Off-Policy Deep Reinforcement Learning without Exploration

CSE291C

Abstract

Review for the paper "Off-Policy Deep Reinforcement Learning without Exploration".

1 Introduction

1.1 Summary

The paper deals with the problem of off-policy reinforcement learning from a fixed batch of stored transitions (taken by an expert DDPG policy) with the constraint that no further interaction with the environment is possible. The motivation comes from the fact that standard deep learning off-policy RL algorithms like DQN and DDPG fail to optimally learn in this fixed batch setting, even though theoretically they are supposed to be able to learn from any behavioural policy. The authors show that this inability to learn truly off-policy is because unseen state-action pairs are erroneously estimated to have unrealistic values, which the authors refer to as **extrapolation error**. The authors attribute this error to the mismatch in distribution of data induced by the current policy and distribution of data contained in the fixed batch.

To overcome this extrapolation error, batch constraint Q-learning is proposed that attempts to restrict the action-space of policy so that the agent stays away from states that are unseen in fixed batch training data, assuming that the estimates for these states are inaccurate. This is done by learning a parameterized generative function (VAE used here) such that given a state s, it gives out a distribution of seen-actions $P(a|s)$ which we sample from. An actor model is then trained such that it perturbs the proposed action within a constrained region so as to maximize the Q value (estimated by a critic network).
BCQ algorithm can be viewed as unification of imitation learning and off-policy RL, who behaviour can be tuned towards any regime by changing a fixed set of hyper-parameters.

1.2 Theorem/Algorithms Challenges

1.2.1 Constraining action space idea

The authors explicitly constrain the action space of the learned policy in such a way that it does not go far from behaviour policy. While the idea does help in taking care of extrapolation error, I feel the constraint is a little too restrictive because with this approach our off-policy agent will, more or less, be only as good as behaviour policy. For example, if the behaviour policy is random policy, the learned off policy agent will perform sub-optimally even when there is sufficient data to learn a strong policy. Note that BCQ may outperform behaviour policy due to small exploration noise in actions but I feel the performance can be increased further.

1.2.2 Section 4.1

It seems like authors have considered a discrete setting in the whole of section 4.1. However, they directly switch of deep RL continuous case in section 4.2 without addressing how the discrete assumptions in theoretical foundation would affect their future analysis, if at all.

1.2.3 $\epsilon_{MDP}(s, a)$ equation (8)

The expression can be written in a more intuitive way as:

$$\epsilon_{\text{MDP}}(s,a) = \sum_{s'} (P_M(s'|s,a) - P_B(s'|s,a)) r(s,a,s') + P_M(s'|s,a)\gamma \sum_{a'} \pi(a'|s') Q^{\pi}_{M}(s',a') - P_B(s'|s,a)\gamma \sum_{a'} \pi(a'|s') Q^{\pi}_{B}(s',a')$$

The first term is the difference in expected immediate rewards while following policy under true MDP vs batch MDP. The second and third terms represent the difference in the expected long term rewards when following policy π under true MDP vs batch MDP.

1.2.4 Theorem 2 Proof

Authors define a batch **B** as coherent if for all $(s, a, s') \in B$ then $s' \in B$. They claim this condition is trivially satisfied if the data is collected in trajectories. However, this may not always be the case. Example, if buffer is made up of transitions from parts of trajectories explored by different agents, this assumption and whole theoretical foundations built on it will fall. Also, the whole analysis holds true for finite MDPs only.

1.2.5 From Discrete to Deep RL formulation ???

1.3 Experiments

Authors do a good job of comparing their BCQ algorithm against against Behaviour policy (expert policy used to create off-policy data), off-policy RL algorithms (DDPG and DQN) as well as imitation learning algorithms (behaviour cloning and its variant). They run their experiments on $'HalfCheetah'$, $'Hopper'$ and $'Walker2d'$ Mujoco environments under different fixed batch settings. It is observed that only BCQ matches or outperforms behaviour policy.

