reinforcement-learning-exercises

March 26, 2019

1 21. Reinforcement Learning

21.1 Implement a passive learning agent in a simple environment, such as the 4×3 world. For the case of an initially unknown environment model, compare the learning performance of the direct utility estimation, TD, and ADP algorithms. Do the comparison for the optimal policy and for several random policies. For which do the utility estimates converge faster? What happens when the size of the environment is increased? (Try environments with and without obstacles.)

21.2 Chapter Section **??** defined a **proper policy** for an MDP as one that is guaranteed to reach a terminal state. Show that it is possible for a passive ADP agent to learn a transition model for which its policy π is improper even if π is proper for the true MDP; with such models, the POLICY-EVALUATION step may fail if $\gamma = 1$. Show that this problem cannot arise if POLICY-EVALUATION is applied to the learned model only at the end of a trial.

21.3 [prioritized-sweeping-exercise]Starting with the passive ADP agent, modify it to use an approximate ADP algorithm as discussed in the text. Do this in two steps:

1. Implement a priority queue for adjustments to the utility estimates. Whenever a state is adjusted, all of its predecessors also become candidates for adjustment and should be added to the queue. The queue is initialized with the state from which the most recent transition took place. Allow only a fixed number of adjustments.

2. Experiment with various heuristics for ordering the priority queue, examining their effect on learning rates and computation time.

21.4 The direct utility estimation method in Section Section **??** uses distinguished terminal states to indicate the end of a trial. How could it be modified for environments with discounted rewards and no terminal states?

21.5 Write out the parameter update equations for TD learning with

$$\hat{U}(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g)^2 + (y - y_g)^2} \, .$$

21.6 Adapt the vacuum world (Chapter Section **??**) for reinforcement learning by including rewards for squares being clean. Make the world observable by providing suitable percepts. Now experiment with different reinforcement learning agents. Is function approximation necessary for success? What sort of approximator works for this application?

21.7 [approx-LMS-exercise]Implement an exploring reinforcement learning agent that uses direct utility estimation. Make two versions—one with a tabular representation and one using the function approximator in Equation (Section **??**). Compare their performance in three environments:

1. The 4×3 world described in the chapter.

2. A 10×10 world with no obstacles and a +1 reward at (10,10).

3. A 10×10 world with no obstacles and a +1 reward at (5,5).

21.8 Devise suitable features for reinforcement learning in stochastic grid worlds (generalizations of the 4×3 world) that contain multiple obstacles and multiple terminal states with rewards of $+1$ or -1.

21.9 Extend the standard game-playing environment (Chapter Section **??**) to incorporate a reward signal. Put two reinforcement learning agents into the environment (they may, of course, share the agent program) and have them play against each other. Apply the generalized TD update rule (Equation (Section **??**)) to update the evaluation function. You might wish to start with a simple linear weighted evaluation function and a simple game, such as tic-tac-toe.

21.10 [10x10-exercise] Compute the true utility function and the best linear approximation in x and y (as in Equation (Section **??**)) for the following environments:

1. A 10×10 world with a single $+1$ terminal state at (10,10).

2. As in (a), but add a -1 terminal state at (10,1).

3. As in (b), but add obstacles in 10 randomly selected squares.

4. As in (b), but place a wall stretching from (5,2) to (5,9).

5. As in (a), but with the terminal state at (5,5).

The actions are deterministic moves in the four directions. In each case, compare the results using three-dimensional plots. For each environment, propose additional features (besides x and y) that would improve the approximation and show the results.

21.11 Implement the REINFORCE and PEGASUS algorithms and apply them to the 4×3 world, using a policy family of your own choosing. Comment on the results.

21.12 Investigate the application of reinforcement learning ideas to the modeling of human and animal behavior.

21.13 Is reinforcement learning an appropriate abstract model for evolution? What connection exists, if any, between hardwired reward signals and evolutionary fitness?