Lab meeting

Deep dive into deep RL

Andrea Pierré

February 12th, 2024

Brown University

Outline

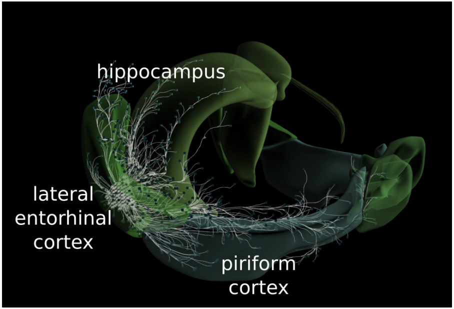
1. Context
2. Deep RL algorithm & building blocs
3. Results
4. Future directions

Outline

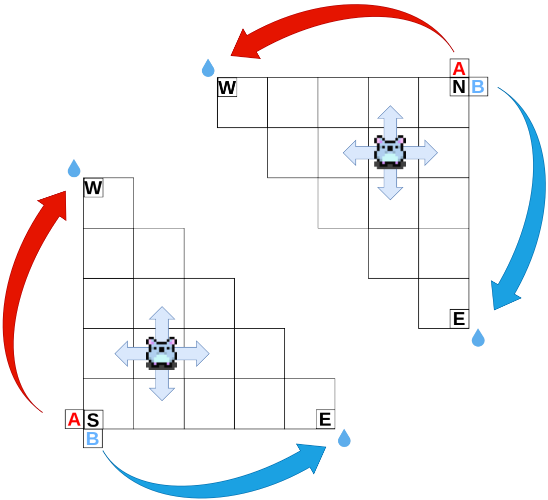
1. Context
2. Deep RL algorithm & building blocs
3. Results
4. Future directions

How do neurons in the LEC integrate sensory and spatial information?

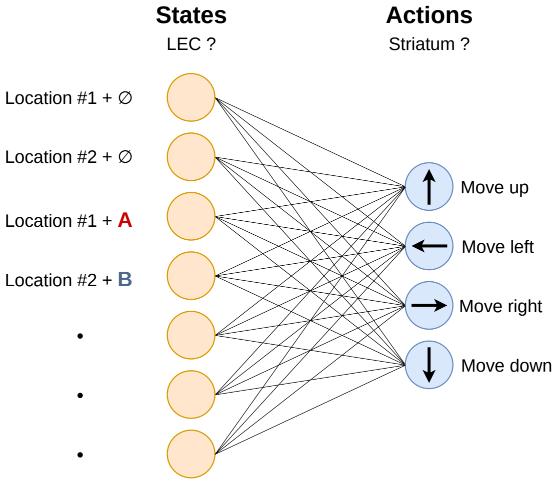
- Piriform Cortex encodes olfactory information
- Hippocampus encodes spatial information
- Lateral Entorhinal Cortex (LEC) encodes both olfactory & spatial information



The half-triangle task



Mapping states to action

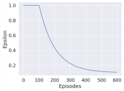


Outline

1. Context
2. Deep RL algorithm & building blocs
3. Results
4. Future directions

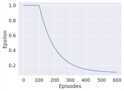
Deep RL lessons learned

- Deep RL is different from supervised learning → the data to optimize on is not fixed (moving target)
- DQN tricks:



Deep RL lessons learned

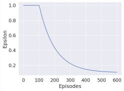
- Deep RL is different from supervised learning → the data to optimize on is not fixed (moving target)
- DQN tricks:
    - Replay buffer → save the data experienced in a buffer and sample from it to break the temporal correlation of the data
    - Exploration warm up with ϵ-greedy → experience more diverse data
    - Batching → update the weights of the network based on several data examples at the same time instead of only one
    - Target network → use 2 networks to stabilize learning (one of them is updated with a lag)



Deep RL lessons learned

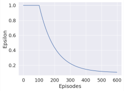
- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)

- DQN tricks:
    - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
    - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
    - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
    - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



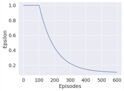
Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
  - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
  - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
  - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
  - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



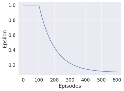
Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
  - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
  - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
  - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
  - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



