Joint RL meeting

Gridworld implementation of Olivia's task

Andrea Pierré

January 30th, 2023

Brown University

Outline

1. Implementation
2. Issues along the road
3. Results
4. Main differences with Niloufar’s model

Outline

1. Implementation
2. Issues along the road
3. Results
4. Main differences with Niloufar's model

Implementation

- RL concepts abstracted in high level functions :
  - `reset()`: reset the environment at the end of the episode
  - `reward()`: define in what conditions the agent get a reward and how much reward it gets
  - `is_terminated()`: define when the end of the episode has been reached
  - `step()`: execute the defined action in the current state

```
new_state, reward, done = env.step(action, state)
```

Implementation

- RL concepts abstracted in high level functions :
  - `reset()`: reset the environment at the end of the episode
  - `reward()`: define in what conditions the agent get a reward and how much reward it gets
  - `is_terminated()`: define when the end of the episode has been reached
  - `step()`: execute the defined action in the current state

```
new_state, reward, done = env.step(action, state)
```

Implementation

- RL concepts abstracted in high level functions :
  - `reset()`: reset the environment at the end of the episode
  - `reward()`: define in what conditions the agent get a reward and how much reward it gets
  - `is_terminated()`: define when the end of the episode has been reached
  - `step()`: execute the defined action in the current state

```
new_state, reward, done = env.step(action, state)
```

Implementation

- RL concepts abstracted in high level functions :
  - `reset()`: reset the environment at the end of the episode
  - `reward()`: define in what conditions the agent get a reward and how much reward it gets
  - `is_terminated()`: define when the end of the episode has been reached
  - `step()`: execute the defined action in the current state

```
new_state, reward, done = env.step(action, state)
```

Implementation

- RL concepts abstracted in high level functions :
    - `reset()`: reset the environment at the end of the episode
    - `reward()`: define in what conditions the agent get a reward and how much reward it gets
    - `is_terminated()`: define when the end of the episode has been reached
    - `step()`: execute the defined action in the current state

```
new_state, reward, done = env.step(action, state)
```

Implementation

- Each step, the agent gets a composite observation:

location	cue
{0,...,24}	North light
	South light
	Odor A
	Odor B

- Convenience functions to translate the movements between the grid positions and the states

Implementation

- Each step, the agent gets a composite observation:

location	cue
{0,...,24}	North light
	South light
	Odor A
	Odor B

- Convenience functions to translate the movements between the grid positions and the states

Implementation

- Wrapper environment to translate the human readable environment (composite state) into a suitable environment for the Q-learning algorithm (flat state)

```
state = {"location": 13, "cue": LightCues.North}
env.convert_composite_to_flat_state(state)
# => 13
```

```
state = 63
env.convert_flat_state_to_composite(state)
# => {"location": 13, "cue": <OdorID.A: 1>}
```

- Human readable objects

```
action = 0
Actions(action).name
# => "UP"
```

Implementation

- Wrapper environment to translate the human readable environment (composite state) into a suitable environment for the Q-learning algorithm (flat state)

```
state = {"location": 13, "cue": LightCues.North}
env.convert_composite_to_flat_state(state)
# => 13
```

```
state = 63
env.convert_flat_state_to_composite(state)
# => {"location": 13, "cue": <OdorID.A: 1>}
```

- Human readable objects

```
action = 0
Actions(action).name
# => "UP"
```

Outline

1. Implementation
2. Issues along the road
3. Results
4. Main differences with Niloufar's model

Not enough states to solve the task

Pre odor - North light

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Pre odor - South light

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

Post odor - Odor A

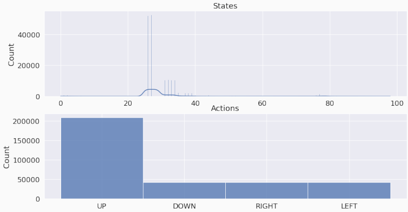
50	51	52	53	54
55	56	57	58	59
60	61	62	63	64
65	66	67	68	69
70	71	72	73	74

Post odor - Odor B

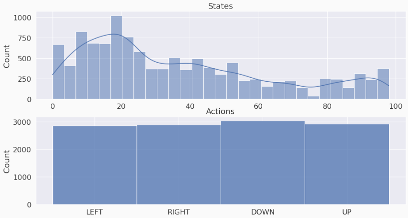
75	76	77	78	79
80	81	82	83	84
85	86	87	88	89
90	91	92	93	94
95	96	97	98	99

