A General View of Offline Reinforcement Learning Algorithms

Chen Kang Ming

School of Computer Science and Engineering

Supervised by: Asst Prof Jun Zhao

School of Computer Science and Engineering

Abstract **-** The advancements in offline reinforcement learning could benefit our daily lives and improve how we train new agents, especially in high-risk or restricted environments. This report aims to give the reader an overview of the current well-known offline reinforcement learning algorithms: Batched Constrained Q-Learning, Bootstrapping Error Accumulation Reduction, Ensemble Deep Q-Network (Ensemble DQN), Random Ensemble Mixture (REM) among many others. This report will also discuss the efficiency of the algorithms against real-world benchmarks in the "Datasets for Deep Data-Driven Reinforcement Learning" tests, where applicable. However, each algorithm is not without its limitations. This report also aims to highlight to the reader the roadblocks in the algorithms when facing different types of data, and challenges in the field of offline reinforcement learning in general. The analysis will end off with the proposal of future considerations and suggestions in developing more robust offline reinforcement learning algorithms.

Keywords - Offline Reinforcement Learning, Batch RL, off-policy, REM, BEAR, BCQ, D4RL

1: INTRODUCTION

Reinforcement learning algorithms are instructions that tell the agent how to derive the optimal actions to take within a stochastic environment in which actions are probabilistic. In the algorithm, the reward function, which outputs an indicator informing the agent of its reward (or penalty) at a certain stage, instructs the agent on its next actions within the environment (Levine et al., 2020) . This then determines the best policy, which the agent will undertake in the course of running the algorithm.

In the last few years, we have seen a recent shift of focus towards offline reinforcement learning, otherwise known as *batch reinforcement learning , off-policy reinforcement learning or data-driven reinforcement learning*. In offline reinforcement learning, the agent is only exposed to collected data from past iterations of the same task, and is then required to derive an optimal solution and policy from that, devoid of interaction with any new or future environments. Figure 1 (Kumar, 2019) illustrates the general difference between reinforcement learning and offline reinforcement learning as mentioned above. Inroads in similar concepts within machine learning are not foreign - data-driven machine learning has seen interest in the region of supervised learning as well.

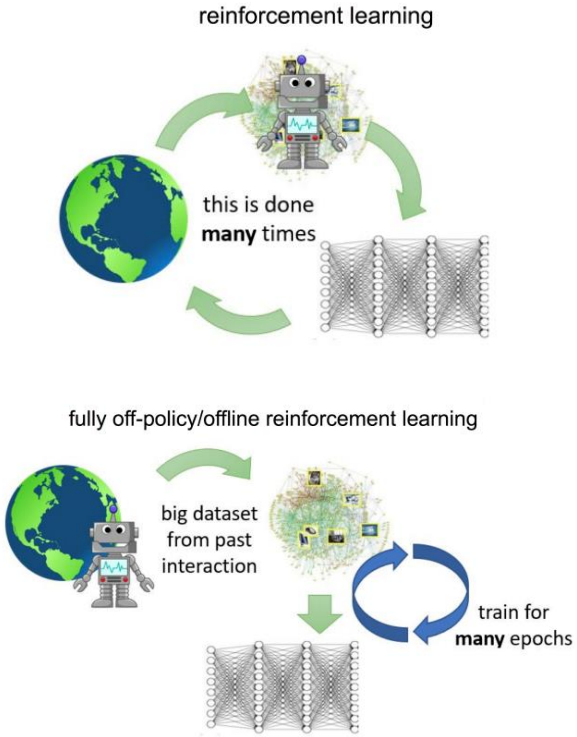

Figure 1: Difference between reinforcement learning (top) and offline reinforcement learning (bottom)

Offline reinforcement learning has seen use in "Datasets for Deep Data-Driven Reinforcement Learning", also known as D4RL (Fu et al., 2020). D4RL provides a platform for various offline reinforcement learning algorithms to gauge their performance under different tasks, like in a kitchen with various tools (*FrankaKitchen*) , in vehicular navigation (*CARLA*) and path solving (*AntMaze*) amongst many others. Furthermore, testing of offline reinforcement learning algorithms have also extended to the "Atari 2600" and "MuJoCo" benchmarks as well (Agarwal et al., 2019). In real-world situations, offline reinforcement learning would also benefit use cases where interaction with the environment contains huge risk, or is generally

tedious to collect (too large an environment, international boundaries and agreements etc.

This report summarises the current literature of the topic on offline reinforcement learning, focusing on the algorithms that are currently reported, tested and finally, to give opinions, critiques and suggestions on future work within offline reinforcement learning.

