

QLearning For Videogames

Steven Walton
University of Oregon
Github: [stevenwalton/Retro-Learner](https://github.com/stevenwalton/Retro-Learner)

CIS 510: Multi-Agent Systems
3 June 2019

What is Q-Learning?

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

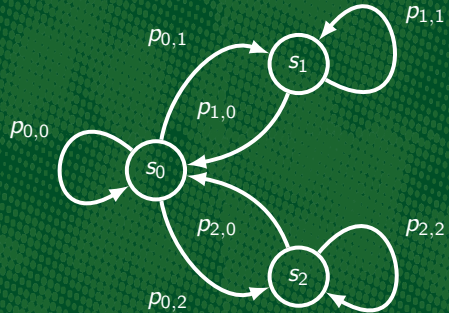


Figure: Sample Markov Decision Process

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)

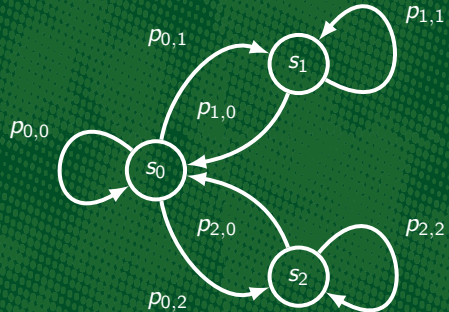


Figure: Sample Markov Decision Process

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

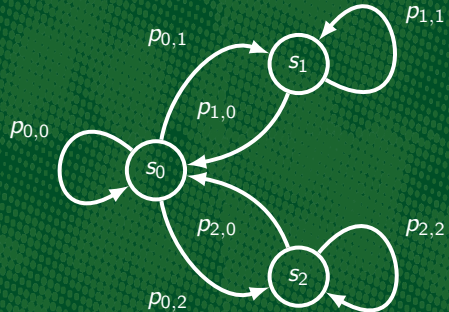


Figure: Sample Markov Decision Process

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 screenshot displays the BizHawk emulator interface. On the left, a Super Mario Kart game is running, showing a track with a red car. On the right, the RAM Search tool is open, displaying a table of 17 addresses and their values. The table is titled "17 addresses" and "WRAM". The table has columns for Address, Value, Prev, and Chan. The values are highlighted in red. The RAM Search tool also includes a search bar, a list of addresses, and a comparison operator section.

Address	Value	Prev	Chan
000003	17	17	12861
000034	49	49	15333
000100	88	88	12513
000101	113	113	12513
00007A	153	153	12517
0000F0	32	32	13467
000E4E	7	7	13472
0013EA	1	1	12511
0015EA	97	97	12511
0016EA	253	253	12512
0017EA	66	66	12511
0018EA	4	4	13471
001EE4	2	2	13469
001EE6	20	20	13471
01DE32	4	4	13471

Below the table, the RAM Search tool shows a list of addresses removed, with a count of 0 addresses removed.

Integration

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

The screenshot shows the BizHawk emulator interface. On the left, a Super Mario Kart game is running, displaying a track with a large 'K' and a timer of 03' 27" 68. On the right, the RAM Search tool is open, showing a list of 17 addresses in WRAM. The addresses 000100, 000101, 000102, and 000103 are highlighted in red. The values for these addresses are 88, 88, 88, and 12513 respectively. The RAM Search tool also shows a comparison operator set to 'Not Equal To' and a display size of 1 Byte.

