

QLearning For Videogames

Steven Walton
University of Oregon

CIS 510: Multi-Agent Systems
3 June 2019

What is Q-Learning?

- ▶ Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.

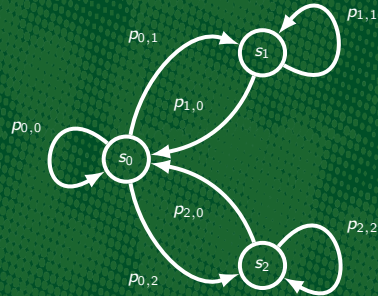


Figure: Sample Markov Decision Process

What is Q-Learning?

- ▶ Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.
- ▶ We want to find an optimal policy (set of actions)

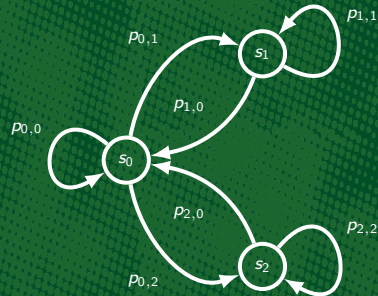


Figure: Sample Markov Decision Process

What is Q-Learning?

- ▶ Markov Decision Processes (MDPs) can represent decision making processes with random outcomes.
- ▶ We want to find an optimal policy (set of actions)
- ▶ We learn with the Bellman Equation, using an iterative process

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

α = learning rate

r = reward

γ = discount

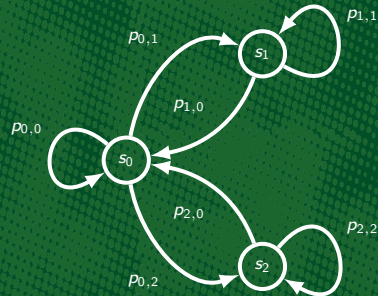


Figure: Sample Markov Decision Process

Framework

- ▶ OpenAI Gym: Allows for abstracts out environments and allows researchers to focus on creating algorithms.
- ▶ OpenAI Retro: Wrapps Gym framework to focus on retro videogames.
- ▶ Allows same code to be run on any game that is integrated.
- ▶ Creates an environment to (supposedly) allow for easy integration of new games



Integration

- ▶ Need to collect RAM values (used BizHawk).

The image shows a screenshot of the SNES Bizhawk emulator running Mario Kart 64. The game is in progress, showing a track with Mario's kart. A red circle highlights the timer at the top right, which reads 03' 27" 68. Below the game screen, the SNES WRAM memory dump is visible, with the address 000100 highlighted in red, showing the value 58 71 27 00. To the right, the RAM Search tool is open, displaying a table of 17 addresses in WRAM. The table has columns for Address, Value, Prev, and Chan. The addresses 000100 and 000101 are highlighted in red, showing values 88 and 113 respectively, with channel 12513. The RAM Search tool also includes search filters and comparison options.

Address	Value	Prev	Chan
000003	17	17	12861
000034	49	49	15333
000100	88	88	12513
000101	113	113	12513
00007A	153	153	12517
000DF0	32	32	13467
000E4E	7	7	13472
0013EA	1	1	12511
0015EA	97	97	12511
0016EA	253	253	12512
0017EA	66	66	12511
001EBA	4	4	13471
001EE4	2	2	13469
001EE6	20	20	13471
01DE32	4	4	13471

Integration

- ▶ Need to collect RAM values (used BizHawk).

RAM Search

File Settings Search Options Columns

Search New C X [Icons]

17 addresses WRAM

Address	Value	Prev	Chan...
000003	17	17	12861
000034	49	49	15333
000100	88	88	12513
000101	113	113	12513
00007A	153	153	12517
000DF0	32	32	13467
000E4E	7	7	13472
0013EA	1	1	12511
0015EA	97	97	12511
0016EA	253	253	12512
0017EA	66	66	12511
001EBA	4	4	13471
001EE4	2	2	13469
001EE6	20	20	13471
01DE32	4	4	13471

Compare To / By

☒ Previous Value

☐ Specific Value: 0

☐ Specific Address: 000000

☐ Specific Changes: 0

☐ Difference of: 0

Comparison Operator

☐ Equal To

☒ Not Equal To

☐ Less Than

☐ Greater Than

☐ Less Than or Equal To

☐ Greater Than or Equal To

☐ Different by: 0

Size Display

1 Byte Unsigned

Search [Icons]