2: MAIN ISSUES OF OFFLINE REINFORCEMENT LEARNING

Offline reinforcement learning faces one main issue - that is for the policy to generalise well to most if not all use cases of the scenario in the environment, given a fixed dataset. In typical reinforcement learning, given regular exposure to the environment and its changes, it would be then easier to learn a constantly updating environment. In offline reinforcement learning however, this is not the case. Off-policy methods like the Soft Actor-Critic algorithm (SAC) struggle in giving an accurate fitting with an optimal policy, as shown in Figure 2. Furthermore, increasing the size of datasets also does not necessarily improve the performance - as referenced in Figure 2 (Yellow line as opposed to the green line) (Kumar, 2019). HalfCheetah in this and subsequent graphs is a simulation environment that aims to train a quadruped cheetah agent to be able to run, through the user's reinforcement learning algorithm of choice.

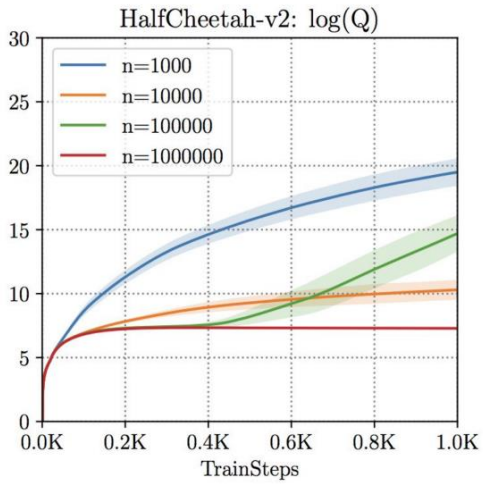

Figure 2: Logarithmic graph of SAC performance measured by Q-value, with varying dataset sizes

Other issues include error accumulation due to the presence of out-of-distribution (OOD) actions. With every action being backed up, sub-optimal and mediocre policies are collected in deriving an optimal best-case policy, and the backed up data skews the final policy as well. In the process of policy learning, the agent may take a different method compared to the method used within the data itself. This would mean that rewarding the agent becomes harder and more ambiguous too.

In this paper, well-known offline reinforcement learning algorithms will be discussed. This includes Batched Constrained Q-Learning (BCQ), Bootstrap Error Reduction (BEAR), Ensemble DQN and Random Ensemble Mixture (REM) These algorithms seek to give a solution to the problem within offline reinforcement learning and to circumvent any inaccuracies brought about by the nature of offline reinforcement learning. However, each algorithm has its weakness, which will also be highlighted below.

3: OFFLINE REINFORCEMENT LEARNING ALGORITHMS

3.1: Batched Constrained Q-Learning (BCQ)

BCQ is a dynamic programming algorithm that aims to reduce extrapolation error, by way of introducing fine-tuning to the policies derived by the algorithm. The agents are "trained to take actions" where (1) distance of actions to the batch are as close as possible to each other, (2) where "familiar data" can be found within the batch, (3) and logically to maximise value for each policy (Fujimoto et al., 2019). In BCQ, (1) is heavily prioritised - one which hence will aid in reducing said extrapolation error from the policy. The inspiration for BCQ comes from the shortcomings of DQN and Deep Deterministic Policy Gradient (DDPG): where they are unable to learn effectively when data is not correlated to the current distribution, which hinders the very motivation of offline reinforcement learning.

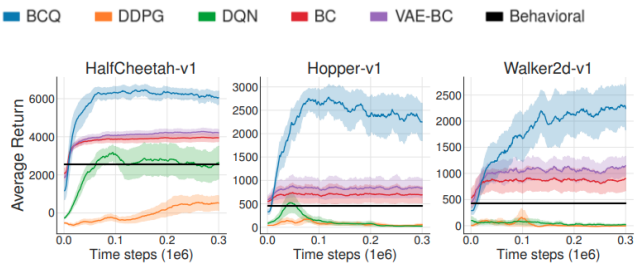

Figure 3: Imperfect demonstration performance of various algorithms against a baseline behavioural policy

In Figure 3, as extracted from Fujimoto et al., the imperfect demonstration performance shows that BCQ outperforms all other DQN- and DDPG-based algorithms, as well as some forms of behavioural

cloning. “Imperfect demonstration” here refers to training the agent using data collected by an expert policy, with introduction of noise (Fujimoto et al., 2019). However, one issue faced by BCQ is that it is entirely policy-dependent- after all, the constraints on the algorithm are determined by the policy itself. Given a relatively suboptimal or low-return policy, BCQ tends to perform poorly (Kumar, 2019). However, in the presence of undirected data, like in *CARLA* in D4RL, BCQ does better compared to BEAR and REM which will be discussed in the later sections.

