

Coursework 2 – Tic-Tac-To: Markov Decision Processes & Reinforcement Learning (worth 25% of your final mark)

POLICY ITERATION

Question 4 (1 point): As in Question 2, this time test your Policy Iteration Agent against each of the provided agents 50 times and report on the results – how many games they won, lost & drew. The other agents are: *random*, *aggressive*, *defensive***Ans:**

- `initRandomPolicy()`: initializes random move generator and assigns random move to non terminal game state
- `evaluatePolicy()`: finds maximum change less than epsilon and updates policy values.
- `improvePolicy()`: replaces actions with best moves and changes flag if no changes were made (ie if stable)
- `Train()` : initializes and executes the algorithm using `evaluatePolicy()` and `improvePolicy()`, terminates and updates policy when stable.

	WINS	LOSS	DRAWS
RANDOM	50	0	0
AGRESSIVE	50	0	0
DEFENSIVE	46	0	4

Random:

```
97
98
99 // @Test
100 public void testRandom() {
101     System.out.println("Against Random Agent:");
102     int[] results=playAgainstEachOther(new PolicyIterationAgent(), new RandomAgent(), 50);
103     System.out.println("Wins: " + results[0] + " Losses: " + results[1] + " Draws: " + results[2]);
104     assertEquals(0, results[1]);
105 }
106
107
108
109 }
110
```

Finished after 2.115 seconds.
Runs: 1/1 Errors: 0 Failures: 0

Playing move: O(2,1)

```
|X|O| |
|X| | |
| |O| |
```

Playing move: X(2,0)

```
|X|O| |
|X| | |
|X|O| |
```

X won!

Wins: 50 Losses: 0 Draws: 0

DEFENSIVE:

```
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75 // @Test
76 public void testDefensive() {
77     System.out.println("Against Defensive Agent:");
78     int[] results=playAgainstEachOther(new PolicyIterationAgent(), new DefensiveAgent(), 50);
79     System.out.println("Wins: " + results[0] + " Losses: " + results[1] + " Draws: " + results[2]);
80     assertEquals(0, results[1]);
81 }
82
83
84 //
85 // @Test
86 // public void testAggressive() {
87 //     System.out.println("Against Aggressive Agent:");
88 // }
```

Finished after 1.473 seconds
Runs: 1/1 Errors: 0 Failures: 0

TestPolicyIterationAgent [Runner: JUnit]

Playing defensive move
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

AGGRESSIVE:

```
84 //
85 //
86 // @Test
87 // public void testAggressive() {
88 //     System.out.println("Against Aggressive Agent:");
89 //
90 //     int[] results=playAgainstEachOther(new PolicyIterationAgent(), new AggressiveAgent(), 50);
91 //     System.out.println("Wins: " + results[0] + " Losses: " + results[1] + " Draws: " + results[2]);
92 //     assertEquals(0, results[1]);
93 // }
94 //
95 //
96 //
```

Finished after 1.64 seconds
Runs: 1/1 Errors: 0 Failures: 0

TestPolicyIterationAgent [Runner: JUnit]

Playing move: O(1,0)

```
|X| | |
|O|X| |
| |O| |
```

Playing move: X(2,2)

```
|X| | |
|O|X| |
| |O|X|
```

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