However, I feel that comparison against more advanced imitation learning algorithms which can handle noisy data (eg **DAgger** or **DART**) should also be made. This is because BCQ lies more in imitation learning regime than off-policy RL, since it essentially constricts the exploration options for policy agent whereas RL algorithms are all about optimally tuning exploration vs exploitation.

One surprising observation from the experiments is that even in Concurrent batch setting, off-policy DDPG agent failed to match performance of behaviour policy. I did not expect this since both agents are trained on transitions from essentially same buffer.

1.4 Extensions

I had a few thoughts in mind while I was reading the paper and decided to run few small experiments to test them. The base code is made available by authors on github. I run some experiments on simpler $'Pendulum'$ gym environment and others on $'Hopper'$ Mujoco environment because of lack of GPU cluster with Mujoco license installed. The experiments that I will go in detail in later sections are :

- Effect of 'expertness' of behaviour policy.
- What if buffer data is small enough that VAE can't learn $P(a|s)$ nicely.

- Robustness of algorithm to hyper-parameters.
- Using Prioritized Experience Replay instead of Uniform sampling from buffer
- Using KL loss term annealing while training VAE.
- Effect of large actor perturbations on BCQ performance.
- The papers asks to sample n(=10 by default) from VAE and then takes the best one. Is it necessary? Will the VAE collapse to a mode if we reduce this number ?
- Effect of more exploration noise while training expert policy?

2 Related Work

Fixed-Batch reinforcement learning has been previously explored in many papers. Fitted Q-iteration is one such batch algorithm which reformulates the Q function determination problem as a sequence of regression problems by iteratively extending the optimization horizon. However, this method does not provide any convergence guarantees, especially in the case of infinite time horizon tasks.

There are papers on off-policy evaluation like Liu[3]. But it uses a model based approach for estimating target policy values directly using approximated MDP by using a variant of importance sampling estimate. This paper does not use a model based approach for policy evaluation.

Approaches which use importance sampling for off-policy learning under fixed batch setting may not be possible. Remember that for importance sampling, you need the ratios of action probabilities of the current policy and behaviour policy. Since we only have transitions from behaviour policy in the fixed batch, computing this ratio would be task in itself. Retrace(λ) algorithm proposed by Munos[4] is one such algorithm referenced in this paper.

Isele & Cosgun[2] observed that the performance of off-policy agent is strongest when distribution of data in the replay buffer matched the test distribution. This observation points towards the importance of considering extrapolation error while learning from trajectory buffer.

There also Imitation learning approaches related to this paper. Cheng[1] proposes a strategy for policy learning that first performs a small but random number of Imitation learning iterations before switching to a policy gradient RL method. However this method requires further on-policy data.

also proposes to combine imitation and reinforcement learning via the idea of reward shaping. The idea is to use imitation learning on expert policy rollouts to learn value function which is an approximation to the actual optimal value function and then use RL algorithms to learn an approximate optimal policy. The more optimal the behaviour policy is, the more planning horizon can be shortened.

Both of the above approaches try and combine imitation learning with RL methods, which seems to be closely related to this paper. However, either IL or RL method is used at a time for policy learning, whereas BCQ is more of continuous interpolation between the two regimes.

Looking at the BCQ algorithm, a very famous related work that comes to mind is TRPO algorithm by Schulman[5], which aims to keep the updated policy similar to the previous policy. TRPO forces policy updates into a constrained range before collecting new data, limiting errors introduced by large changes in the policy. This is quite similar to the approach by BCQ where we keep policy actions close to behaviour policy. However, TRPO is an on-policy algorithm which needs interaction with the environment which does not satisfy fixed batch constraint.

The novelty of the paper is that BCQ provides a universal framework for both imitation learning and off-policy RL. The advantage is that BCQ is able to learn from imperfect behaviour policy with small noises. Imitation learning is known to fail when exposed to suboptimal trajectories from not-so-expert behaviour policy whereas off-policy RL is known to diverge under the constraint that no further interaction with the environment is allowed.