ϵ-greedy when Q-values are identical

Vanilla ϵ-greedy

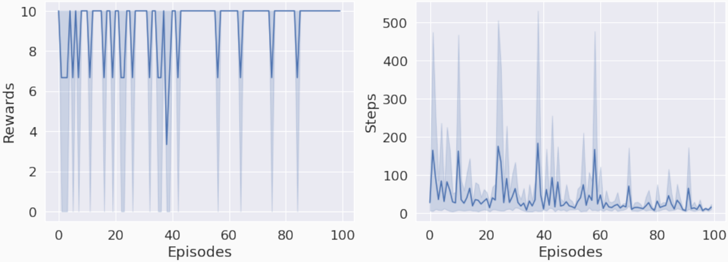


Randomly choosing actions with the same Q-values

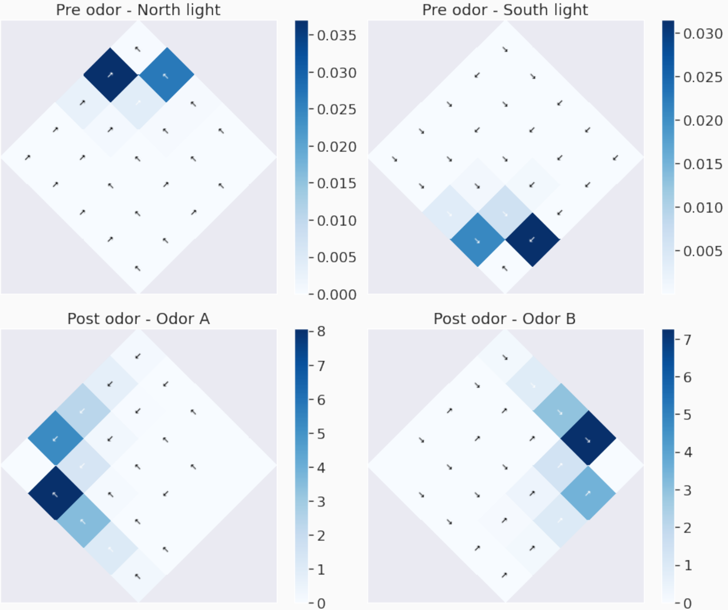


Outline

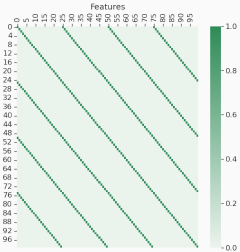
Standard Q-learning – allocentric environment

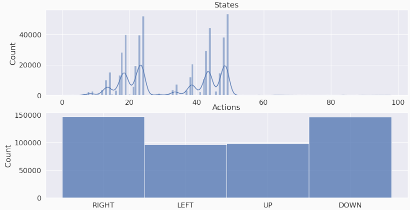


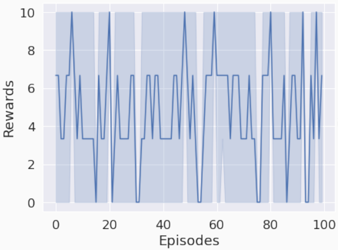
Standard Q-learning – allocentric environment

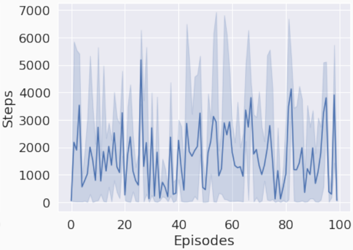


Q-learning with function approximation – allocentric environment – without joint representation

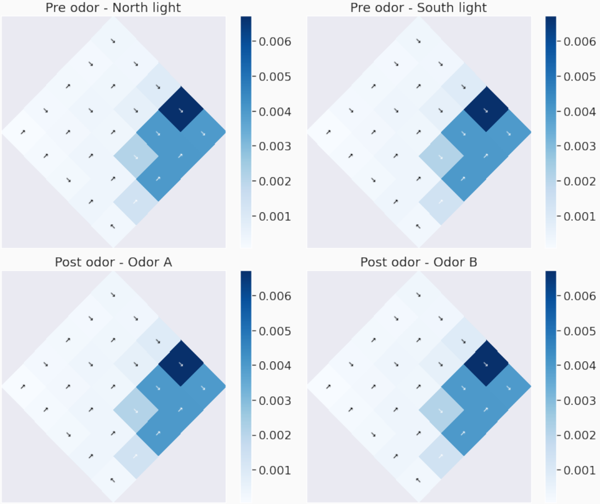




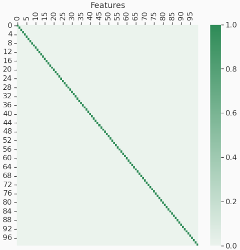


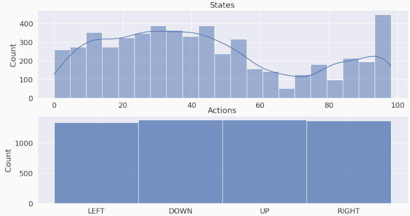


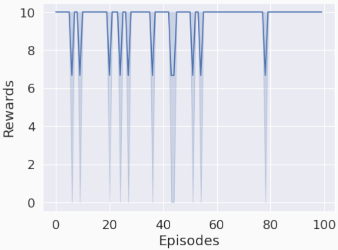
Q-learning with function approximation – allocentric environment – without joint representation

