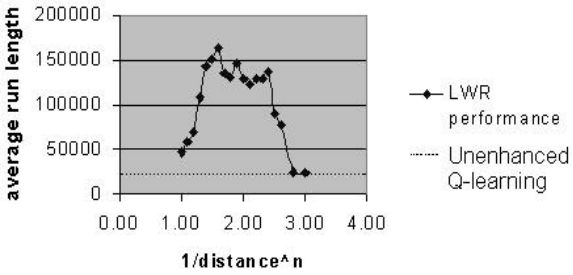

Figure 7. Performance of LWR compared to standard Q-learning for the arena shown in Figure 4.

Once experimentation had showed that the LWR improves the collision-avoidance performance for a specific environment, the next step was to examine whether the learned value function was specific to that environment. Such would be the case for RL algorithms that use a state-sequence memory to distinguish states that would otherwise appear identical. If the Q-learning has learnt a good memoryless policy, then this policy, though probably not perfectly adapted for other environments, will still usefully serve as a controller without having to re-do the computationally expensive Q-learning step.

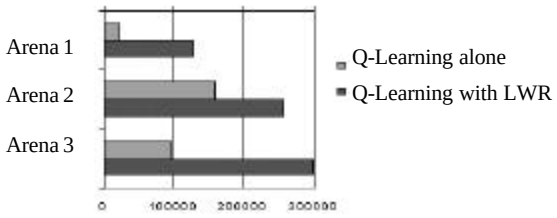

Figure 8. A comparison of the collision avoidance performance in different environments. Arena 1 is the original arena shown in figure 4, arenas 2 and 3 are shown in the appendix. In all three environments the use of LWR significantly enhanced the performance.

As can be seen in figure 8, the controller performed quite well in other arenas in comparison to the original. In fact, it actually performed better. The explanation behind this boils down to is that the original environment presented more of a challenge with hard-to-detect obstacle corners. See the appendix for figures of collision locations within the original and alternate arenas.

Discussion:

Figure 9 shows a simplified interpretation of how LRW smoothes the values along the boundary between two states B and C. As shown in figure 9 by the curved lines, the LWR approximation can affect the action selection (on the basis of highest action value) near the boundaries of the Q-table states.

When the collision points of all three arenas are compared, it can be seen that all collisions were against corners, with more acutely angled corners having more collisions, and obtusely angled corners having much fewer collisions. The self-evident statement that could therefore be made is that any change in action selection which resulted in improved performance would result in fewer collisions against corners.

Using figure 9 as an analogy, let us say that State B represents the condition of a robot nearing a corner. The robot has no idea that it is getting dangerously close to a corner, as the inputs which it receives mimic the condition of being a safe distance from a perpendicular wall. The state-action values learned for State C represent the composite conditions of the robot colliding with a corner, as well as the robot coming somewhat close to a perpendicular wall. In States B and C, Action 1 would be to go forward, and Action 2 would be to turn away.

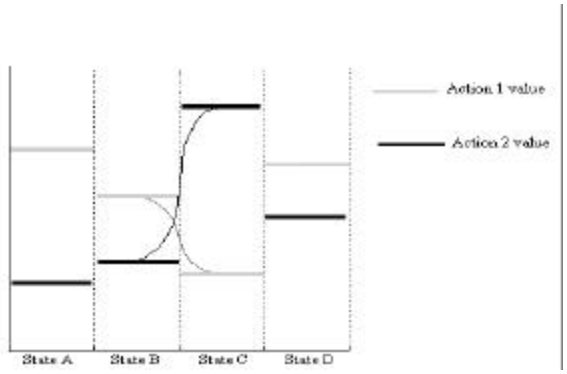

Figure 9. Boundary smoothing

When a discrete-state action selection method is used in this example, and the inputs are in a range belonging to State B, it can be seen that action 1 is selected. By the time the input values have risen to the threshold of State C, it is too late to turn away—a collision has already occurred. However, if LWR had been used to smooth the values along the boundary between the states, the robot would have "known" it was approaching an undesirable state as the input values neared the state boundary, and turned away before a collision occurred.

Since the value function was presumed to be a well-behaved function (i.e., no discontinuities, etc..) smoothing the state boundaries would reproduce the value function more exactly. Unfortunately, there was no a priori method to determine how much smoothing was too much, so it seemed wisest to start with little smoothing (by a large distance-weighting factor), and then slowly decrease the weighting factor until the collision-avoidance performance began to decline again—which implied that there was now too much smoothing.

Conclusion:

This paper demonstrated the application of Q-learning together with LWR to an obstacle-avoidance problem. The significance of this method lies in that this is a procedure which allows a continuous-valued problem to be approached as a discrete-state problem in order to minimize the computational complexity without sacrificing all of the possible accuracy obtainable with additional state information. The benefit from this method is straightforward – it becomes possible to attempt the characterization of complex continuous-state problems that were previously immune to everything but theoretical massage.

References:

[1] H. Aljibury, *Improving the Performance of Q-learning with Locally Weighted Regression.* Masters Thesis, University of Florida, 2001.

[2] C. J. C. H. Watkins, *Learning from Delayed Rewards.* Ph.D. Thesis, King's College, Cambridge, 1989.

