A5 - Monte Carlo Probabilistic Agent

CS 4300
Fall 2016
Leland Stenquist - 2,4,6
Ryan Keepers - 1,3,5

1. Introduction

The Monte Carlo agent is a method of selecting decisions for the wumpus world agent based on probabilistic knowledge. Each action the agent takes in response to a percept is determined by the likelihood of the placement of objects (pits and wumpuses) on the board. Those likelihoods are obtained by sampling numerous similar, randomly generated boards.

The most fundamental question is, of course, how successfully the agent will perform when navigating by Monte Carlo? After that, the questions all pertain to the parameters and efficiency of Monte Carlo itself. How many sample boards need to be generated for Monte Carlo to approximately reach its peak success? Can Monte Carlo be made more efficient? And will improved efficiency have significant trade-offs in terms of success?

2. Method

The main purpose of this assignment was to implement the function CS4300_WP_estimates on and agent CS4300_MC_Agent. This agent would use the Monte Carlo method to solve the Wumpus World. The results of this agent's success and scores would then be compared to a different agent that used set probability for Wumpus and pits. Namely if it detects a stench, any adjacent cell has 1/15 probability of a Wumpus. If it detects a breeze any adjacent cell has a 0.20 probability of a pit.

For each round the agent receives a percept. It tries to detect glitter, if so, it grabs and plans a route home. It then checks for a scream and updates accordingly. The next thing the Wumpus wants to do is find a safe cell in the frontier. If it can it will use A star to plan a path to that cell. If it can't find a safe cell it will try to shoot the wumpus. If it does not have an arrow it will venture into a possibly dangerous cell. This is where Monte Carlo comes in.

Monte Carlo is probabilistic way of trying to determine the probability of danger in different cells. Using the information given, by previous percepts, the agent generates N boards. Once N appropriate boards have been generated averages are taken on the locations of pits and Wumpus to try to determine the probability that a cell is lethal. The agent can then use that information to make the best choice on what cell to visit.

The function CS3400_MC_updates is what actually uses Monte Carlo. Because Monte Carlo is not efficient, CS4300 should be called as little as possible in order optimize the agent. In our version of the MC_agent Monte Carlo is only called when any percept comes in that does not include glitter. We also use an optimization on CS4300_MC_updates which is to generate the board for Wumpus and pits separately. We found this to be much faster and on 1000 trials it generated statistically equivalent results to generating the board for Wumpus and pits together.

3. Verification of Program

CS4300_MC_agent verification:
The Monte Carlo agent is verified through a test program called CS4300_WW_tester. This tester runs the agent through four basic board scenarios. The first board has a gold, but no wumpus or pits, and it is always expected to succeed. The second board has a gold surrounded by pits, and is always expected to fail. The third board has no pits, but the wumpus is standing on the gold; it is always expected to succeed. The fourth board is solvable, but depends on the probability outcome. It is expected that the agent will succeed at least once out of 100 attempts.

CS4300_WP_estimates verification:

Verification of matching a simple board.

Test board	Breezes:	Stenches:	Visited:
0 0 0 0 0 0 0 0 0 4 1 0 0 0 0 0	0 0 0 0 0 0 1 0 0 -1 -1 1 0 0 1 0	0 0 0 0 0 1 0 0 1 -1 -1 0 0 1 0 0	1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1

Probability results

Matches	Pit Probability	Wumpus Probability
10	0 0 0 0 0 0 0 0 0 0 1.0 0 0 0 0 0.1	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
200	0 0 0 0 0 0 0 0 0 0 1.0 0 0 0 0 0.2	0 0 0 0 0 0 0 0 0 0.93 0 0 0 0 0 0

Verification of 10 matching complicated boards.

Test board	Breezes:	Stenches:	Visited:
1 0 0 2 0 0 0 0 0 0 1 3 0 0 0 0	-1 -1 -1 -1 1 0 1 0 0 1 0 -1 0 0 1 -1	-1 -1 -1 -1 0 0 0 1 0 0 -1 -1 0 0 0 -1	0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 0

Boards matching Breezes

1 0 0 2 0 0 0 0 0 0 1 3 0 0 0 0	1 0 0 0 3 0 0 1 0 0 1 0 0 2 0 0	1 0 0 0 0 2 0 0 0 0 1 0 0 3 0 0	0 0 0 3 0 1 0 0 0 0 2 0 0 0 0 1	0 0 1 2 0 1 0 0 0 3 1 0 0 0 0 0
2 0 0 0 0 1 0 0 0 0 0 0 0 3 0 1	1 4 0 0 0 0 0 0 0 0 1 0 0 0 0 0	0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 4	1 0 0 0 0 0 0 1 2 0 1 0 0 0 3 1	0 0 0 2 0 1 0 0 0 0 1 0 0 0 0 3