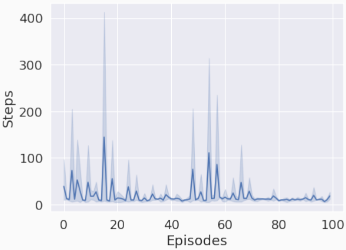


Q-learning with function approximation – allocentric environment – with joint representation

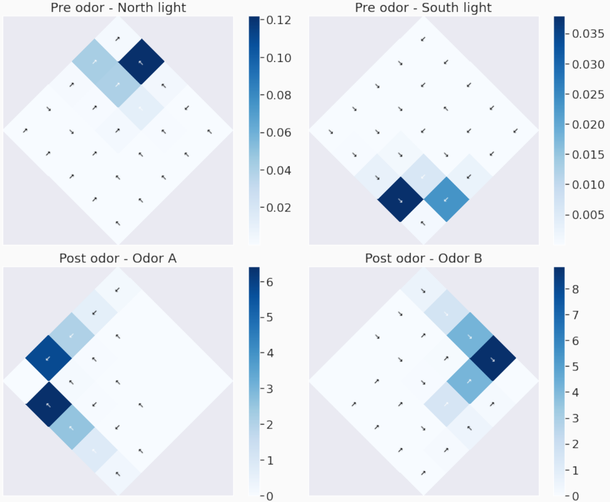




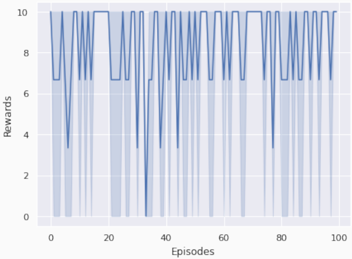


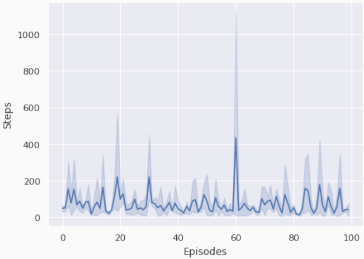


Q-learning with function approximation – allocentric environment – with joint representation



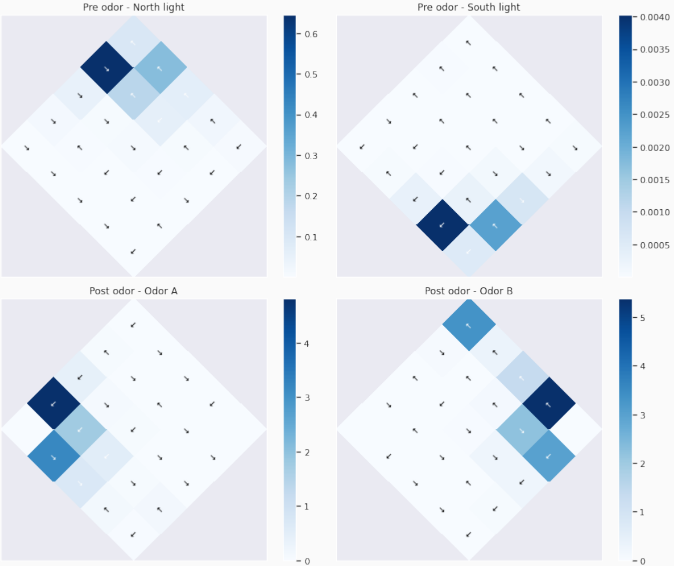
Standard Q-learning – egocentric environment





$\rightarrow$ Agent not learning (yet)

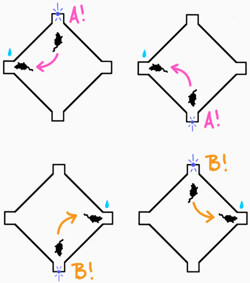
Standard Q-learning – egocentric environment

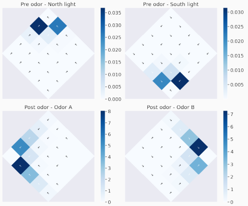


Outline

Main differences with Niloufar's model

- The environment is closer to the real experiment $\rightarrow$ ports are in the corners of the arena, not in the middle of the walls
- Code is clean, readable, and abstracted in high level functions/concepts





Questions ?