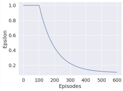
Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
  - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
  - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
  - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
  - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



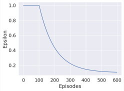
Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
  - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
  - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
  - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
  - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



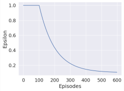
Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
    - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
    - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
    - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
    - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



Deep RL lessons learned

- Deep RL is different from supervised learning $\rightarrow$ the data to optimize on is not fixed (moving target)
- DQN tricks:
  - Replay buffer $\rightarrow$ save the data experienced in a buffer and sample from it to break the temporal correlation of the data
  - Exploration warm up with ϵ-greedy $\rightarrow$ experience more diverse data
  - Batching $\rightarrow$ update the weights of the network based on several data examples at the same time instead of only one
  - Target network $\rightarrow$ use 2 networks to stabilize learning (one of them is updated with a lag)



Deep RL implementation

Algorithm 1: Deep Q-Network (DQN)

```
Initialize replay memory D to capacity N
Initialize action-value network Q with random weights θ
Initialize target action-value network Q̂ with random weights θ⁻ = θ
for episode ← 1 ... M do
    state ← reset(env)
    done ← False
    while done ≠ True do
        Q ← forward_pass(state)                               /* 4 action values vector */
        action ← ε_greedy(action_space, state, Q)
        state_new, reward, done ← env.step(action, state)
        Store transition (state, action, reward, next_state, done) in D
        Sample random minibatch of transitions from D
        Q ← forward_pass(state_new)                           /* 4 action values vector */
        Q_new ← reward + γmax(Q̂)                              /* scalar */
        y ← max(Q)                                            /* scalar */
        if done = True then
            ŷ_pred ← reward                                   /* scalar */
        else
            ŷ_pred ← Q_new                                    /* scalar */
        end
        Loss ← (y − ŷ_pred)²
        Perform a gradient descent step on Loss with respect to the network parameters θ
        Every C steps reset Q̂ = Q
    end
end
```

Inspired from (Mnih et al., 2015)