3 Experiments

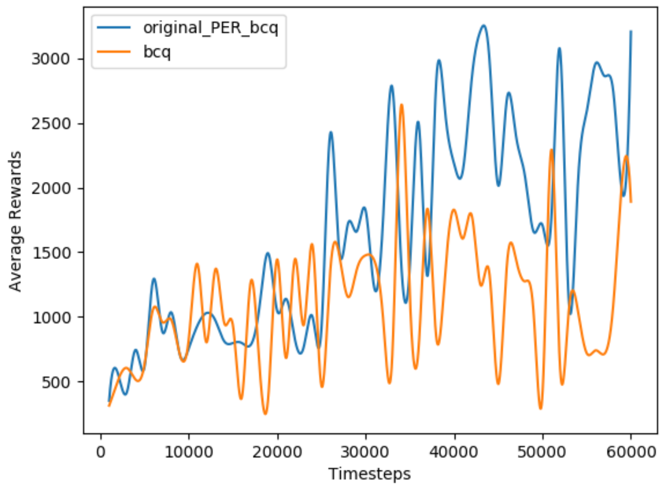
The paper implements their experiments in Mujoco environment like $Hopper$. I was able to run a few experiments in $Hopper$ environment on my Laptop and rest on simpler $Pendulum$ OpenAI gym environment on datahub GPU cluster.

3.1 Hopper Mujoco Environment

3.1.1 Prioritized Experience Replay

The authors used uniform sampling from experience replay for creating training batch for the DDPG network. Prioritized Experience Replay (PER) is one of the most important and conceptually straightforward improvement for DDPG network. It weights the buffer samples so that samples with more 'priority' are sampled more frequently. The following graph compares original BCQ vs PER BCQ.

Figure 1: PER-BCQ performance vs Original BCQ

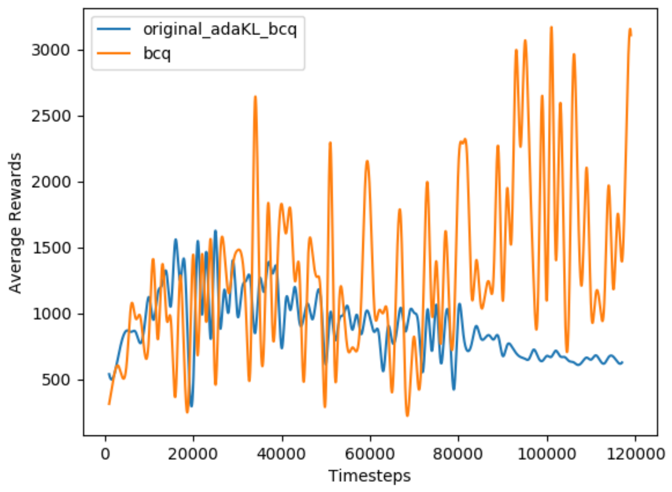


Comparing the plots, we can assert that incorporating PER into their algorithm would be beneficial for performance.

3.1.2 KL loss annealing

Adaptive KL loss annealing is a popular method which helps VAE learn better latent space representations. The approach is to first train the VAE using the reconstruction loss only for a few epochs and then gradually increase the scaling on KL loss from 0 to 1.

Figure 2: Adaptive KL annealing impact



From the plots, we can see that KL annealing is actually detrimental to the policy performance. This is rather surprising and I think this may be because BCQ VAE does not have the problem of posterior mode collapse.