[3] M. Kearns and S. Singh, "Finite-sample Convergence Rates for Q-learning and Indirect Algorithms," in *Advances in Neural Information Processing Systems 12*, M. Kearns, S. A. Solla, and D. Cohn, Eds., Cambridge, MA: MIT Press, 1999, pp 996-1002.

[4] G. J. Gordon, "Chattering in SARSA()," CMU Learning Lab, Pittsburgh, PA, Internal Report, 1996.

[5] C. Szepesvári and M. L. Littman, "A Unified Analysis of Value-function-based Reinforcement-learning Algorithms," *Neural Computation*, 11:8, pp. 2017-2059, 1999.

[6] S. Singh, T. Jaakkola, and M. I. Jordan, "Reinforcement Learning with Soft State Aggregation," *Advances in Neural Information Processing Systems*, G. Tesauro and D. Touretzky, Eds., 1995, vol. 7, pp 361-368.

[7] J. N. Tsitsiklis and B. Van Roy, "An Analysis of Temporal-difference Learning with Function Approximation," M.I.T., Cambridge, MA, Tech. Rep. LIDS-P-2322, 1996.

[8] Cleveland, W.S. and Loader, C. "Smoothing by Local Regression: Principles and Methods" (with discussion). in *Statistical Theory and Computational Aspects of Smoothing*, W. Hardle and M.G. Schimek, Eds. Heidelberg: Physica-Verlag, 1996, pp. 10-49; pp. 80-102; pp. 113-120.

[9] T. Hastie and C. Loader, "Local Regression: Automatic Kernel Carpentry," *Statistical Science*, vol. 8, pp.120-143, 1993.

[10] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*, 11:(1-5), pp. 11-73, 1997.

[11] J. Forbes and D. Andre, "Practical Reinforcement Learning in Continuous Domains," Computer Science Division,

University of California, Berkeley, Tech. Rep. UCB/CSD-00-1109, 2000.

[12] D. Nikovski, "Visual Memory-based Learning for Mobile Robot Navigation," in *Proceedings of the Second International Conference on Computational Intelligence and Neurosciences*, 1997, vol. 2, pp.1-4.

[13] P. Tadepalli and D. Ok, "Scaling up Average Reward Reinforcement Learning by Approximating the Domain Models and the Value Function," in *Proc. 13th Int. Conf. on Machine Learning*, 1996, pp. 471-479.

[14] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning for Control," *Artificial Intelligence Review Special Issue on Lazy Learning Algorithms*, 11:75-113, 1997.

Appendix:

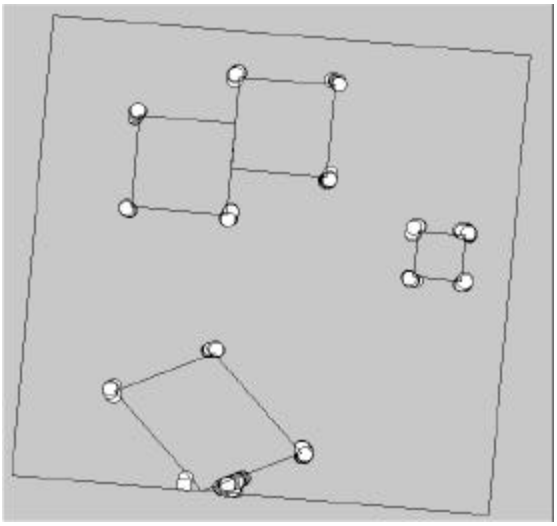

Figure 10. Collisions in the first arena

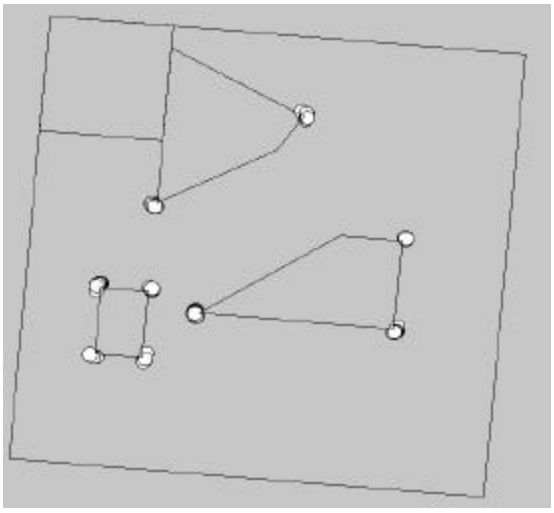

Figure 11. Collisions in the second arena.

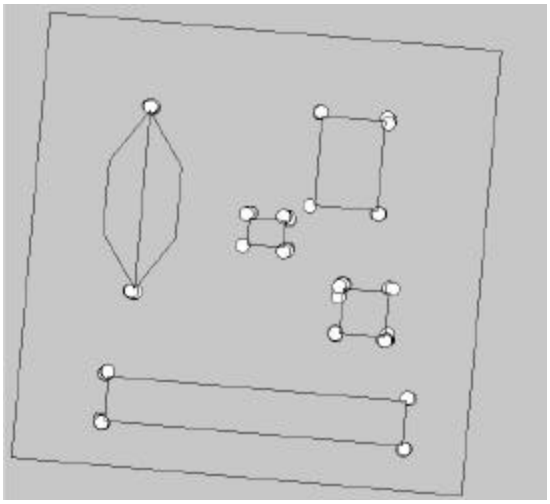

Figure 12. Collisions in the third arena.