Addr.	Value	Prev	Chan.
000003	17	17	12861
000034	49	49	15333
000100	88	88	12513
000101	113	113	12513
00007A	153	153	12517
0000F0	32	32	13467
000E4E	7	7	13472
0013EA	1	1	12511
0015EA	97	97	12511
0016EA	253	253	12512
0017EA	66	66	12511
0018EA	4	4	13471
001EE4	2	2	13469
001EE6	20	20	13471
01DE32	4	4	13471

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": 0261825,
5        "type": "I"
6      },
7      "start": {
8        "address": 0257864,
9        "type": "I"
10     },
11     "checkpoint": {
12       "address": 0261852,
13       "type": "I"
14     },
15     "rank": {
16       "address": 0261696,
17       "type": "I"
18     },
19     "coins": {
20       "address": 0261120,
21       "type": "I"
22     },
23     "minute": {
24       "address": 0257796,
25       "type": "I"
26     },
27     "second": {
28       "address": 0257794,
29       "type": "I"
30     },
31     "millisecond": {
32       "address": 0257793,
33       "type": "I"
34     }
35   }
36 }
```

```
1  {
2    "done": {
3      "variables": {
4        "finish": {
5          "reference": "coins",
6          "op": "I"
7        }
8      }
9    },
10    "reward": {
11      "variables": {
12        "rank": {
13          "reward": 1.0
14        }
15      }
16    },
17    "actions": {
18      [
19        [
20          [
21            [
22              ["UP"], ["DOWN"],
23              ["LEFT"], ["RIGHT"],
24              ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],
25              ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"],
26              ["A", "B"], ["A", "Y"]
27            ]
28          ]
29        ]
30      ]
31    }
32 }
```


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
3 "info": {
4   "lap": {
5     "address": 0261825,
6     "type": "I"
7   },
8   "start": {
9     "address": 0257864,
10    "type": "I"
11  },
12  "checkpoint": {
13    "address": 0261852,
14    "type": "I"
15  },
16  "rank": {
17    "address": 0261696,
18    "type": "I"
19  },
20  "coins": {
21    "address": 0261120,
22    "type": "I"
23  },
24  "minute": {
25    "address": 0257796,
26    "type": "I"
27  },
28  "second": {
29    "address": 0257794,
30    "type": "I"
31  },
32  "millisecond": {
33    "address": 0257793,
34    "type": "I"
35  }
36 }
```

```
1 {
2   "done": {
3     "variables": {
4       "finish": {
5         "reference": "coins",
6         "op": "I"
7       }
8     }
9   },
10  "reward": {
11    "variables": {
12      "rank": {
13        "reward": 1.0
14      }
15    }
16  },
17  "actions": {
18    [
19      ["UP"], ["DOWN"],
20      ["LEFT"], ["RIGHT"],
21      ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],
22      ["R", "B"], ["R", "Y"], ["R", "A", "B"], ["R", "A", "Y"],
23      ["A", "B"], ["A", "Y"]
24    ]
25  }
```

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": 0261825,
5        "type": "I"
6      },
7      "start": {
8        "address": 0257864,
9        "type": "I"
10     },
11     "checkpoint": {
12       "address": 0261852,
13       "type": "I"
14     },
15     "rank": {
16       "address": 0261696,
17       "type": "I"
18     },
19     "coins": {
20       "address": 0261120,
21       "type": "I"
22     },
23     "minute": {
24       "address": 0257796,
25       "type": "I"
26     },
27     "second": {
28       "address": 0257794,
29       "type": "I"
30     },
31     "milliseconds": {
32       "address": 0257793,
33       "type": "I"
34     }
35   }
36 }
```

```
1  {
2    "done": {
3      "variables": {
4        "finish": {
5          "reference": "coins",
6          "op": "I"
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 }
```

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
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

"info": {
  "lap": {
    "address": 0261825,
    "type": "I"
  },
  "start": {
    "address": 0257864,
    "type": "I"
  },
  "checkpoint": {
    "address": 0261852,
    "type": "I"
  },
  "rank": {
    "address": 0261696,
    "type": "I"
  },
  "coins": {
    "address": 0261120,
    "type": "I"
  },
  "minute": {
    "address": 0257796,
    "type": "I"
  },
  "second": {
    "address": 0257794,
    "type": "I"
  },
  "millisecond": {
    "address": 0257793,
    "type": "I"
  }
}

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

"done": {
  "variables": {
    "finish": {
      "reference": "coins",
      "op": "I"
    }
  }
},
"reward": {
  "variables": {
    "rank": {
      "reward": 1.0
    }
  }
},
"actions": {
  [{"UP", "DOWN"}],
  [{"LEFT", "RIGHT"}],
  [{"A", "B"}, {"Y", "START"}, {"R", "R"}, {"A", "A"}, {"B", "B"}, {"Y", "Y"}, {"R", "R"}, {"A", "A"}, {"Y", "Y"}, {"A", "B"}, {"A", "Y"}]
}
```



