RL Algorithm : Update 01.06

Decisions :

- Discrete Action Space
- Discrete Observation Space ? (distance can be segmented in Intervals)
- Offline Learning
- Q-Family Algorithms

Observations :

Positive :

- Discrete Action space offers the possibility to use the Q-Algorithms family
- Q-Family Algorithms : one of the best documented Algorithm-Families with examples
- Offline Learning allows us to avoid some Hardware challenges :
    - Related to real time communication between Learning Algorithm and agent
      → It is crucial to get a response from the Learning Algorithm in Time in order for the learning process to work

      → could have been avoided by running the learning algorithm on the RPI, but Computational power will probably not be sufficient.

Negative :

- Discrete Action space makes for less flexibility for the learning process. *(intuitive)*
- Offline Learning : Q-Learning is based on a Bootstrapping :

Bootstrapping :

Bootstrapping refers to the method where an estimate is updated based on other estimates. In the context of Q-learning and DQN, this involves updating the Q-value of a state-action pair using the estimated Q-value of the subsequent state-action pair. The Q-learning update rule is:

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

Here, $\max_{a'} Q(s',a')$ is the bootstrap estimate, which is used to update the current Q-value.

Why it works in the context of online learning ?

- In an Online Context **Bootstrapping** works well, since the chosen action is immediately executed and feedback about the reward is received.
  The estimation of the subsequent action (max a′) is corrected from real Data in real time.

Extrapolation Error:

- In offline reinforcement learning, the agent learns from a fixed dataset of experiences without further interaction with the environment. If the fixed dataset does not adequately cover the state-action space, the Q-network may be forced to make predictions on state-action pairs it has never seen before.
- Bootstrapping can amplify these errors because the Q-values are updated based on other estimated Q-values, which may themselves be inaccurate. This can lead to significant extrapolation errors where the model makes highly inaccurate predictions about unseen states and actions.

Overestimation Bias :

- Q-learning is prone to overestimation bias, where the Q-values tend to be higher than the true expected rewards. In an offline setting, this problem can be exacerbated because the lack of new data means there is no opportunity for the model to correct these overestimations.

Why is Q-learning prone to overestimation Bias ?

- Bootstrapping inherently involves taking the maximum Q-value of the next state, which can further propagate and amplify any overestimated Q-values.

Possible Solutions :

Problem related to Discrete Action Space :

→ This should be addressed with a large enough set of discrete possible actions
→ Here the design of the Action Space is gonna be key for the learning process o work well
→ Will probably be resolved by trial and error
→ First Iteration would probably be a moderate big Action Space that is going to offer insight about its design (we can segment it more or less)

Problem related to Offline Learning :

BCQ :

- This solution was presented in the Paper titled *"Off-Policy Deep Reinforcement Learning without Exploration"* from *Scott Fujimoto, David Meger and Doina Precup*

Main Goal of the Algorithm :

- It aims to avoid the totally random estimation of *Q-Values* of *State,Action* pairs that do not appear in the *training Data Set*
- Apply Constraints to the Learning Agent so that it only tries to evaluate Actions that are present in the *training Data Set* or similar actions.

How does it achieve its goal ? *(Broadly)*

Generative Model for Action Selection:

- BCQ uses a state-conditioned generative model to generate plausible actions similar to those in the dataset. This generative model, typically a conditional variational auto-encoder (VAE), is trained to approximate the distribution of actions in the batch.

Action Perturbation:

- A perturbation model is used to adjust the generated actions slightly, allowing the policy to explore actions within a small range around those generated by the VAE.

- *What is the problem with BCQ ?* + Decision regarding BCQ
    - Problem : The Implementation of the conditional VAE can be tricky and is probably out of our Capabilities-Scope
    - Decision : We decided to drop this Solution

- **Offline DQN + Creation of a Large enough Dataset:**
    - As presented, the Problem of DQN in offline training is rooted in the possible Lack of Data in the *Training Data Set*
    - In our “simple” Problem-Setting, we estimate that is possible to create a large enough *Training Data Set*, that could minimize the Bad effect of **bootstrapping** by covering a relatively big percentage of the (*State,Action)* pairs.
    - ***Challenges :***
        - *Implementation Challenge :* We will have to adapt DQN to offline training.
        - *Data Set creation :* In order for the *training Data Set* to be large and varied enough, we should preferably use manual exploration (manually control the Robot in the Data Collection phase)
          → We should watch Memory problems (we might need to store the Data directly on a Server)
          → This adds a Step in our Process : provide an UI to control effectively the robot in order to Collect Data manually