

# Implementation of Q-Learning for Dots and Boxes

## IMPLEMENTATION OF Q-LEARNING TECHNIQUE – 2 x 2 GRID

Learning rate : 0.6

Discount factor: 0.7

Epsilon : 0.6 – for epsilon greedy policy

**For 100 games – self Play:**

Time consumed : 1 second

Agent 1 wins : 36

Agent 2 wins : 35

Draws : 29

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 29 16:06:40 2019
4
5 @author: Srujan Panuganti
6 """
7
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from play import play as pl
15 from display import display
16 import pickle
17
18 total_games = 100
19 game_count = 0
20 env = dbs(grid_size = (2,2))
21 display_it = display(grid_size = (2,2))
22 agent1 = q_agent()
23 agent2 = q_agent()
24 q_class = q_learn(total_actions = 12)
25 game1 = pl(env,agent1,agent2,display_it)
26 #winner, q_class = game1.play_game(q_class)
27
28 agent1_wins = 0
29 agent2_wins = 0
30 draws = 0
31
32 while(game_count < total_games):
33     env = dbs(grid_size = (2,2))
34     agent1 = q_agent()
35     agent2 = q_agent()
36     game1 = pl(env,agent1,agent2,display_it)
37     winner, q_class = game1.play_game(q_class)
38     game_count +=1
39
40     if winner == 1:
41         agent1_wins += 1
42     elif winner == 2:
```

Name	Type	Size	Value
agent1_wins	int	1	36
agent2_wins	int	1	35
draws	int	1	29

```
Python console
Console 1/A
agent1 turn True
agent2 turn True
agent 2 won 0 4
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent2 turn True
agent2 turn False
agent1 turn True
agent1 turn True
agent1 turn True
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn True
agent1 turn True
agent1 turn True
agent1 won 4 0
agent 1 wins = 36 agent 2 wins = 35 draws = 29
In [5]:
```

**For 1000 games – Self Play:**

Time consumed: 13 seconds

Agent 1 wins :445

Agent 2 wins: 314

Draws : 241

```
untitled0.py test.py
2 """
3 Created on Mon Apr 29 16:06:40 2019
4
5 @author: Srujan Panuganti
6 """
7
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from play import play as pl
15 from display import display
16 import pickle
17
18 total_games = 1000
19 game_count = 0
20 #env = dbs(grid_size = (2,2))
21 display_it = display(grid_size = (2,2))
22 #agent1 = q_agent()
23 #agent2 = q_agent()
24 q_class = q_learn(total_actions = 12)
25 #game1 = pl(env,agent1,agent2,display_it)
26 #winner, q_class = game1.play_game(q_class)
27
28 agent1_wins = 0
29 agent2_wins = 0
30 draws = 0
31
32 while(game_count < total_games):
33     env = dbs(grid_size = (2,2))
34     agent1 = q_agent()
35     agent2 = q_agent()
36     game1 = pl(env,agent1,agent2,display_it)
37     winner, q_class = game1.play_game(q_class)
38     game_count +=1
39
40     if winner == 1:
41         agent1_wins += 1
42     elif winner ==2:
43         agent2_wins += 1
```

Name	Type	Size	Value
agent1_wins	int	1	445
agent2_wins	int	1	314
draws	int	1	241

```
Python console
Console 1/A
agent1 turn True
agent1 turn True
agent1 won 4 0
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn False
agent2 turn True
agent2 turn False
agent1 turn True
agent1 turn True
agent1 won 3 1
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent2 turn False
agent1 turn True
agent2 turn False
agent1 turn True
agent2 turn True
agent1 turn False
agent2 turn True
agent1 turn True
agent2 turn False
agent1 turn True
draw 2 2
agent 1 wins = 445 agent 2 wins = 314 draws = 241
In [3]:
```

For 10000 games – Self Play:

Time consumed 1 minute 45 seconds

Agent 1 wins : 4440

Agent 2 wins : 3181

Draws : 2379