0 addresses removed

SNES WRAM - 0x20000 Address 0

Address	Value
0000D0	02 00 00 00 00 00 C0 00 00 00 00 C0 00 00 00
0000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00
000100	58 71 27 00 03 08 00 A0 00 00 00 00 00 17 Xq
000120	FF FF 00 85 07 00 02 00 00 00 00 00 00 0E
000130	08 00 06 00 0E 00 02 00 0C 00 04 00 0A 00
000140	00 00 FF FF 0E 00 00 1E 00 30 02 00 85 00
000150	00 00 00 00 04 00 04 00 00 00 00 00 00 20
000160	00 0F 00 00 00 00 00 00 00 3D AD 2E FF 4E FF
000170	8E FF 00 06 00 00 00 FF FF 10 30 00 00 00
000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00
000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00



Integration

- ▶ Need to collect RAM values (used BizHawk).
- ▶ Convert Hex to decimal and add rambase number and integrate into scenario and data files

```
1 {
2   "info": {
3     "lap": {
4       "address": 8261825,
5       "type": "|█"
6     },
7     "lapsize": {
8       "address": 8257864,
9       "type": "|█"
10    },
11    "checkpoint": {
12      "address": 8261852,
13      "type": "|█"
14    },
15    "rank": {
16      "address": 8261696,
17      "type": "|█"
18    },
19    "coins": {
20      "address": 8261120,
21      "type": "|█"
22    },
23    "minute": {
24      "address": 8257796,
25      "type": "|█"
26    },
27    "second": {
28      "address": 8257794,
29      "type": "|u1"
30    },
31    "millisecond": {
32      "address": 8257793,
33      "type": "|u1"
34    }
35  }
36 }
```

```
1 {
2   "done": {
3     "variables": {
4       "finish": {
5         "reference": "coins",
6         "op": "█"
7       }
8     }
9   },
10  "reward": {
11    "variables": {
12      "rank": {
13        "reward": 1.0
14      }
15    }
16  },
17  "actions": [
18    [[], ["UP"], ["DOWN"]],
19    [[], ["LEFT"], ["RIGHT"]],
20    [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"], ["A"], ["B"], ["Y"], ["R"], ["A", "B"], ["R", "A", "Y"], ["A", "B"], ["A", "Y"]]]
21 ]
```



Integration

- ▶ Need to collect RAM values (used BizHawk).
- ▶ Convert Hex to decimal and add rambase number and integrate into scenario and data files
- ▶ Define actions and rewards for the game.

```
1 {  
2   "info": {  
3     "lap": {  
4       "address": 8261825,  
5       "type": "|█"  
6     },  
7     "lapsize": {  
8       "address": 8257864,  
9       "type": "|█"  
10    },  
11    "checkpoint": {  
12      "address": 8261852,  
13      "type": "|█"  
14    },  
15    "rank": {  
16      "address": 8261696,  
17      "type": "|█"  
18    },  
19    "coins": {  
20      "address": 8261120,  
21      "type": "|█"  
22    },  
23    "minute": {  
24      "address": 8257796,  
25      "type": "|█"  
26    },  
27    "second": {  
28      "address": 8257794,  
29      "type": "|u1"  
30    },  
31    "millisecond": {  
32      "address": 8257793,  
33      "type": "|u1"  
34    }  
35  }  
36 }
```

```
1 {  
2   "done": {  
3     "variables": {  
4       "finish": {  
5         "reference": "coins",  
6         "op": "█"  
7       },  
8     },  
9   },  
10  "reward": {  
11    "variables": {  
12      "rank": {  
13        "reward": 1.0  
14      },  
15    },  
16  },  
17  "actions": [  
18    [[], ["UP"], ["DOWN"]],  
19    [[], ["LEFT"], ["RIGHT"]],  
20    [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],  
21    ["R"], ["B"], ["A"], ["Y"], ["R"], ["A", "B"], ["R", "A", "Y"],  
22    ["A", "B"], ["A", "Y"]]  
23  ]  
24 }
```