3.2: Bootstrapping Error Accumulation Reduction (BEAR)

BEAR, like BCQ, aims to solve the issue of “out-of-distribution” (OOD) errors. However, BEAR introduces “sampled Maximum Mean Discrepancy” (MMD) as a support constraint within the algorithm (Kumar, 2019). The main motivation of BEAR is to allow for the policy to be as close to the support constraint as possible, even in the presence of OOD errors- however, only requiring the policy to place “non-zero probability mass on actions with non-negligible behaviour policy density”. In other words, actions that are taken by agents that cannot be ignored, have to be considered (hence the “non-zero probability mass”) amidst trying to fit to the support constraint. Figure 4 illustrates the MMD formula (Kumar, 2019).

$$\mathrm{MMD}^2(X,Y) = \frac{1}{n^2}\sum_{i,i'} k(x_i, x_{i'}) - \frac{2}{nm}\sum_{i,j} k(x_i, y_j) + \frac{1}{m^2}\sum_{j,j'} k(y_j, y_{j'}).$$

Where $X = \{x_1, \ldots, x_n\}$
$Y = \{y_1, \ldots, y_n\}$
And k represents any radial basis function (RBF) kernel

Figure 4: MMD formula illustration, with legend

In testing done by Kumar (2019), BEAR has returned stellar performance compared to other behavioural cloning or dynamic programming methods from low- to high-return policies. In particular, BEAR almost matches the performance of a standard naive reinforcement learning (naive RL) in a low-return policy scenario, and in an average dataset, outperforming all other algorithms tested, as shown in Figure 5 and 6 (Kumar, 2019).

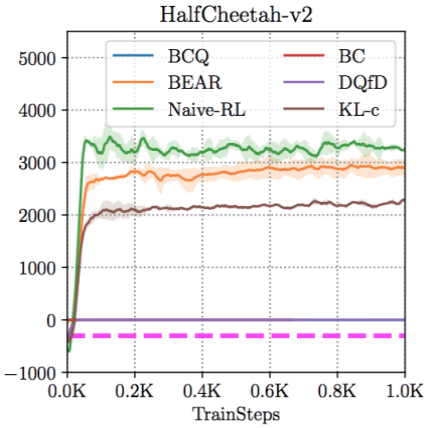

Figure 5: BEAR in a low-return random policy, compared to other algorithms and naive RL and also notably BCQ as discussed above.

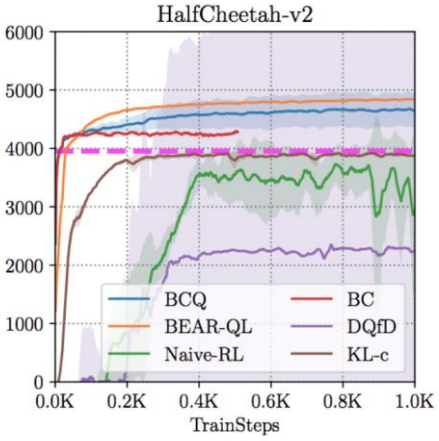

Figure 6: BEAR in a medium-quality dataset performs better than all other algorithms, even Naive RL.

However, BEAR does not have strong performance in an environment with undirected data, such as *CARLA* and *FrankaKitchen* in the D4RL datasets (Fu et al., 2020). This refers to the agents performing actions that are not integral towards the final goal and reward-giving stage. This is a weakness also shared with the REM algorithm, which will be discussed in a later part of the report.

3.3: Ensemble Deep Q-Networks (Ensemble DQN)

To understand Ensemble DQN, we must first explore DQN. DQN is a combination of two concepts: standard Q-Learning and a reinforcement learning algorithm in the presence of deep neural networks (TensorFlow, 2021). DQN has proven to be a good-performing algorithm, especially in the realm of games like in the Atari 2600 benchmark.

Ensemble DQN is essentially an extension of DQN, which now considers multiple Q-functions and their

corresponding targets and hence, utilises the average within the Q-value-estimates (Figure 7) (Agarwal et al., 2019).

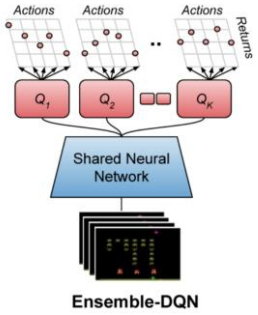

Figure 7: Visual on Ensemble-DQN.

3.4: Random Ensemble Mixture (REM)

In REM, we consider an extension from the ensemble-DQN method. Using the same principle of multiple Q-functions to give a Q-value estimate, REM proposes that the convex combination of the Q-value estimates is then the final Q-value estimate, and using that to train the agent. Figure 8 illustrates this concept.

