Model-Based Offline Reinforcement Learning

Pascal Schindler, Karam Daaboul
Karlsruhe Institute of Technology
Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe, Germany
ujvhi@student.kit.edu, daaboul@kit.edu

***Abstract*—In reinforcement learning, offline model-based methods aim to combine the benefits of large and diverse datasets with the sample-efficiency of model-based methods. The goal of this paper is to provide an overview of the latest offline model-based reinforcement learning ideas by analyzing how these techniques incorporate conservatism into the learning process.**

I. Introduction

The successes of machine learning in recent years are primarily based on supervised learning [5] [6]. In supervised learning, it is possible to use large and diverse datasets to recognize abstract rules and patterns in the data. Compared to supervised learning, reinforcement learning is seen as an online learning paradigm. An agent acts in the environment and receives a feedback signal. After that, the agent adapts its behavior and, over time, increases its performance by learning better behavior in the environment. The currently emerging field of offline reinforcement learning is dedicated to the task of incorporating the successes of data-driven supervised learning into reinforcement learning. The goal is to derive a reliable policy from large amounts of data [1]. Model-free algorithms, a subarea of reinforcement learning, are already able to achieve good results on baseline data sets, but need hundreds of thousands to millions of transitions to achieve sufficient results [2]. In contrast, model-based algorithms, an emerging subarea of reinforcement learning, show an extreme sample efficiency. By building a model of the environment, the agent can navigate artificially through the environment and informally predict the future. Combining model-based reinforcement learning and data-driven models promises highly efficient use of data to derive reliable models, thus avoiding the dangers of online reinforcement learning in risky areas.

II. Preliminaries

A. Offline Reinforcement Learning

So far, it is quite challenging to use common reinforcement learning (RL) algorithms in real life applications since many algorithms initially work according to the trial-and-error principle [17] [18]. Only with much experience and environment exploration, the algorithm can find a good policy and solve the underlying task well. The initial trial-and-error approach makes it difficult to use RL in domains where human life is involved. Prominent examples of risky environments are RL applications in healthcare or autonomous driving. One approach to this problem is the application of data-driven methods in RL [7] [8] [9]. In pure offline RL, interaction with the environment is not possible [1] [3]. The agent is only allowed to interact with the previously collected data set and the goal is to derive the best possible policy given the data set size and its quality. This is a non-trivial task [4]. When applying conventional RL algorithms to static data sets in an offline RL fashion, a phenomenon called distributional shift occurs. Distributional shift refers to the deviation between the learned policy and the policy used to generate the dataset, also called behavior policy [1] [4]. This phenomenon is reflected by the fact that the policy achieves very high rewards during the training but performs very poorly during deployment time in the environment. The reason for this behavior is that the algorithms assign a very high value to out-of-distribution states [9] [10]. The common approach to this problem is to incorporate some form of pessimism or conservatism into the learning process [11] [12]. This leads to out-of-distribution states not being overestimated, and the learned policy staying close to behavior policy [4]. Overcoming the problem of distribution shift is a crucial step for the success of offline RL.

B. Model-Based Reinforcement Learning

In general, the algorithm landscape in RL can be divided into two areas. Model-free and model-based algorithms. Model-free algorithms derive a policy directly and in the case of offline RL either try to keep the learned policy close to the data set behavior [9] [8] [11] [16] or apply uncertainty quantification techniques to stabilize the Q-function [13] [14] [15]. In contrast, model-based algorithms firstly generate a model of the environment, called environment or dynamics model, to represent its dynamics. Hereby, we consider a standard Markov Decision Process (MDP) $\mathcal{M} = (S, A, T, r, \mu_0, \gamma)$, with S and A as state- and action-space, $T(s'|s,a)$ as the environment transition dynamics, $r(s,a)$ as the reward function, μ_0 as the initial state-distribution and γ as the discount factor. In model-based RL (MBRL), we learn a dynamic function model $\widehat{T}$ typically via maximum likelihood-estimation: $\min_{\widehat{T}} \mathbb{E}_{s,a,s' \sim D} \left[\log \widehat{T}(s'|s,a) \right]$. Hereby, D can denote a static dataset or represents a previously collected buffer. Once the model is learned, we can construct the MDP $\widehat{\mathcal{M}} = (S, A, \widehat{T}, \hat{r}, \mu_0, \gamma)$ with the same state-action-space as the true MDP but with the learned environment dynamics. The optimal policy in the model is recovered as