```
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from play import play as pl
15 from display import display
16 import pickle
17
18 total_games = 10000
19 game_count = 0
20 #env = dbs(grid_size = (2,2))
21 display_it = display(grid_size = (2,2))
22 #agent1 = q_agent()
23 #agent2 = q_agent()
24 q_class = q_learn(total_actions = 12)
25 #game1 = pl(env,agent1,agent2,display_it)
26 #winner, q_class = game1.play_game(q_class)
27
28 agent1_wins = 0
29 agent2_wins = 0
30 draws = 0
31
32 while(game_count < total_games):
33     env = dbs(grid_size = (2,2))
34     agent1 = q_agent()
35     agent2 = q_agent()
36     game1 = pl(env,agent1,agent2,display_it)
37     winner, q_class = game1.play_game(q_class)
38     game_count +=1
39
40     if winner == 1:
41         agent1_wins += 1
42     elif winner ==2:
43         agent2_wins += 1
44     elif winner == 0:
45         draws += 1
46
47 print('agent 1 wins = ',agent1_wins,'agent 2 wins = ',agent2_wins,'draws = ',draws)
48
49 pickle_out = open("dict10000_games_2x2.pickle","wb")
```

Name	Type	Size	Value
agent1_wins	int	1	4440
agent2_wins	int	1	3181
draws	int	1	2379

```
Python console
Console 1/A
agent2 turn True
agent2 turn False
agent1 turn True
agent1 won 3 1
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent2 turn False
agent1 turn True
agent2 turn True
agent2 won 1 3
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn False
agent1 turn False
agent2 turn True
agent2 won 1 3
agent 1 wins = 4440 agent 2 wins = 3181 draws = 2379
In [4]:
```

# IMPLEMENTATION OF Q-LEARNING TECHNIQUE – 3 x 3 GRID

## For 100 games – self -Play

Time consumed: 1 minute 30 seconds

Agent 1 wins : 44 , Agent 2 wins : 56

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 29 16:06:40 2019
4
5 @author: Srujan Panuganti
6 """
7
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from play import play as pl
15 from display import display
16 import pickle
17
18 total_games = 100
19 game_count = 0
20 env = dbs(grid_size = (3,3))
21 display_it = display(grid_size = (3,3))
22 agent1 = q_agent()
23 agent2 = q_agent()
24 q_class = q_learn(total_actions = 24)
25 game1 = pl(env,agent1,agent2,display_it)
26 #winner, q_class = game1.play_game(q_class)
27
28 agent1_wins = 0
29 agent2_wins = 0
30 draws = 0
31
32
33 while(game_count < total_games):
34     env = dbs(grid_size = (3,3))
35     agent1 = q_agent()
36     agent2 = q_agent()
37     game1 = pl(env,agent1,agent2,display_it)
38     winner, q_class = game1.play_game(q_class)
39     game_count +=1
40     print('game number = ',game_count)
41
42     if winner == 1:
```

Name	Type	Size	Value
agent1_wins	int	1	44
agent2_wins	int	1	56
draws	int	1	0

```
Python console
Console 1/A
game number = 86
agent 2 won 1 8
game number = 87
agent 1 won 8 1
game number = 88
agent 1 won 8 1
game number = 89
agent 1 won 6 3
game number = 90
agent 1 won 5 4
game number = 91
agent 1 won 6 3
game number = 92
agent 2 won 3 6
game number = 93
agent 2 won 4 5
game number = 94
agent 1 won 6 3
game number = 95
agent 2 won 2 7
game number = 96
agent 2 won 1 8
game number = 97
agent 2 won 1 8
game number = 98
agent 2 won 3 6
game number = 99
agent 1 won 5 4
game number = 100
agent 1 wins = 44 agent 2 wins = 56 draws = 0
In [8]:
```

## For 1000 games – Self Play:

Time consumed : 14 minutes 10 seconds

Agent 1 wins = 504 , Agent 2 wins = 496

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 29 16:06:40 2019
4
5 @author: Srujan Panuganti
6 """
7
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from play import play as pl
15 from display import display
16 import pickle
17 import time
18
19 start = time.time()
20
21 total_games = 1000
22 game_count = 0
23 env = dbs(grid_size = (3,3))
24 display_it = display(grid_size = (3,3))
25 agent1 = q_agent()
26 agent2 = q_agent()
27 q_class = q_learn(total_actions = 24)
28 #game1 = pl(env,agent1,agent2,display_it)
29 #winner, q_class = game1.play_game(q_class)
30
31 agent1_wins = 0
32 agent2_wins = 0
33 draws = 0
34
35
36 while(game_count < total_games):
37     env = dbs(grid_size = (3,3))
38     agent1 = q_agent()
39     agent2 = q_agent()
40     game1 = pl(env,agent1,agent2,display_it)
41     winner, q_class = game1.play_game(q_class)
42     game_count +=1
43     print('game number = ',game_count)
```

Name	Type	Size	Value
agent1_wins	int	1	504
agent2_wins	int	1	496
draws	int	1	0