- ▶ Need to collect (used BizHawk)
- ▶ Convert Hex to add rambase numbers into save data files
- ▶ Define actions at the game.
- ▶ Not all RAM values easy to get nor (scripts)
- ▶ Thanks SethBlind



```

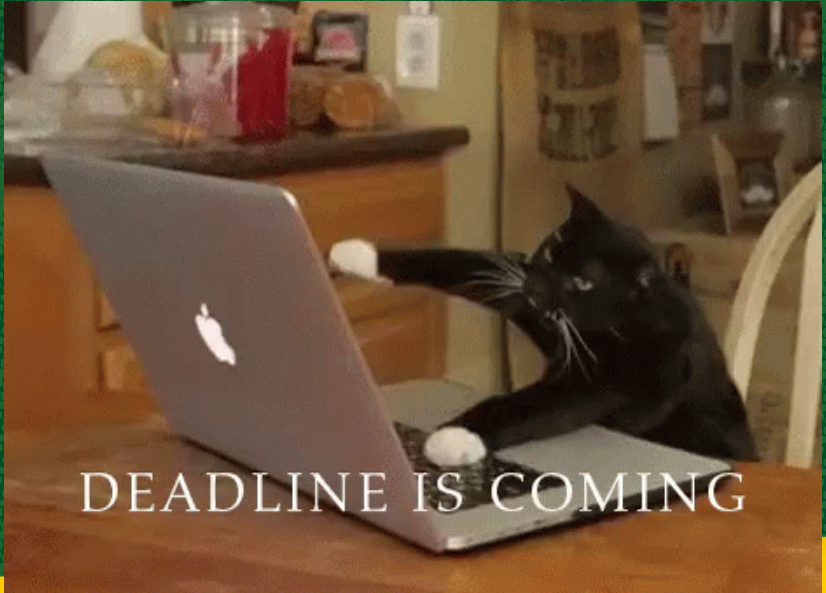
"variables": {
  "finish": {
    "reference": "coins",
    "op": "■"
  }
},

"rd": {
  "variables": {
    "rank": {
      "reward": 1.0
    }
  }
},

"ons": [
  ["UP", "DOWN"],
  ["LEFT", "RIGHT"],
  ["A", "B"], ["Y"], ["START"], ["R"], ["R", "A"],
  ["B"], ["B", "Y"], ["R", "A", "B"], ["R", "A", "Y"],
  ], ["A", "Y"]
]

```

So Let's Make Something Work



So Let's Make Something Work

Goals

- ▶ Get a program to work nicely with retro.

So Let's Make Something Work

Goals

- ▶ Get a program to work nicely with retro.
- ▶ Enable easy parameterisation of variables.

So Let's Make Something Work

Goals

- ▶ Get a program to work nicely with retro.
- ▶ Enable easy parameterisation of variables.
- ▶ Test test test

Let's see some games!

(Airstriker Genesis)



Figure: Airstriker Genesis Random: Max 160



Figure: Airstriker Genesis Longer Run: Max 940

Let's see some games!

(Donkey Kong Country)



Figure: Donkey Kong Country with lookahead 100, 10x

Let's see some games!

(Super Mario Brothers)



Figure: Super Mario Bros, low training at 10x



Figure: Super Mario Bros, a weekend of training at 10x