$\hat{\pi} = \arg\max_\pi J(\widehat{\mathcal{M}}, \pi)$ and can be obtained in various ways such as planning algorithms [19].

III. Model-based Offline Reinforcement Learning

As the name suggests, model-based offline RL is the combination of offline RL and model-based RL. Hereby, the dynamics model of the environment is learned from a static data set rather than through online interaction with the environment. By interacting with the learned model, it is possible for the agent to derive an optimal policy [2] or optimize a sequence of actions (MOPO). In comparison to model-free offline RL, the augmented data generated from the model is used to maximize the expected cumulative rewards.

A. Challenges in Model-Based Reinforcement Learning

One major challenge of model-based RL is the occurrence of model bias. As stated before, the offline RL setting does not allow online interaction. Normally, MBRL models can correct occurring bias by interacting with the environment. This is not possible with the opposed restriction to learn in an offline fashion. Especially out-of-distribution states and actions are prone to errors and can lead to a significant bias. In these areas of the state action spaces, the learned policy can exploit errors in the dynamics model and falsely obtain high rewards. This can lead to inferior performance when the model is deployed in the real MDP [2]. Additionally, the model accuracy of the dynamics model is not always good which is reflected in a bad asymptotic performance behavior [24]. Furthermore, in high-dimensional environments, it is difficult to use these predictive dynamics models. Inaccuracies in the model cause long rollouts to become too inaccurate. One approach to tackle this problem of inaccuracy in the dynamics model are ensemble methods. In the past, ensembles of dynamic models have proven to be a very effective means to prevent the exploitation of inaccuracies in the dynamics model [25]. Overall, a combination of model ensembles and short rollout horizons has proven to be very effective in preventing such exploitation [22].

To formalize the problem of inaccuracy, in [22], some performance guarantees for MBRL algorithms are formulated. By performance guarantees, we mean a theoretical property that specifies the relationship between the planning performance in the model and the actual planning performance in the real environment:

$$\eta_{\mathcal{M}}[\pi] \geq \hat{\eta}_{\widehat{\mathcal{M}}}[\pi] - C. \tag{1}$$

We denote $\eta_{\mathcal{M}}[\pi]$ as the return of policy π in the true MDP $\mathcal{M}$ and $\hat{\eta}_{\widehat{\mathcal{M}}}[\pi]$ as the return of the policy π in the modeled MDP $\widehat{\mathcal{M}}$. The additional variable C models the gap between those two properties. This gap represents the two fundamental error quantities in MBRL: firstly, the distributional shift due to the visitation of out-of-distribution states and actions and, secondly, the generalization error due to sampling. In [22], a trade-off between these two error sources can be observed. Hereby, the model error increases as the KL-divergence between the current policy and the data-collecting policy increases. However, the strength of the model error can be influenced by the amount of data. This indicated that the more data is collected by the behavior policy, the lower the model error afterward. Furthermore, models that have been trained on more data show a better generalization behavior

In the following sections, we will present the different design choices used by the latest state-of-the-art offline MBRL algorithms. This review will divide those algorithms into two categories based on how they estimate uncertainty in the model [26] [27] [28] [29].

B. Explicit Uncertainty Quantification: MDP Augmentation

Before we introduce the actual algorithms and their function, we need some intuition.