Boards matching Stenches

1 0 0 4 0 0 1 0 1 0 1 0 0 1 1 1	1 1 0 3 0 0 0 0 0 1 0 0 0 1 2 0	2 1 1 0 0 0 1 0 0 0 0 3 0 0 1 1	0 1 0 0 0 2 0 0 0 0 0 3 0 0 0 1	0 0 1 3 0 1 0 1 0 2 1 0 0 1 0 1
0 1 1 3 0 1 2 1 0 1 0 0 0 0 0 0	0 0 0 0 1 1 1 0 0 0 0 3 0 2 1 0	0 0 1 3 0 0 1 0 0 1 1 2 0 1 0 0	0 2 0 3 1 0 1 0 0 0 1 0 0 0 1 0	0 0 1 3 0 0 1 0 1 0 0 0 0 0 0 2

Probability results

Matches	Pit Probability	Wumpus Probability
10	.5 0 .1 0 0 .5 0 .2 0 0 .8 0 0 0 0 .3	0 0 0 .6 0 0 0 0 0 0 0 .3 0 0 0 0
200	.44 0 .2 0	0 0 0 .49

	0 .66 0 .19 0 0 .7 0 0 0 0 .47	0 0 0 0 0 0 0 .44 0 0 0 0

4. Data and Analysis

Monte Carlo vs Set Probability

Our data and analysis compares the Monte Carlo method versus another method where we simply assign the danger of any unsafe board to 1/15 for wumpus and 0.20 for pit. Monte Carlo definitely performed well.

	Mean Score	**Mean Success(%)**	**Mean Time(secs)**
No MC	10.670	54.62	0.0199
50 trials	360.86	70.4	0.2954
100 trials	385.86	71.6	0.629
200 trials	416.8	73.2	1.1194

Split MC vs Standard MC

In trying to optimize the original specs for the assignment (1000, 5000, 10000) we found an optimization that we tried to prove. The reason we did not mention this in class is that we had not been able to run sufficient stats to say if it worked or not. This optimization was to validate the wumpus board and the pits board in CS4300_WC_estimates separately so that the generated boards did not have to be quite so specific. We wanted to share the results of this strategy. We call this strategy Split MC. when we validate the board taking in account wumpus and pits together we call this Standard MC.

	Standard MC	**Split MC**
1000 Trials		
Mean score	354.176	415.764

Mean success (%)	70	73.2
Mean time (seconds)	48.41	5.3995
10 Trials		
Mean score	343.6	343.4880
Mean success (%)	69.6	69.6
Mean time (seconds)	0.58	0.0834
50 Trials		
Mean score	297.14	360.86
Mean success (%)	67.2	70.4
Mean time (seconds)	2.61	0.2954
100 Trials		
Mean score	385.816	385.86
Mean success (%)	71.6	71.6
Mean time (seconds)	2.6835	0.6295
200 Trials		
Mean score	383.55	416.7960
Mean success (%)	71.6	73.2
Mean time (seconds)	10.38	1.1194

Average of 50 repetitions on the first 50 boards.

	Mean Score	**Mean Success(%)**	**Mean Time(secs)**
10 trials	345.87	69.6	0.1796
25 trials	343.04	69.5	0.4067
50 trials	329.62	68.8	0.784

5. Interpretation

The primary question we sought to answer is how successfully the Monte Carlo (henceforth MC) agent performed. The non-MC agent, which only used the base probability estimation for any probabilistic decision making, seemed to fare just slightly better than random, ending with a mean success rate a few percent above 50%. This seems about appropriate, given that the agent isn't totally random, but is not very knowledgeable about its environment either. Using MC for agent decisions definitely increased the success rate, by as much as 20% averaged across the 250 boards.

The number of sample boards produced, however, doesn't have as large of an impact as expected. Running a minimum requirement of just 10 good samples seems to produce a minimally accurate result (though with higher variance across multiple runs). Choosing to run 50 or more trials doesn't seem to change the expected outcome so much as it continues to reduce variance between multiple runs. Therefore the difference in large and small sample sizes isn't so much a matter of total success rates, but of consistency across runs.

As for efficiency, it seems there is a way to substantially improve the efficiency of the Wumpus World setting without a negatively affecting the rate of success. Much of the bottleneck at high sample board counts involves a board situation with an extremely minor chance of producing a matching board. But insofar as our probabilities are concerned, it's only important to match either the breezes or the stenches. Matching both at the same time is overkill. By separating those parts we achieve a significantly higher rate of expected good boards. This reduces the time cost of finding probabilities by 80% or more.

In implementing this optimization, we worried that the exchange for time cost would be a significant drop in success rate. However, across various runs with a variety of sample counts, the success rate tended to match that of the original probability creation. Therefore we view this change as a wholesale improvement over the original implementation.

6. Critique

Over the 250 boards that we tested the Monte Carlo on we had about 71% success. This is much better than the other method where we had about 55% success. I would say that, though Monte Carlo is not time efficient, it has definite benefits. The first is that it seems to perform well. The next is that it is extremely easy to implement.

Is it worth the time cost? It might be. There are many ways that it could be implemented more efficiently. The random board generator could take into account information already known. It could use multi-threading. It is intuitive that more random boards generated tends to give it more

success. It seemed that we had about the same performance whether we generated 200 random boards or 1000, it performed well.

7. Log

Ryan : 18 hours
Sections 1, 3, 5
Leland : 18 hours
Sections 2, 4, 6