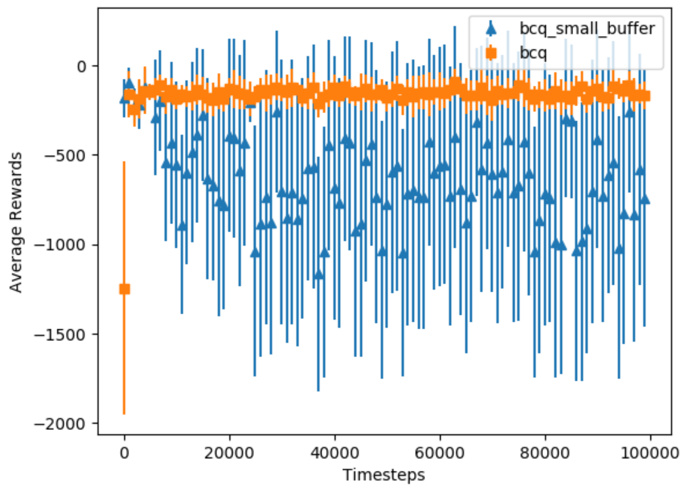
3.2 Pendulum OpenAI gym Environment

The error bars correspond to one standard deviation.

3.2.1 Effect of Buffer size

Buffer size plays an important role in off-policy RL algorithms. Since BCQ uses a VAE to learn distribution $P(a|s)$, what would happen if buffer size is small enough that learning such distribution becomes very difficult for the VAE ?

Figure 3: Effect of small buffer size



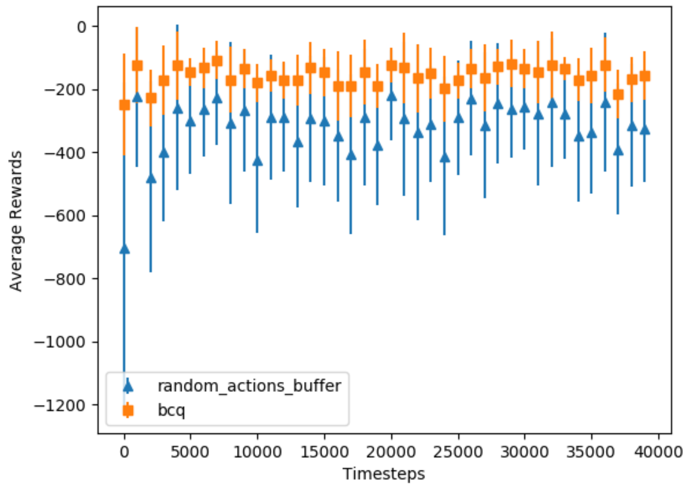
The figure clearly shows that smaller buffer size has a huge impact on the policy performance.

3.2.2 Effect of 'expertness' of behaviour policy

Out of curiosity, I created the buffer replay from a completely random behaviour policy.

This leads to a suboptimal performance from the learned policy. This is not very surprising since BCQ hinges towards more on imitation learning side and hence its only as good as behaviour policy. However, this is also one drawback of BCQ, since even after providing enough coverage over distributions (large buffer size) from random actions, the model can not learn a good policy.

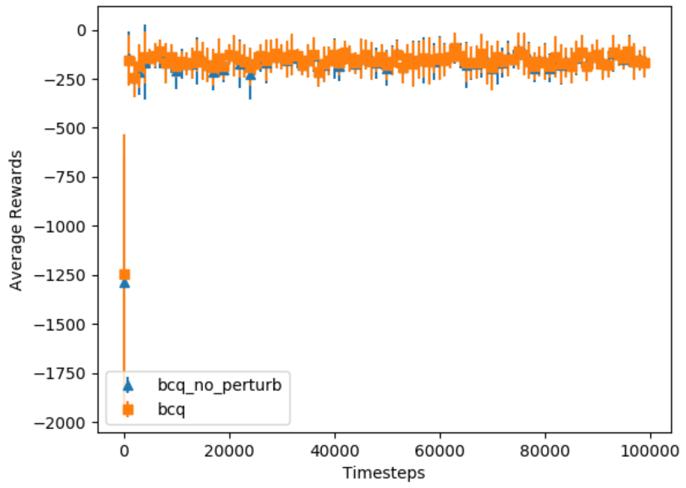
Figure 4: Impact of random behaviour policy