Integration

- ▶ Need to collect RAM values (used BizHawk).
- ▶ Convert Hex to decimal and add rambase number and integrate into scenario and data files
- ▶ Define actions and rewards for the game.
- ▶ Not all RAM values are so easy to get nor define. (Lua scripts)

```
1 {  
2   "info": {  
3     "lap": {  
4       "address": 8261825,  
5       "type": "|█"  
6     },  
7     "lapsize": {  
8       "address": 8257864,  
9       "type": "|█"  
10    },  
11    "checkpoint": {  
12      "address": 8261852,  
13      "type": "|█"  
14    },  
15    "rank": {  
16      "address": 8261696,  
17      "type": "|█"  
18    },  
19    "coins": {  
20      "address": 8261120,  
21      "type": "|█"  
22    },  
23    "minute": {  
24      "address": 8257796,  
25      "type": "|█"  
26    },  
27    "second": {  
28      "address": 8257794,  
29      "type": "|u1"  
30    },  
31    "millisecond": {  
32      "address": 8257793,  
33      "type": "|u1"  
34    }  
35  }  
36 }
```

```
1 {  
2   "done": {  
3     "variables": {  
4       "finish": {  
5         "reference": "coins",  
6         "op": "█"  
7       }  
8     },  
9   },  
10  "reward": {  
11    "variables": {  
12      "rank": {  
13        "reward": 1.0  
14      }  
15    }  
16  },  
17  "actions": [  
18    [{"UP", ["UP"], ["DOWN"]},  
19    [{"LEFT", ["LEFT"], ["RIGHT"]},  
20    [{"A", ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],  
21    [{"R", ["B"], ["R"], ["Y"], ["R", "A", "B"], ["R", "A", "Y"],  
    [{"A", "B"], ["A", "Y"]]  
21  ]
```



Integration

- ▶ Need to collect RAM values (used BizHawk).
- ▶ Convert Hex to decimal and add rambase number and integrate into scenario and data files
- ▶ Define actions and rewards for the game.
- ▶ Not all RAM values are so easy to get nor define. (Lua scripts)
- ▶ Thanks SethBling

```
1 {  
2   "info": {  
3     "lap": {  
4       "address": 8261825,  
5       "type": "|█"  
6     },  
7     "lapsize": {  
8       "address": 8257864,  
9       "type": "|█"  
10    },  
11    "checkpoint": {  
12      "address": 8261852,  
13      "type": "|█"  
14    },  
15    "rank": {  
16      "address": 8261696,  
17      "type": "|█"  
18    },  
19    "coins": {  
20      "address": 8261120,  
21      "type": "|█"  
22    },  
23    "minute": {  
24      "address": 8257796,  
25      "type": "|█"  
26    },  
27    "second": {  
28      "address": 8257794,  
29      "type": "|u1"  
30    },  
31    "millisecond": {  
32      "address": 8257793,  
33      "type": "|u1"  
34    }  
35  }  
36 }
```

```
1 {  
2   "done": {  
3     "variables": {  
4       "finish": {  
5         "reference": "coins",  
6         "op": "█"  
7       },  
8     },  
9   },  
10  "reward": {  
11    "variables": {  
12      "rank": {  
13        "reward": 1.0  
14      },  
15    },  
16  },  
17  "actions": [  
18    [[], ["UP"], ["DOWN"]],  
19    [[], ["LEFT"], ["RIGHT"]],  
20    [[], ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],  
21    ["R"], ["B"], ["R"], ["Y"], ["R", "A", "B"], ["R", "A", "Y"],  
22    ["A", "B"], ["A", "Y"]]  
23  ]  
24 }
```



- ▶ Need to collect RAM values (used BizHawk)
- ▶ Convert Hex to decimal and add rambase number and integrate into scenario and data files
- ▶ Define actions and rewards for the game.
- ▶ Not all RAM values are so easy to get nor define. (Lua scripts)
- ▶ Thanks SethBling



```

{
  {
    "reference": "coins",
    "value": 100
  }
}

{
  "hard": 1.0
}

["DOWN"],
["RIGHT"],
["B"], ["Y"], ["START"], ["R"], ["R"], ["A",
"Y"], ["R", "A", "B"], ["R", "A", "Y"],
"]]]

```

So Let's Make Something Work



So Let's Make Something Work

Goals

- ▶ Get a program to work nicely with retro
- ▶