2nd Lt David Crow

ENG/20M

Assignment #4 – Gridworld

Artificial Intelligence – CSCE 523

**Execution Instructions**

To run the program, simply compile “header.h,” “functions.cpp,” and “main.cpp” with gcc. Run the executable and answer the prompts. Results can be found in “policies.txt,” “tdZero.csv,” and “tdLambda.csv.”

**Description of Solution**

As expected, I implemented Value Iteration, TD()/SARSA, and TD() to solve this gridworld domain. When training my system, I use the following parameters:

* + This encourages the agent to explore the world because, regardless of world size, the reward remains worthwhile. I also experimented with a goal reward of , but the results were not as promising.
  + Other experiments used rewards of or , but the reward did not scale to large boards.
  + I first attempted to use a constant number of iterations to train the policy (e.g., , , , or ), but each value proved inadequate for at least one world size.
  + In letting the number of episodes equal the number of cells, we ensure that the length of our training scales with the world size. We also increase the likelihood of starting in (and thus training from) every possible cell.
  + I experimented with smaller error values (e.g., and ), but the increase in performance was not worth the increase in execution time. This value seems perfectly adequate.
  + Because we don’t expect values to change significantly from one episode to the next, we can use a moderately high discount factor without negatively affecting our results. In my tests, I found that this factor often gives better results than does higher values. I imagine this is because we still want to (sometimes) allow for drastic swings in value, and a higher discount factor places too much faith in previous episodes.
  + I remember Dr. Peterson mentioning in class that would likely be sufficient. In my own testing, I found that worked a bit better. I’m not confident in the reason behind this, but I think it’s because we need to allow the agent ample time to explore the space.
  + I’m not terribly sure of how this parameter affects training. However, I know that , so I figured that would work. It did.

**Results**

At the bottom of this report, I compare the rate of convergence of TD() and TD() for four different gridworld instances. In all cases, we see that TD() converges to the optimal value faster than does TD(). In three out of the four test cases, however, we see that TD() initially gives worse results. I imagine we can attribute the fourth case’s dissimilar behavior to the specific gridworld instance in question.

Because actions in this domain are stochastic, we cannot guarantee that an agent will follow the optimal path, and we also cannot guarantee that it won’t get stuck in loops. The various outliers in the four plots are examples of such suboptimal behavior.

For the reader’s benefit, gridworld instances, final policies for each gridworld, and exact evaluation results can be found in the “GridWorld/Evaluation/” directory.

**Experiences**

I enjoyed this return to pure programming (as opposed to Homework 3, in which we used Otter and Graphplan). Additionally, I appreciate being able to program in my language of choice.

I have some eprevious xperience with implementing a gridworld solver, but we used Q-learning and were not required to evaluate the policy. Additionally, we solved much smaller world instances and did not utilize an epsilon-greedy policy. Overall, multiple small changes – like those just mentioned – ensured that this assignment was still unique and nontrivial.

Figure 1: Rate of Convergence for TD() and TD() for Gridworld of Size

Figure 2: Rate of Convergence for TD() and TD() for Gridworld of Size

Figure 3: Rate of Convergence for TD() and TD() for Gridworld of Size

Figure 4: Rate of Convergence for TD() and TD() for Gridworld of Size