3.2.3 Effect of actor perturbations

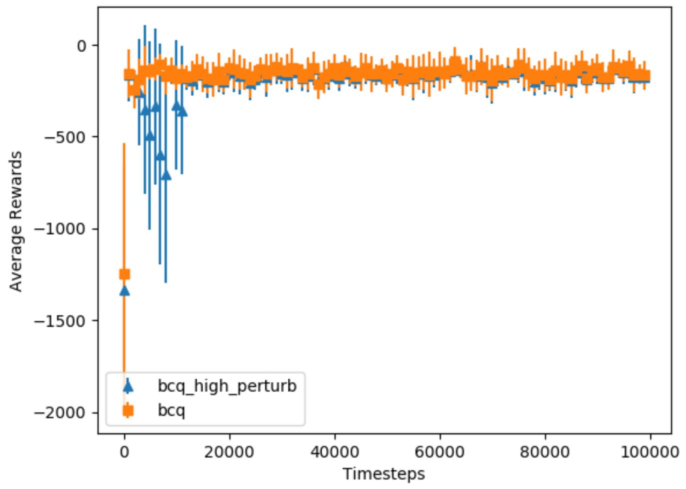
BCQ actor network produces a perturbation around the action sampled from VAE. The plot below compare the impact of no perturbation and high perturbation with original BCQ.

Figure 5: No actor perturbation impact



This is surprising because I was hoping no actor perturbation should lead to a decrease in performance since we are constraining exploration by our policy agent. I think this is because the environment is very simple and there is no need to aggressively explore the surrounding.

Figure 6: Large actor perturbation impact

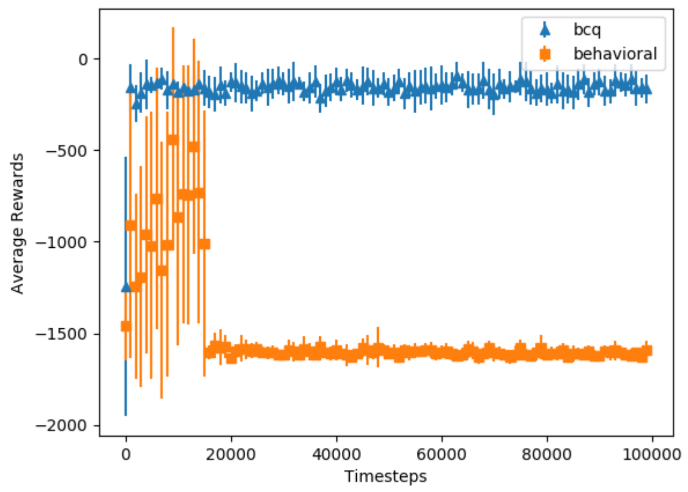


We can observe that initially the model performs suboptimally but learns to adapt over time and quickly equals the original BCQ performance.

3.2.4 Comparison with standard off-policy DDPG algo

To reproduce the result that standard off policy DDPG fails to learn from fixed batch setting, I made a comparison and is given in the figure below. It shows that off-policy DDPG diverges after some timesteps because of assigning erroneously high value estimate of states unseen in buffer.

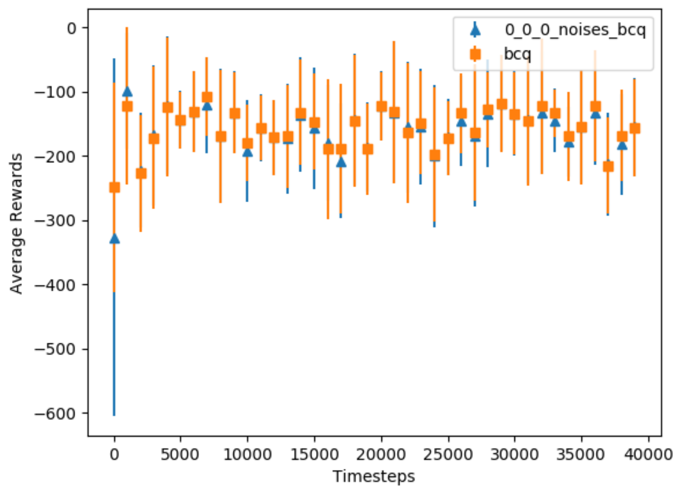
Figure 7: Comparison with standard off-policy DDPG