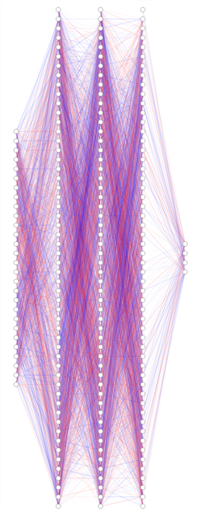
Outline

1. Context
2. Deep RL algorithm & building blocs
3. Results
4. Future directions

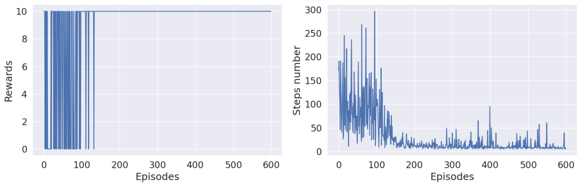
How deep RL feels like

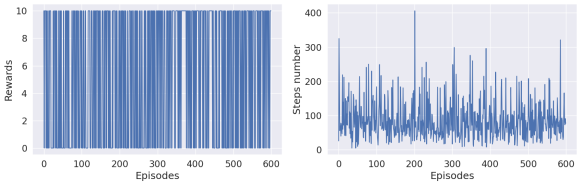
Networks architectures

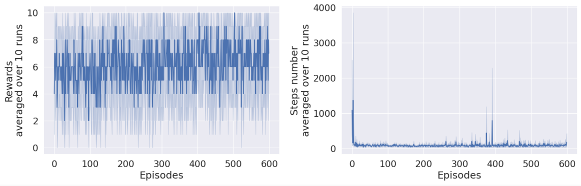
$$28 \to 54 \to 54 \to 54 \to 4$$



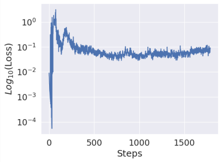
Rewards & steps

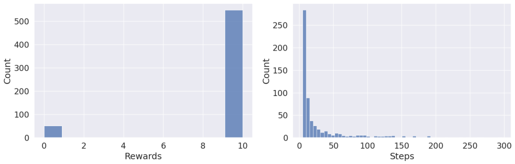


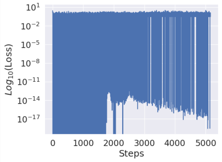


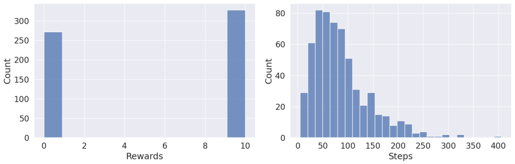


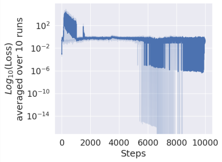
Loss, rewards & steps distributions

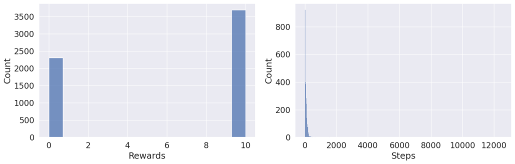




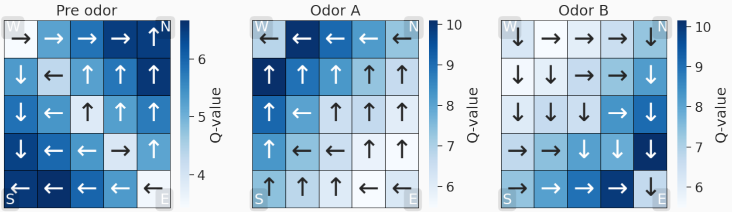








Policy learned (when it works)



Outline

Adding memory into the task

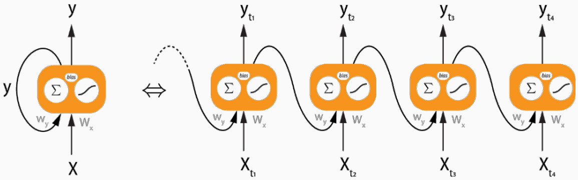
Current environment

step	location	cue	reward
1	2	No odor	0
2	3	No odor	0
3	4	Odor A	0
4	3	Odor A	0
5	2	Odor A	0
6	1	Odor A	0
7	0	Odor A	10

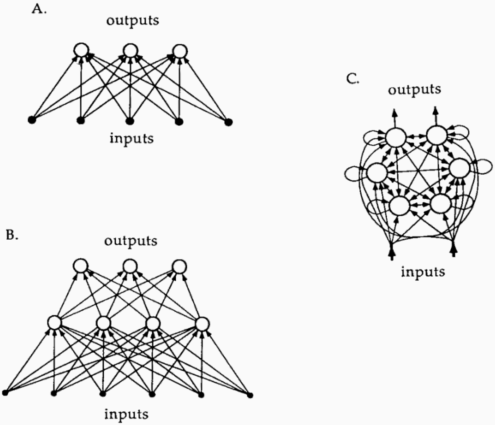
With memorization needed

step	location	cue	reward
1	2	$\emptyset$	0
2	3	$\emptyset$	0
3	4	Odor A	0
4	3	$\emptyset$	0
5	2	$\emptyset$	0
6	1	$\emptyset$	0
7	0	$\emptyset$	10

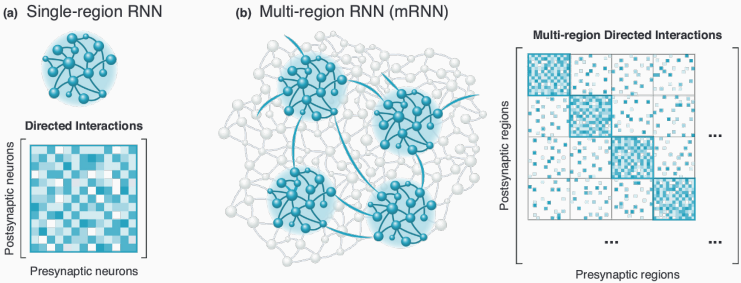
RNN for memorization and sequence modeling



Feedback connectivity



Network of networks



(Perich and Rajan, 2020)

Questions ?