Firstly, we denote the difference in the value-function with different dynamics models, the real dynamics T and the learned dynamics $\hat{T}$, as follows [2]:

$$G^\pi_{\widehat{\mathcal{M}}}(s,a) = \mathop{\mathbb{E}}_{s' \sim \hat{T}(s,a)} \left[V^\pi_{\mathcal{M}}(s')\right] - \mathop{\mathbb{E}}_{s' \sim T(s,a)} \left[V^\pi_{\mathcal{M}}(s')\right]. \tag{2}$$

Intuitively, this governs the performance difference of the dynamics model between the real MDP $\mathcal{M}$ and the learned MDP $\widehat{\mathcal{M}}$. Therefore, if $\mathcal{M} = \widehat{\mathcal{M}}$, it follows that $G^\pi_{\widehat{M}}(s,a) = 0$. From this equation follows:

$$\eta_{\widehat{\mathcal{M}}}(\pi) - \eta_{\mathcal{M}}(\pi) = \gamma \mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[G^\pi_{\widehat{\mathcal{M}}}(s,a)\right]. \tag{3}$$

In equation (3), we can interpret $G^\pi_{\widehat{\mathcal{M}}}(s,a)$ as a performance measurement of the policy π in the real MDP and the modeled MDP and see that the performance is highly dependent on the accuracy of the dynamics mode. Furthermore, we can derive the following corollary:

$$\begin{aligned}\eta_{\mathcal{M}}(\pi) &= \mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[r(s,a) - \gamma G^\pi_{\widehat{\mathcal{M}}}(s,a)\right] \\ &\geq \mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[r(s,a) - \gamma |G^\pi_{\widehat{\mathcal{M}}}(s,a)|\right].\end{aligned} \tag{4}$$

Together with equation (4), we can interpret that a policy that receives high rewards in the estimated MDP $\widehat{\mathcal{M}}$ while minimizing $G^\pi_{\widehat{M}}(s,a)$, also receives high rewards in the real MDP $\mathcal{M}$.

Now, to come to the first algorithms, we introduce Model-based Offline Policy Optimization (MOPO). Firstly, we use equation (4) to derive the following bounds:

$$\begin{aligned}&\eta_{\mathcal{M}}(\pi) \geq \mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[r(s,a) - \gamma |G^\pi_{\widehat{\mathcal{M}}}(s,a)|\right] \geq \\ &\mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[r(s,a) - \lambda u(s,a)\right] \geq \mathop{\bar{\mathbb{E}}}_{(s,a)\sim p^\pi_{\hat{T}}} \left[\tilde{r}(s,a)\right] = \eta_{\widehat{\mathcal{M}}_u}(\pi).\end{aligned} \tag{5}$$

Hereby, we define the uncertainty-penalized reward as $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. In this definition, u is an error estimator of the modeled dynamics $\hat{T}$ Furthermore, we redefine the uncertainty-penalized MDP $\widehat{\mathcal{M}}_u = (S, A, \widehat{T}, \tilde{r}, \mu_0, \gamma)$.

In equation (5), we now formulated a lower bound of the true return and, therefore, incorporated conservatism into the MDP. In MOPO, the environment dynamics are modeled through a neural network with a Gaussian distribution over the next state and reward as output: $\widehat{T}_{\theta,\phi}(s_{t+1}, r|s_t, a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\phi(s_t, a_t))$. In general, bootstrap ensembles empirically perform well in quantifying the model uncertainty [29]. Therefore, in MOPO, an ensemble of N dynamics models $\{\widehat{T}^i_{\theta,\phi} = \mathcal{N}(\mu^i_\theta, \Sigma^i_\phi)\}^N_{i=1}$ is learned via maximum likelihood estimation. Moreover, the uncertainty-penalized reward is modeled through the maximum standard deviation in the learned ensemble $u(s,a) = \max^N_{i=1} \|\Sigma^i_\phi(s,a)\|_F$. The reward is computed in the following fashion: $\tilde{r}(s,a) = \hat{r}(s,a) - \lambda \max_{i=1,\dots,N} \|\Sigma^i_\phi(s,a)\|_F$ with $\hat{r}$ as the mean predicted reward of $\widehat{T}$.

