

PolicyIterationAgent Report

Functions Modified

public void initRandomPolicy()

Initialised random policy for all the states by choosing a random move out of all the legal moves in each state. Added randomly chosen move to current policy.

protected void evaluatePolicy(double delta)

Added code to evaluate current policy until the values converge based on provided delta using. Created new policy map to temporarily store updated values for each state. Looped through each state in the current policy and selected action that was set by current policy. Calculated the values of future rewards for that action using the Bellman equation by summing up the expected rewards and then updating the value for that state. Then checking if values have converged (by comparing change to delta) and then updating policy values map with the local policy map for that state. If values have converged then breaking out of while loop.

protected boolean improvePolicy()

Added code to improve policy by calculating values for all actions in each state and choosing optimal one. Looped through each state in the current policy and calculating values for all possible actions using the Bellman equation. Sum up expected reward for each action and updated the best action to take by using the one with the highest value. Check if the best action is different from the current action in policy and if it is then set policyImproved to true. If the action is not different (even if value changed slightly) then policyImproved is false.

public void train()

Calls evaluatePolicy(delta) to calculate the state values using the action set by the current policy (initially random), evaluating current policy till state values converge. Then calls improvePolicy to update policy by calculating best action for each state based on the values. This process is repeated until the policy is stable (no improvements to the actions).

Testing PolicyIterationAgent

Performance Report

| Opponent | Wins | Losses | Draws |
|------------------|------|--------|-------|
| Defensive Agent | 46 | 0 | 4 |
| Aggressive Agent | 50 | 0 | 0 |
| Random Agent | 50 | 0 | 0 |

Results after testing agent in the class file 'TestPolicyIterationAgent.java'

Against Defensive Agent

```
Playing move: O(0,2)
|X|X|O|
|O|X| |
| | |O|

Playing move: X(2,1)
|X|X|O|
|O|X| |
| |X|O|

X won!
Wins: 46 Losses: 0 Draws: 4
```

Against Aggressive Agent

```
Playing move: O(2,1)
|X| | |
| | | |
|O|O|X|

Playing move: X(1,1)
|X| | |
| |X| |
|O|O|X|


X won!
Wins: 50 Losses: 0 Draws: 0
```

Against Random Agent

```
Playing move: O(1,2)
|X| |X|
| |O|O|
|O| |X|

Playing move: X(0,1)
|X|X|X|
| |O|O|
|O| |X|

X won!
Wins: 50 Losses: 0 Draws: 0
```

```
Finished after 1.795 seconds
Runs: 3/3      ✖ Errors: 0      ✖ Failures: 0
████████████████████████████████████████████████████████████████████████████████
>  TestPolicyIterationAgent [Runner: JUnit 4] (1.790 s)
```