3.2.5 Robustness to noise

To check robustness of BCQ algo wrt exploration noise during behaviour policy training and randomness during buffer creation, I ran a few experiments. Note that these are just hyperparameter values in the model, and hence this experiment amounts to checking robustness of algo wrt hyperparameters.

Figure 8: No noises added



The policy learns almost similar to behaviour policy in this case.

Figure 9: Medium level noises

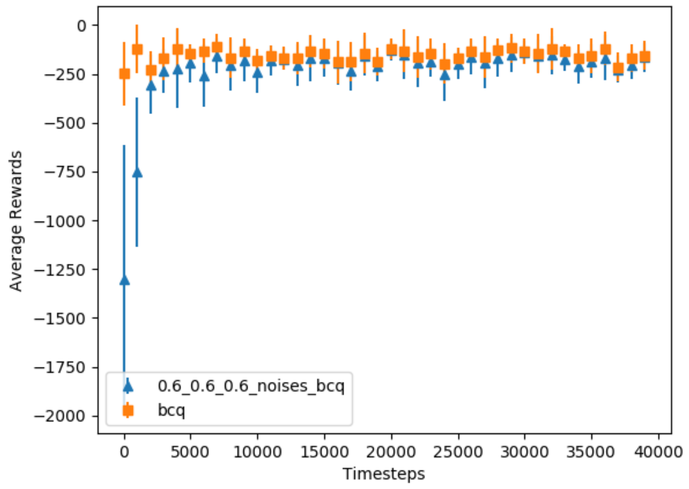
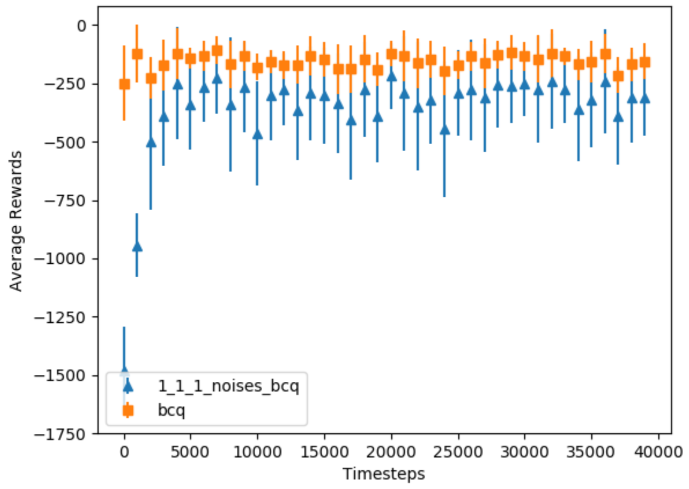


Figure 10: High level noises

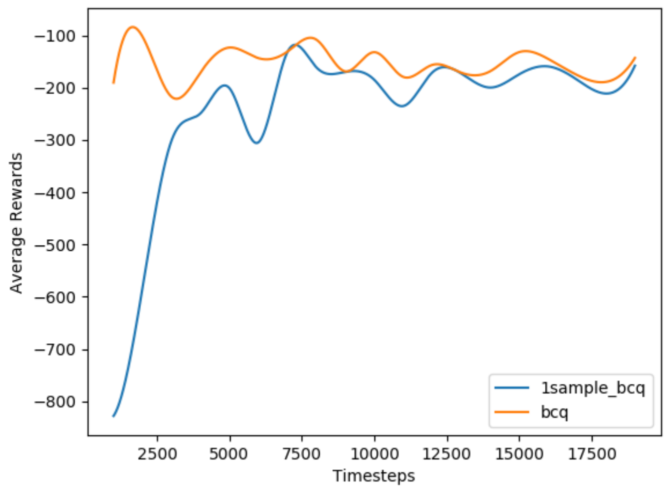


From the above figures, we can observe that BCQ starts performing suboptimally as you increase noise levels. Hence we can say that the algorithm can not learn optimally from highly imperfect demonstrations.