```
Python console
Console 1/A
game number = 986
agent 2 won 2 7
game number = 987
agent 1 won 6 3
game number = 988
agent 1 won 8 1
game number = 989
agent 2 won 2 7
game number = 990
agent 1 won 8 1
game number = 991
agent 1 won 6 3
game number = 992
agent 1 won 8 1
game number = 993
agent 1 won 8 1
game number = 994
agent 1 won 7 2
game number = 995
agent 1 won 7 2
game number = 996
agent 2 won 2 7
game number = 997
agent 1 won 6 3
game number = 998
agent 2 won 3 6
game number = 999
agent 1 won 5 4
game number = 1000
agent 1 wins = 504 agent 2 wins = 496 draws = 0
In [9]:
```

For 10000 games:

Time consumed = 3516.756 seconds (59.6 minutes)

Agent 1 wins = 5045 , Agent 2 wins = 4955

The screenshot shows a Jupyter Notebook with a Python script and its output. The script simulates 10,000 games between two agents on a 3x3 grid. Agent 1 uses a Q-learning algorithm, while Agent 2 is a random agent. The results are summarized in a table and printed in the console.

```
14 from play import play as pl
15 from display import display
16 import pickle
17 import time
18
19 start = time.time()
20
21 total_games = 10000
22 game_count = 0
23 env = dbs(grid_size = (3,3))
24 display_it = display(grid_size = (3,3))
25 agent1 = q_agent()
26 agent2 = q_agent()
27 q_class = q_learn(total_actions = 24)
28 #game1 = pl(env,agent1,agent2,display_it)
29 #winner, q_class = game1.play_game(q_class)
30
31 agent1_wins = 0
32 agent2_wins = 0
33 draws = 0
34
35 while(game_count < total_games):
36     env = dbs(grid_size = (3,3))
37     agent1 = q_agent()
38     agent2 = q_agent()
39     game1 = pl(env,agent1,agent2,display_it)
40     winner, q_class = game1.play_game(q_class)
41     game_count +=1
42     print('game number = ',game_count)
43
44     if winner == 1:
45         agent1_wins += 1
46     elif winner ==2:
47         agent2_wins += 1
48     elif winner == 0:
49         draws += 1
50
51
52 end = time.time()
53
54
55 print('agent 1 wins = ',agent1_wins,'agent 2 wins = ',agent2_wins,'draws = ',draws)
```

Name	Type	Size	Value
agent1_wins	int	1	5045
agent2_wins	int	1	4955
draws	int	1	0

```
game number = 9986
agent 1 won 9 0
game number = 9987
agent 2 won 4 5
game number = 9988
agent 1 won 6 3
game number = 9989
agent 2 won 2 7
game number = 9990
agent 1 won 6 3
game number = 9991
agent 2 won 2 7
game number = 9992
agent 2 won 3 6
game number = 9993
agent 2 won 2 7
game number = 9994
agent 1 won 6 3
game number = 9995
agent 1 won 7 2
game number = 9996
agent 1 won 9 0
game number = 9997
agent 2 won 1 8
game number = 9998
agent 1 won 5 4
game number = 9999
agent 2 won 3 6
game number = 10000
agent 1 wins = 5045 agent 2 wins = 4955 draws = 0
time taken = 3516.75639462471
```

## PLAYING WITH RANDOM AGENT 2X2 GRID:

Agent 1 is initialized with q table (based on 100 games experience), where as agent 2 is a random agent

Agent 1 won = 787, agent 2 wins = 77, draws = 136

Agent 1 is initialized with q table (based on 1000 games experience), where as agent 2 is a random agent

Agent 1 won = 804, agent 2 wins = 65, draws = 131

Agent 1 is initialized with q table (based on 10000 games experience), where as agent 2 is a random agent

Agent 1 won = 827, agent 2 wins = 61, draws = 112