Another approach that uses an augmented MDP is the Model-based Offline Reinforcement Learning (MOReL) algorithm. MOReL uses a pessimistic-MDP (P-MDP) that models the uncertainty of the dynamics model [30] [31] [32] [33]. Hereby, the P-MDP is used to divide the state-space into "known" and "unknown" regions. The agent will receive a large negative reward when entering the unknown state-space. For the quantification of unknown state-spaces, MOReL utilizes an Unkwnon state-action detector (USAD) to partition the state space:

$$U^\alpha(s,a) = \begin{cases} \textit{known} & \text{if } D_{TV}\left(\hat{T}(\cdot|s,a), T(s,a)\right) \leq \alpha \\ \textit{unknown} & \text{otherwise} \end{cases}$$

with $D_{TV}\left(\hat{T}(\cdot|s,a), T(s,a)\right)$ as the total variation distance between $\hat{T}(\cdot|s,a)$ and $T(s,a)$. Intuitively, USAD benefits from a greater state coverage and, therefore, from larger and more diverse datasets. To incorporate the USAD into the (α,κ)-pessimistic-MDP, the MDP is altered in the following way: $\widehat{\mathcal{M}}_p = \{S \cup \textit{HALT}, A, r_p, \hat{T}_p, \hat{\mu}_0, \gamma\}$. Hereby, *HALT* defines an absorbing state, which provides a large negative reward κ and the episode is terminated. The new modified reward and dynamics are given in the following way:

$$\widehat{T}_p(s'|s,a) = \begin{cases} \delta(s' = \text{HALT}) & \text{if } U^\alpha(s,a) = \text{unknown} \\ & \text{or } s = \text{HALT} \\ \widehat{T}(s'|s,a) & \text{otherwise} \end{cases}$$

and

$$r_p(s,a) = \begin{cases} -\kappa & \text{if s} = \text{HALT} \\ r(s,a) & \text{otherwise} \end{cases}$$

Overall, MOReL incorporates pessimism into the learning process by greatly penalizing the entering of unknown states and terminating the episode. This helps to perform policy search in known dataset states, but, at the same time, prevents drifting into unknown states.

C. Implicit Uncertainty Quantification: Policy Regularization

In the previous section, we discussed the approach of incorporating conservatism into the algorithms by augmenting the MDP. Hereby, conservatism helps the learned policy to stay close to the behavior policy in the dataset and, therefore, prevents distributional shift between those two policies. However, incorporating the uncertainty quantification methods of MOPO or MOReL into high-dimensional observation spaces such as image-based environments can impose a major challenge [34] [35] [36] [37]. Therefore, another approach is to use policy regularization techniques which implicitly quantify uncertainty by bounding the learned policy to the behavior policy.

Hereby, the offline model-based RL with Adaptive Behavioral Regularization (MABE) algorithm completely renounces any uncertainty quantification in the sense of dynamics models evaluation and, instead, learns a dynamics model, reward and adaptive behavioral prior. The adaptive behavioral prior describes a policy that approximates the behavioral policy in the dataset while giving more importance to trajectories leading to high rewards. This is helpful with diverse datasets collected with multiple non-expert policies. During the learning process, the objective of MABE is to maximize the reward while staying close to the adaptive behavioral prior with a KL-divergence penalty [38]. Hereby, the dynamics model is learned with maximum likelihood learning similar to MOPO or MOReL.

