

# 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.

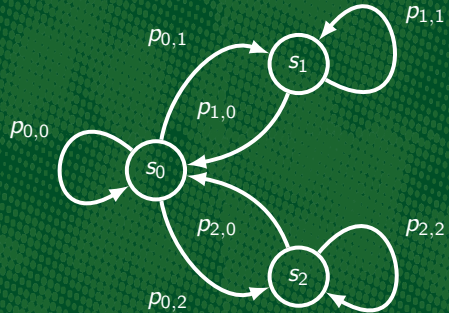


Figure: Sample Markov Decision Process

# 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)

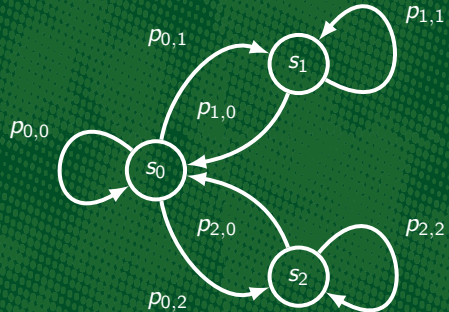


Figure: Sample Markov Decision Process

# 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
$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$
  - $\alpha$  = learning rate
  - $r$  = reward
  - $\gamma$  = discount

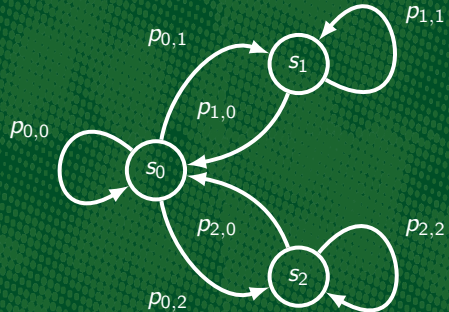


Figure: Sample Markov Decision Process



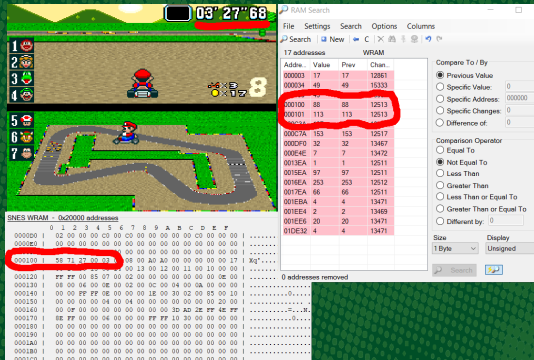
# Framework

- ▶ 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).



# Integration

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

The screenshot shows a Super Mario Kart race in progress. The timer at the top right indicates 03' 27" 68. Four black arrows point from the timer's digits to specific RAM addresses in the BizHawk RAM Search window:

- Arrow from '0' to address 000003 (value 17)
- Arrow from '3' to address 000034 (value 49)
- Arrow from '2' to address 000100 (value 88)
- Arrow from '7' to address 000101 (value 113)

The RAM Search window shows a list of 17 addresses in WRAM. The values for the addresses pointed to by the arrows are highlighted with a red box:

Address	Value	Prev	Chan.
000003	17	17	12861
000034	49	49	15333
000100	88	88	12513
000101	113	113	12513

Below the RAM Search window, the SNES WRAM memory dump is visible. The address 000100 is highlighted with a red box, showing the value 58 71 27 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": 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    "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        ["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    }
26 }
```

# 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    "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      [
19        ["UP"], ["DOWN"],
20        ["LEFT"], ["RIGHT"],
21        ["A"], ["B"], ["Y"], ["START"], ["R"], ["R"], ["A"],
22        ["R", "B"], ["B", "Y"], ["R", "A", "B"], ["R", "A", "Y"],
23        ["A", "B"], ["A", "Y"]
24      ]
25    }
26 }
```

- ▶ Need to collect (used BizHawk)
- ▶ Convert Hex to add rambase number integrate 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)

reward: xscroll



Figure: Super Mario Bros, low training at 10x



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

# Questions?

(Let's look at code)