```

5 @author: Srujan Panuganti
6 """
7
8
9 import numpy as np
10 from operator import add
11 from dots_and_boxes import dots_and_boxes as dbs
12 from q_algorithm import q_learn
13 from agent import q_agent
14 from agent import random_agent
15 from play import play as pl
16 from display import display
17 import pickle
18
19 total_games = 1000
20 game_count = 0
21 # =====
22 # env = dbs(grid_size = (2,2))
23 # =====
24 display_it = display(grid_size = (2,2))
25 # =====
26 # agent1 = q_agent()
27 # =====
28 # =====
29 # agent2 = random_agent()
30 # =====
31
32 pickle_in = open("dict10000_games_2x2.pickle", "rb")
33 q_table = pickle.load(pickle_in)
34
35 q_class = q_learn(q_table)
36
37 # =====
38 # =====
39 # game1 = pl(env, agent1, agent2, display_it)
40 # winner = game1.play_game(q_class)
41 # =====
42 # =====
43
44
45
46 agent1_wins = 0
47 agent2_wins = 0

```

Name	Type	Size	Value
agent1_wins	int	1	827
agent2_wins	int	1	61
draws	int	1	112

```

Python console
Console 1/A
agent1 turn False
agent2 turn True
agent 1 won 3 1
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn True
agent1 turn False
agent2 turn True
draw 2 2
agent1 turn False
agent2 turn False
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn True
agent1 turn False
agent2 turn False
agent1 turn True
agent1 turn True
agent 1 won 4 0
agent 1 wins = 827 agent 2 wins = 61 draws = 112
In [31]:
History log IPython console

```

## PLAYING WITH RANDOM AGENT 3X3 GRID:

Agent 1 is initialized with q table (based on 100 games experience), where as agent 2 is a random agent

For 100 games, Agent 1 won = 94, agent 2 wins = 6, draws = 0

Time taken = 49 seconds

Agent 1 is initialized with q table (based on 1000 games experience), where as agent 2 is a random agent

Agent 1 won = 95, agent 2 wins = 5, draws = 0

Time taken = 50 seconds

Agent 1 is initialized with q table (based on 10000 games experience), where as agent 2 is a random agent

Agent 1 won = 94, agent 2 wins = 6, draws = 0

```

14 from play import play as pl
15 from display import display
16 import pickle
17 import time
18
19 start = time.time()
20
21 total_games = 100
22 game_count = 0
23
24 pickle_in = open("dict10000_games.pickle", "rb")
25 q_table = pickle.load(pickle_in)
26
27 #print(q_table)
28
29
30 env = dbs(grid_size = (3,3))
31 display_it = display(grid_size = (3,3))
32 agent1 = q_agent()
33 agent2 = random_agent()
34
35 q_class = q_learn(24, q_table)
36 #q_class = q_learn(q_table)
37
38
39 #game1 = pl(env, agent1, agent2, display_it)
40 #winner = game1.play_game(q_class)
41
42 agent1_wins = 0
43 agent2_wins = 0
44 draws = 0
45
46
47 while(game_count < total_games):
48     env = dbs(grid_size = (3,3))
49     agent1 = q_agent()
50     agent2 = random_agent()
51     game1 = pl(env, agent1, agent2, display_it)
52     winner = game1.play_game(q_class)
53     game_count +=1
54     print('game number = ', game_count)
55
56     if winner == 1:

```

Name	Type	Size	Value
agent1_wins	int	1	94
agent2_wins	int	1	6
draws	int	1	0

```

Python console
Console 1/A
game number = 100
agent 1 won 5 4
game number = 87
agent 2 won 4 5
game number = 88
agent 1 won 8 1
game number = 89
agent 1 won 8 1
game number = 90
agent 1 won 7 2
game number = 91
agent 1 won 7 2
game number = 92
agent 1 won 8 1
game number = 93
agent 1 won 8 1
game number = 94
agent 1 won 8 1
game number = 95
agent 1 won 7 2
game number = 96
agent 1 won 6 3
game number = 97
agent 1 won 5 4
game number = 98
agent 1 won 5 4
game number = 99
agent 1 won 6 3
game number = 100
agent 1 wins = 94 agent 2 wins = 6 draws = 0
time taken = 48.34918975830078

In [46]:
History log IPython console

```

## FUNCTION APPROXIMATION FOR 3X3 GRID:

Attempts have made to implement a neural network with 1 input layer, 2 hidden layer and an output layer is implemented to train the game.

Randomly few states are selected from the q-table whose q-values are not equal to zero as the features to train the network. But promising results are not obtained