Firstly, the adaptive behavioral prior, a parameterized generative model, is learned in the following fashion:

$$\pi^{\text{adapt}}_\alpha \in \arg\max_{\pi_\alpha} \mathbb{E}_{(\tau)\sim D}\left[\sum_{t=0}^{|\tau|} w(s_t,a_t)\cdot \pi_\alpha(a_t|s_t)\right]$$
$$\text{s.t.}\, \mathbb{E}_{s\sim D}[\text{D}_{\text{KL}}(\pi_\alpha \| \pi_\beta)] \leq \delta,$$

with π_β as the empirical behavioral policy, $w(s_t,a_t)$ as the weighting function and δ as constraint threshold. The weighting function is defined as: $w(s_t,a_t) = \hat{Q}(s_t,a_t)\cdot(1-\gamma)/r_{\max}$ with $r_{\max}$ as the maximum reward in the dataset and η as temperature hyperparameter. Hereby, the weighting function helps to nudge the adaptive behavioral prior towards trajectories leading to high rewards. Overall, with the adaptive behavioral prior, a generative model is used to represent the dynamics and average statistics in the data set. The goal of the restricted objective is to be biased towards high-reward trajectories while staying close to the empirical behavioral policy. This helps to stay conservative and prevents deviating from dataset information.

Overall, we update the Q-function in the following fashion:

$$\bar{Q} = r(s_t,a_t) + \gamma[Q_\phi(s'_t, \pi_\theta(\cdot|s'_t)) - \beta \text{D}_{\text{KL}}(\pi_\theta(\cdot|s'_t), \pi_\alpha(\cdot|s'_t))]. \tag{6}$$

We can see that a large deviating from the adaptive behavioral prior is penalized by measuring the KL-divergence between the current policy on the behavioral prior and, implicitly, nudges the learned policy towards the adaptive behavioral prior.

Another way to utilize policy regularization is Behavior-Regularized Model-ENsemble (BREMEN). BREMEN learns an ensemble of dynamics models and an implicitly regularized policy. Here, appropriate parameter initialization and conservative trust-region learning updates are used. Firstly, BREMEN

learns K dynamics models $\hat{f}_\phi = \{\hat{f}_{\phi_1}, ..., \hat{f}_{\phi_K}\}$ with the following objective [40]:

$$\min_{\phi_i} \frac{1}{|D|} \sum_{(s_t, a_t, s_{t+1}) \in D} \frac{1}{2} \| s_{t+1} - \hat{f}_{\phi_i}(s_t, a_t) \|_2^2. \quad (7)$$

Secondly, BREMEN performs policy updates with behavior regularization. To approximate the behavior policy in the dataset, a policy $\hat{\pi_\beta}$ is trained via behavior cloning [39]:

$$\min_{\beta} \frac{1}{|D|} \sum_{(s_t, a_t) \in D} \frac{1}{2} \| a_t - \hat{\pi}_\beta(s_t) \|_2^2. \quad (8)$$

Afterward, the policy π_θ is initialized as Gaussian with the $\hat{\pi}_\beta$ mean and a standard deviation of 1. This initialization with the BC parameters can be seen as an implicit policy regularization towards the true behavior policy and, therefore, implicitly prevents behavior shifting. To further tighten the constraint towards the behavior policy, BREMEN uses a KL-based trust-region optimization to fine-tune the policy [41]:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s,a \sim \pi_{\theta_k}, \hat{f}_{\phi_i}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}} \right]$$
$$\text{s.t.} \mathop{\mathbb{E}}_{s \sim \pi_{\theta_k}, \hat{f}_{\phi_i}} [\mathrm{D_{KL}}(\pi_\theta(\cdot|s) \| \pi_{\theta_k}(\cdot|s))] \leq \delta, \; \pi_{\theta_0} = \mathcal{N}(\hat{\pi_\beta}, 1),$$

where $A^{\pi_{\theta_k}}$ is the advantage of π_{θ_k}. This is computed via roll-outs in the dynamics models with maximum step size δ. The imagined trajectories are generated by randomly selecting a dynamics model from the ensemble. By recursively performing offline updates combined with online fine-tuning, BREMEN shows an extreme deployment efficiency.

