**CS 580 Reinforcement Learning**

**HW2**

**Yang Zhang 11529139**

**Part I. Implementation of Monte Carlo ES**

Result:

policy = [3, 3, 3, 0, 0, 0, 0, 0, 0, 3, 0, 2]

After Monte Carlo ES algorithm reaches its convergence, it ends up with the same optimal policy as policy iteration algorithm’s.

**Part II. Compare exploring starts with soft policies**

Result:

ES policy : [3, 3, 3, 0, 0, 0, 0, 0, 0, 3, 0, 2]

soft\_policy ( e = 0.3) : [{0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.075, 1: 0.075, 2: 0.7749999999999999, 3: 0.075}]

Observation : From the results above, the soft\_policy would choose optimal action at each state by the chance of 78%. There are 5 critical states [0, 1, 2, 10, 11] (only one optimal action allowed), so that the probability of producing optimal policy by soft monte-carlo (e=3) is 0.78^5 = 28.87%

**Part III The effectiveness of randomness in Soft policy**

**e = 0.1**, reach optimal after 200 iterations

learned soft policy:

[{0: 0.025, 1: 0.025, 2: 0.025, 3: 0.925}, {0: 0.025, 1: 0.025, 2: 0.025, 3: 0.925}, {0: 0.025, 1: 0.025, 2: 0.025, 3: 0.925}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.925, 1: 0.025, 2: 0.025, 3: 0.025}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.925, 1: 0.025, 2: 0.025, 3: 0.025}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.925, 1: 0.025, 2: 0.025, 3: 0.025}, {0: 0.025, 1: 0.025, 2: 0.025, 3: 0.925}, {0: 0.925, 1: 0.025, 2: 0.025, 3: 0.025}, {0: 0.025, 1: 0.025, 2: 0.925, 3: 0.025}]

**e = 0.2**, reach optimal after 150 iterations

learned soft policy:

[{0: 0.05, 1: 0.05, 2: 0.05, 3: 0.8500000000000001}, {0: 0.05, 1: 0.05, 2: 0.05, 3: 0.8500000000000001}, {0: 0.05, 1: 0.05, 2: 0.05, 3: 0.8500000000000001}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.8500000000000001, 1: 0.05, 2: 0.05, 3: 0.05}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.8500000000000001, 1: 0.05, 2: 0.05, 3: 0.05}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.8500000000000001, 1: 0.05, 2: 0.05, 3: 0.05}, {0: 0.05, 1: 0.05, 2: 0.05, 3: 0.8500000000000001}, {0: 0.8500000000000001, 1: 0.05, 2: 0.05, 3: 0.05}, {0: 0.05, 1: 0.05, 2: 0.8500000000000001, 3: 0.05}]

**e = 0.3**, reach optimal after 140 iterations

learned soft policy:

[{0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.075, 1: 0.075, 2: 0.075, 3: 0.7749999999999999}, {0: 0.7749999999999999, 1: 0.075, 2: 0.075, 3: 0.075}, {0: 0.075, 1: 0.075, 2: 0.7749999999999999, 3: 0.075}]

**e = 0.5**, reach optimal after 160 iterations

learned soft policy:

[{0: 0.125, 1: 0.125, 2: 0.125, 3: 0.625}, {0: 0.125, 1: 0.125, 2: 0.125, 3: 0.625}, {0: 0.125, 1: 0.125, 2: 0.125, 3: 0.625}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.625, 1: 0.125, 2: 0.125, 3: 0.125}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.625, 1: 0.125, 2: 0.125, 3: 0.125}, {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}, {0: 0.625, 1: 0.125, 2: 0.125, 3: 0.125}, {0: 0.125, 1: 0.125, 2: 0.125, 3: 0.625}, {0: 0.625, 1: 0.125, 2: 0.125, 3: 0.125}, {0: 0.125, 1: 0.125, 2: 0.625, 3: 0.125}]

**Conclusion:** In general, bigger randomness will lead to faster convergence. The reason is that with more randomness, the agent will exploring all possible state action pair faster.

**Part IV** The effectiveness of changing reward in Monte-carlo ES

|  |  |
| --- | --- |
| Reward Settings | Average Iterations to reach optimal |
| Standard setting (G 100, F -100, ELSE -3) | 300 |
| G 1000, F -1000, ELSE -3 | 200 |
| G 100, F -100, ELSE -30 | 150 |

From the above table, we can see that with more negative reward for making a move, the agent can learn the optimal policy faster.