RL Algorithms comparison | Yessmine

Requirements:

offline

easy to implement

focus on discrete (Reward Function)

Model-Free RL:

- Involves methods that learn to act without needing a model that predicts the environment's behavior.
- Contrasts with model-based RL, which builds a model of the environment to simulate and plan actions.

On-Policy RL:

- **Learning Approach**: Improves the policy that is currently being used.
- **Examples**: SARSA, on-policy actor-critic methods.
- **Advantages**: Simplicity, direct policy improvement.
- **Disadvantages**: Less sample-efficient, challenging exploration-exploitation trade-off.

Off-Policy RL:

- **Learning Approach**: Improves a different policy from the one used to generate data.
- **Examples**: Q-learning, DQN.
- **Advantages**: More sample-efficient, flexible exploration.
- **Disadvantages**: More complex, potential stability issues.

Zied's Research Outcome

Algorithms that are useful for offline learning with Discrete action and observation spaces, that are Model free , and off policy (exploration),

offline DQN, QR-DQN and Batch-Q Learning
*
QR-DQN usually shows better results but is a bit more complex to implement.

Implementation :
-Like online algorithms but has to be adapted (still to research but seems doable)

Yin's Research Outcome

BCQ

▼ ~~Tool D4RL: Algo evaluation Tool~~

Rich Baselines: D4RL provides a comprehensive set of baselines, including common offline algorithms like BCQ, BEAR, BRAC, and others.

Evaluation

• BCQ is an advanced offline RL algorithm specifically designed to mitigate issues inherent in BQ, such as overestimation and extrapolation errors.

BQ as subcategory of DQN.

- If you have a relatively simple problem and want a straightforward implementation, you might start with Offline DQN or Batch Q-Learning. → Also the easiest
  - Offline DQN: can provide a solid foundation in RL concepts, including value-based methods, experience replay, and neural network approximation.
  - **Batch Q-Learning:** can help you understand the basics of Q-learning algorithms and how they operate without the added complexity of deep neural networks.
- If capturing uncertainty in Q-values is crucial for your application, QR-DQN**(Quantile Regression DQN)** could be a good choice.

- If you're dealing with complex environments and want a method explicitly designed for offline RL with improved exploration strategies, BCQ**(Batch-Constrained Q-learning)** might be worth exploring.

→ BQ, BCQ or QR-DQN.

Decision

BQ

Q-Learning: (steps)

1. **Initialize** the Q-function arbitrarily (e.g., all zeros).
2. **Observe** the current state s.
3. **Select** an action a using an exploration policy (e.g., ε-greedy).
4. **Take** the action and observe the next state s' and the reward R_{t+1}.
5. **Update** the Q-value using the Bellman equation:

   $$Q(s,a) \leftarrow Q(s,a) + \alpha [R_{t+1} + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

   where α is the learning rate.
6. **Repeat** steps 2-5 for each time step and episode until convergence.