Another algorithm that does not need to augment the MDP is Conservative Offline Model-Based Policy Optimization (COMBO) [19]. COMBO follows the design principle of many model-free offline RL algorithms [42] [44] [45] and incorporates pessimism into the policy or value function training. COMBO learns the dynamics model from a static dataset and, afterward, uses the model and imagined trajectories to learn a conservative value function. Here, conservative refers to the penalization of the value function in out-of-distribution state-action tuples. The obtained learned Q-function even provides a tighter bound to the true Q-function compared to Conservative Q-Learning (CQL) [4], on which COMBO is based on. The conservative policy update is performed in the following way:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \beta \left(\mathbb{E}_{s,a \sim p(s,a)}[Q(s,a)] - \mathbb{E}_{s,a \sim D}[Q(s,a)] \right)$$
$$+ \frac{1}{2} \mathbb{E}_{s,a,s' \sim d_f} \left[\left(Q(s,a) - \hat{\mathcal{B}}^\pi \hat{Q}^k(s,a) \right)^2 \right]. \quad (9)$$

Here, $p(s,a)$ is a sampling distribution defined as $p(s,a) = d^\pi_{\widehat{\mathcal{M}}}(s)\pi(a|s)$ with $d^\pi_{\widehat{\mathcal{M}}}(s)$ as discounted marginal state distribution when rolling out policy π in the dynamics model $\widehat{M}$. Furthermore, d_f is a mixture of offline dataset trajectories and synthetically collected trajectories in the dynamics model. This results in such behavior that the Q-values on state-action tuples obtained from the model's roll-out are pushed down and the Q-values on state-action tuples from the dataset are pushed up. For the Q-value update with the Bellman backup, a mixture of model-generated and real data is used. Compared to the original CQL algorithm, COMBO learns on a richer set of states since model-generated states are also used. Moreover, the mixture ratio between data set data and roll-out data seems crucial in the trade-off between conservatism and generalization.

IV. Off-Policy Evaluation

Methods like COMBO are based on a classical actor-critic architecture. Here, a policy function is learned from the actor and a value function is learned from the critic. The actor acts with the help of the policy function through the environment and collects data. The critic uses this data and tries to evaluate the actor's actions. Thereby, actor-critic methods rely on off-policy evaluation. One approach for evaluating the value function is Fitted Q Evaluation (FQE) [46] [47] [48]. In general, off-policy evaluation (OPE) methods try to predict the value function of a target policy from batch episodes. Among all the OPE methods, FQE can convince by its simplicity. The following represents a simplified algorithmic framework:

Algorithm 1 Fitted Q Evaluation [46]

Input: Dataset $\mathcal{D}$, policy π to evaluate

1: **for** $n_{updates}$ **do**
2: Sample $\{s_i, a_i, r_i, s'_i\}_{i=1}^{batchsize}$ from $\mathcal{D}$
3: Update critic in the following fashion: {d as divergence measure}
4: $-\nabla_\theta d(Q_\theta(s_t, a_t), r_t + \gamma \mathbb{E}_{a \sim \pi_\phi(s_{t+1})} Q_\theta(s_{t+1}, a))$
5: **end for**

In several experiments, FQE was shown to be a simple and scalable tool for policy evaluation in the offline setting and was able to significantly reduce over-estimation. Therefore, FQE approaches the problem of policy shift.

V. Conclusion

In reinforcement learning, it's about making decisions. In general, a distinction is made between model-free methods, which derive a policy directly from the given data, and model-based methods, which first generate a predictive model of the world in order to move within it and find optimal solutions. If we now have the additional restriction that the policies may only be derived from a static data set, we are in offline reinforcement learning. The motivation to use model-based RL algorithms also in the area of offline RL is the extreme sample efficiency of model-based methods. This has the advantage in the offline setting that a reliable policy can be derived quickly from a limited data set.