3.2.6 Effect of sampling only once from VAE instead of 10

From the figure below, we can see that 1sample BCQ is a little worse than sampling action 10 times from VAE. This makes sense intuitively because it would be easier for the actor network to choose a perturbation if we sample more actions from VAE posterior distribution so as to maximize score by critic network.

Figure 11: Sampling just once from VAE



4 Theorems/Algorithms

Theorem 1 states that performing Q-learning by sampling from batch B converges to optimal policy under MDP M_B, rather than the real MDP M. Intuitively, this makes sense because if you look at the Bellman-like equation

$$T^{\pi}Q = E_{s'}[r + \gamma \max_{a} Q(s', a)]$$

s' is now sampled from MDP M_B instead of MDP M.

Lemma 1 basically says that we can exactly evaluate the state-action value if and only if the transition probabilities for such pair are same for MDPs M and M_B. This makes sense because if they were not, our monte carlo estimate for finding expectation would be biased.

All the theoretical analysis after this point is done for deterministic MDP. Theorem 2 states that we can exactly estimate state-action values for all states if and only if policy π is batch constrained. A Batch constraint policy is defined as a policy whose all possible actions a in every state s is stored as (s, a) tuple in batch B. This directly comes from using Lemma 1 and noting that for deterministic MDP $P_B(s'|s, a) = P_M(s'|s, a)$.

Theorem 3 and 4 are the most interesting theorem given in this paper. It says that for a deterministic MDP and any coherent batch B, BCQ policy converges to optimal batch-constraint policy $\pi^* \in \Pi_B$ such that $Q_{\pi^*}(s, a) \geq Q_{\pi}(s, a)$ for $\pi \in \Pi_B$. This essentially means that BCQ is guaranteed to outperform any behaviour policy after observing that for any behaviour policy π, it belongs to $\pi \in \Pi_B$. This also means that BCQ should outperform any imitation learning policy since it can only be as good as behaviour policy.

The theoretical analysis is good but in practice, deterministic MDP assumption might not hold always and that's why the theoretical guarantees start to crumble. Also, all the analysis done is for discrete state and action spaces. Going towards deep RL framework, approximation errors will be introduced since it deals with continuous state action spaces.

5 Applications

5.1 Medical Treatment Optimization

Suppose an expert doctor diagnoses a disease gives you a collection of episodes containing sequence of observations and actions including the outcomes. Now you want to learn a treatment policy from the given fixed batch, at least as good as the doctor's current practice. Assuming you cannot consult the doctor anymore, the problem falls exactly in the paradigm of BCQ algorithm using which such a policy can be learned.

5.2 Recommendation system using off-policy data

Assume there is an education system portal which gives information and questions to students in order to teach a topic. In the MDP framework, current state is the information known about the student in some form of metric, eg. affinity of the student towards particular topic. Actions correspond to giving students some information or asking them a question of particular difficulty. Now given this information as a fixed batch, we want to learn a teaching policy that is specifically tuned for a student.

In both the above cases, fixed batch assumption is necessary since deploying a new policy is either costly or risky. Example, the cost one bad decision in healthcare can result in patient's death.

References

[1] Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast policy learning through imitation and reinforcement, 2018.

[2] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning, 2018.

[3] Yao Liu, Omer Gottesman, Aniruddh Raghu, Matthieu Komorowski, Aldo A Faisal, Finale Doshi-Velez, and Emma Brunskill. Representation balancing mdps for off-policy policy evaluation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2644–2653. Curran Associates, Inc., 2018.

[4] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning, 2016.

[5] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2015.