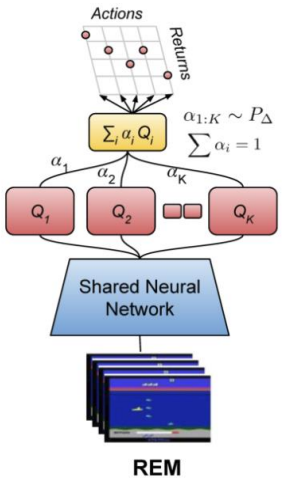

Figure 8: Illustration of REM concept, where a combination of Q-value estimates is used to determine the actions and returns

The concept draws motivation from dropout - where one “drops out” certain values with a certain probability, to prevent overfitting of the current distribution (Srivastava et al., 2014). In REM, each training step uses multiple Q-value estimates probabilistically to ensure optimal performance, and reduce the overfitting as mentioned above.

REM however is currently formulated to tackle problems with discrete action spaces. In D4RL for example, the focus is on problems with continuous states. During the D4RL algorithm performance comparison, a continuous action version (which is currently marked as not functional) (Agarwal et al., 2019)(Agarwal, 2019) was utilised; however, there were no results tabulated from the study.

4: ANALYSIS AND SUGGESTIONS

Discussing the various algorithms, one can easily see the unique feature that each algorithm has to facilitate offline reinforcement learning. For example, BEAR has MMD as a support constraint; REM uses dropout, and Ensemble DQN utilises a combination of Q-values generated. One potential suggestion would be to combine two ideas into one algorithm - theoretically, for example, one could explore the option of the combined effects of the concepts of DQN and BEAR, for example. Many support constraints can be created for many different runs of the same policy, and the combination of these support constraints could be used in deriving the optimal policy in the problem.

Other algorithms not discussed here present differing solutions to the problems as mentioned in the summaries of the algorithms discussed. For example, Constrained Q-learning (CQL) aims to tackle the issue of OOD actions by introducing a lower bound for Q-values (Kumar et al., 2020). Positively, it also fares well in D4RL, being the best-performing algorithm in *AntMaze, Adroit* and even *FrankaKitchen*, outperforming BEAR and SAC (Fu et al., 2020). CQL shows great promise within the realm of offline RL, and should be explored further.

In D4RL, we are concerned with the performance of an algorithm in continuous action problems, which is more aligned with real-world situations. Future algorithms should heavily consider a focus on its feasibility in such situations, in order to benefit roadblocks in current implementations of offline RL, and to scale it to be used in applications all around the world.

5: CONCLUSION

Offline reinforcement learning shows huge promise in recent years to match the performance of conventional reinforcement learning methods. Although further optimization is required for certain facets of implementation, like how to take care of undirected data, offline RL remains an exciting realm of supervised learning to follow and observe. In future, industry sectors like the medical and engineering sectors as well as autonomous driving

could look to tap into the power of offline RL for their use cases.

5: ACKNOWLEDGEMENTS

I would like to thank Assistant Professor Zhao Jun for giving me the opportunity to be a part of this URECA research project. The project has broadened my horizons with regards to offline RL as well as the fundamental concepts of supervised learning and reinforcement learning. I would also like to thank Wenhan and the other PhD students for lending their support and guidance for my project. They were patient and gave resourceful ideas for the project, which I am thankful for.

REFERENCES

Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Kumar, A. (2019, December 5). Data-Driven Deep Reinforcement Learning. The Berkeley Artificial Intelligence Research Blog. https://bair.berkeley.edu/blog/2019/12/05/bear/

Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2020). D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219.

Agarwal, R., Schuurmans, D., & Norouzi, M. (2019). An Optimistic Perspective on Offline Reinforcement Learning. ArXiv. https://doi.org/10.48550/ARXIV.1907.04543

Fujimoto, S., Meger, D., & Precup, D. (2019, May). Off-policy deep reinforcement learning without exploration. In International conference on machine learning (pp. 2052-2062). PMLR.

Introduction to RL and Deep Q Networks | TensorFlow Agents. (2021). TensorFlow. https://www.tensorflow.org/agents/tutorials/0_intro_rl#deep_q-learning

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(56), 1929–1958. Ανακτήθηκε από http://jmlr.org/papers/v15/srivastava14a.html

Agarwal, R. (2019). GitHub - agarwl/off_policy_mujoco: PyTorch implementation of BCQ for "Off-Policy Deep Reinforcement Learning without Exploration." GitHub. https://github.com/agarwl/off_policy_mujoco

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems, 33, 1179-1191.