Nevertheless, just as with model-free methods, there is the problem of policy shift, where the learned policy deviates too far from the behavior policy of the data set. Additionally, there is the disadvantage that model bias is an additional error source

in model-based methods compared to model-free methods. For this, a certain conservatism or pessimism must be incorporated into the algorithms using various tricks. Furthermore, there is the strong need for uncertainty quantification to prevent such model bias. Surprisingly, although the topic of model-based RL is very new in the field of offline RL, the by now designed algorithms are already capable of achieving the same results on standardized data sets as the latest model-free algorithms in a much shorter time.

Furthermore, the main reason for the success of machine learning is the use of large datasets. Here, model-based offline RL methods have the potential to derive a good predictive model from a dataset and use it to generate further synthetically generated data. Given a good quality of the artificial data, this can help to learn an even better predictive model. This is especially useful in environments where data collection is particularly difficult or safety critical.

In addition, it has already been shown that offline model-based RL algorithms can also achieve good out-of-distribution generalization, i.e. the learned model can solve tasks that are not represented in the data set. This is because the predictive model can also be used for other tasks since it is a artificial representation of the environment.

Nevertheless, many obstacles still need to be overcome. It is especially important to find a reliable method to incorporate uncertainty into the algorithm. Also, as with model-free algorithms, there is still the problem of reward design, i.e., designing a suitable reward function that produces the desired behavior. Furthermore, it has been shown that especially neural networks are susceptible to adversarial attacks which is an additional problem in safety critical environments. In summary, offline model-based RL has great potential, but it must always be ensured that sufficient robustness is guaranteed.

REFERENCES

[1] Levine, Sergey, et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." arXiv preprint arXiv:2005.01643 (2020).

[2] Yu, Tianhe, et al. "Mopo: Model-based offline policy optimization." arXiv preprint arXiv:2005.13239 (2020).

[3] Agarwal, Rishabh, Dale Schuurmans, and Mohammad Norouzi. "An optimistic perspective on offline reinforcement learning." International Conference on Machine Learning. PMLR, 2020.

[4] Kumar, Aviral, et al. "Conservative q-learning for offline reinforcement learning." arXiv preprint arXiv:2006.04779 (2020).

[5] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[7] Ernst, Damien, Pierre Geurts, and Louis Wehenkel. "Tree-based batch mode reinforcement learning." Journal of Machine Learning Research 6 (2005): 503-556.

[8] Fujimoto, Scott, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration." International Conference on Machine Learning. PMLR, 2019.

[9] Kumar, Aviral, et al. "Stabilizing off-policy q-learning via bootstrapping error reduction." arXiv preprint arXiv:1906.00949 (2019).

[10] Fujimoto, Scott, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration." International Conference on Machine Learning. PMLR, 2019.

[11] Wu, Yifan, George Tucker, and Ofir Nachum. "Behavior regularized offline reinforcement learning." arXiv preprint arXiv:1911.11361 (2019).

[12] Siegel, Noah Y., et al. "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning." arXiv preprint arXiv:2002.08396 (2020).

[13] Nachum, Ofir, et al. "Algaedice: Policy gradient from arbitrary experience." arXiv preprint arXiv:1912.02074 (2019).

[14] Peng, Xue Bin, et al. "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning." arXiv preprint arXiv:1910.00177 (2019).

[15] Siegel, Noah Y., et al. "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning." arXiv preprint arXiv:2002.08396 (2020).

[16] Jaques, Natasha, et al. "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog." arXiv preprint arXiv:1907.00456 (2019).

[17] Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems." IEEE transactions on systems, man, and cybernetics 5 (1983): 834-846.

[18] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[19] Yu, Tianhe, et al. "Combo: Conservative offline model-based policy optimization." arXiv preprint arXiv:2102.08363 (2021).

[20] ZHAO, Tingting, et al. "Review of Model-Based Reinforcement Learning." Journal of Frontiers of Computer Science and Technology 14.6 (2020): 918-927.

[21] Walsh, Thomas, Sergiu Goschin, and Michael Littman. "Integrating sample-based planning and model-based reinforcement learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 24. No. 1. 2010.

[22] Janner, Michael, et al. "When to trust your model: Model-based policy optimization." arXiv preprint arXiv:1906.08253 (2019).

[23] Watters, Nicholas, et al. "Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration." arXiv preprint arXiv:1905.09275 (2019).

[24] Clavera, Ignasi, et al. "Model-based reinforcement learning via meta-policy optimization." Conference on Robot Learning. PMLR, 2018.

[25] Osband, Ian, John Aslanides, and Albin Cassirer. "Randomized prior functions for deep reinforcement learning." arXiv preprint arXiv:1806.03335 (2018).

[26] Azizzadenesheli, Kamyar, Emma Brunskill, and Animashree Anandkumar. "Efficient exploration through bayesian deep q-networks." 2018 Information Theory and Applications Workshop (ITA). IEEE, 2018.

[27] Burda, Yuri, et al. "Exploration by random network distillation." arXiv preprint arXiv:1810.12894 (2018).

[28] Lowrey, Kendall, et al. "Plan online, learn offline: Efficient learning and exploration via model-based control." arXiv preprint arXiv:1811.01848 (2018).

[29] Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." arXiv preprint arXiv:1805.12114 (2018).

[30] Kidambi, Rahul, et al. "Morel: Model-based offline reinforcement learning." arXiv preprint arXiv:2005.05951 (2020).

[31] Kearns, Michael, and Satinder Singh. "Near-optimal reinforcement learning in polynomial time." Machine learning 49.2 (2002): 209-232.

[32] Brafman, Ronen I., and Moshe Tennenholtz. "R-max-a general polynomial time algorithm for near-optimal reinforcement learning." Journal of Machine Learning Research 3.Oct (2002): 213-231.

[33] Kakade, Sham, Michael J. Kearns, and John Langford. "Exploration in metric state spaces." Proceedings of the 20th International Conference on Machine Learning (ICML-03). 2003.

[34] Ovadia, Yaniv, et al. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift." arXiv preprint arXiv:1906.02530 (2019).

[35] Begoli, Edmon, Tanmoy Bhattacharya, and Dimitri Kusnezov. "The need for uncertainty quantification in machine-assisted medical decision making." Nature Machine Intelligence 1.1 (2019): 20-23.

[36] Jiang, Heinrich, et al. "To trust or not to trust a classifier." arXiv preprint arXiv:1805.11783 (2018).

[37] Abdar, Moloud, et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges." Information Fusion (2021).

[38] Cang, Catherine, et al. "Behavioral Priors and Dynamics Models: Improving Performance and Domain Transfer in Offline RL." arXiv preprint arXiv:2106.09119 (2021).
[39] Rajeswaran, Aravind, et al. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations." arXiv preprint arXiv:1709.10087 (2017).
[40] Matsushima, Tatsuya, et al. "Deployment-efficient reinforcement learning via model-based offline optimization." arXiv preprint arXiv:2006.03647 (2020).
[41] Schulman, John, et al. "Trust region policy optimization." International conference on machine learning. PMLR, 2015.
[42] Fujimoto, Scott, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." International Conference on Machine Learning. PMLR, 2018.
[43] Kumar, Aviral, et al. "Stabilizing off-policy q-learning via bootstrapping error reduction." arXiv preprint arXiv:1906.00949 (2019).
[44] Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361, 2019.
[45] Jaques, Natasha, et al. "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog." arXiv preprint arXiv:1907.00456 (2019).
[46] Le, Hoang, Cameron Voloshin, and Yisong Yue. "Batch policy learning under constraints." International Conference on Machine Learning. PMLR, 2019.
[47] Brandfonbrener, David, et al. "Offline RL Without Off-Policy Evaluation." arXiv preprint arXiv:2106.08909 (2021).
[48] Paine, Tom Le, et al. "Hyperparameter selection for offline reinforcement learning." arXiv preprint arXiv:2007.09055 (2020).