Offline Learning for Planning: A Summary

Giorgio Angelotti, Nicolas Drougard, Caroline Ponzoni Carvalho Chanel

▶ **To cite this version:**

Giorgio Angelotti, Nicolas Drougard, Caroline Ponzoni Carvalho Chanel. Offline Learning for Planning: A Summary. Bridging the Gap Between AI Planning and Reinforcement Learning (PRL), ICAPS 2020 Workshop, Oct 2020, Nancy, France. pp.153-161. hal-03125176

HAL Id: hal-03125176
https://hal.science/hal-03125176

Submitted on 29 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive Toulouse Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of some Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in: https://oatao.univ-toulouse.fr/26790

Official URL:

To cite this version :

Angelotti, Giorgio and Drougard, Nicolas and Ponzoni Carvalho Chanel, Caroline Offline Learning for Planning: A Summary. (2020) In: Bridging the Gap Between AI Planning and Reinforcement Learning (PRL), ICAPS 2020 Workshop, 22 October 2020 - 23 October 2020 (Nancy, France). (Unpublished)

Any correspondence concerning this service should be sent to the repository administrator:
tech-oatao@listes-diff.inp-toulouse.fr

Offline Learning for Planning: A Summary

Giorgio Angelotti,[1,2] **Nicolas Drougard,**[1,2] **Caroline P. C. Chanel**[1,2]
[1]ANITI - Artificial and Natural Intelligence Toulouse Institute, Université de Toulouse,
41 Allées Jules Guesde, 31013 Toulouse - CEDEX 6, France
[2]ISAE-SUPAERO, Université de Toulouse,
10 Avenue Edouard Belin, 31055 Toulouse - CEDEX 4, France
{name.surname}@isae-supaero.fr

Abstract

The training of autonomous agents often requires expensive and unsafe trial-and-error interactions with the environment. Nowadays several data sets containing recorded experiences of intelligent agents performing various tasks, spanning from the control of unmanned vehicles to human-robot interaction and medical applications are accessible on the internet. With the intention of limiting the costs of the learning procedure it is convenient to exploit the information that is already available rather than collecting new data. Nevertheless, the incapability to augment the batch can lead the autonomous agents to develop far from optimal behaviours when the sampled experiences do not allow for a good estimate of the true distribution of the environment. Offline learning is the area of machine learning concerned with efficiently obtaining an optimal policy with a batch of previously collected experiences without further interaction with the environment. In this paper we adumbrate the ideas motivating the development of the state-of-the-art offline learning baselines. The listed methods consist in the introduction of epistemic uncertainty dependent constraints during the classical resolution of a Markov Decision Process, with and without function approximators, that aims to alleviate the bad effects of the distributional mismatch between the available samples and real world. We provide comments on the practical utility of the theoretical bounds that justify the application of these algorithms and suggest the utilization of Generative Adversarial Networks to estimate the distributional shift that affects all of the proposed model-free and model-based approaches.

Learning using a single batch of collected experiences is a statistical challenge of crucial importance for the development of intelligent agents, specially in scenarios where the interaction with the environment can be expensive, risky or unpractical. There are countless examples that fall in these categories: the training of unmanned aerial vehicles (Baek et al. 2013), self-driving cars (Mirchevska et al. 2018), medical applications (Jonsson 2018), Human-Robot interaction (Chanel et al. 2020). Several environments are so complex that a direct formulation of a model based on mere intuition is inappropriate and unsafe because, depending on the task, any mistake made by the agent can lead to catastrophic aftermaths. It is therefore necessary to infer the world dynamics from a batch of previously collected experiences. The said data set should be *large* and *diverse* enough for allowing useful information extraction.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The process of learning an optimal policy can be mathematically formalized as the resolution of a Markov Decision Process (MDP) if the state of the system can be considered as fully observable and the action effects are non necessarily deterministic. This paper addresses the problems linked to the resolution of MDPs starting from a single batch of collected experiences by writing up a summary of the state-of-the-art methods on offline learning and planning, and outlining their pros and cons. When the data set is fixed the distributional shift between the true, unknown, underlying MDP and its best data-driven estimate can be non negligible and lead, on resolution, to bad performing policies. This discrepancy can be seen tightly linked to the uncertainty we possess about the model. Several offline learning baselines try to handle this issue or by constraining the policy or by reshaping the reward taking into account a local quantification of the said epistemic uncertainty and hence adapting the classical resolution paradigms (Levine et al. 2020).
The document will be structured as follows:

1. Firstly, a recap of MDP resolution with MDP planning algorithms, but also with reinforcement learning (RL) algorithms is proposed.
2. Then, an intuitive description of offline model learning and batch RL is presented.
3. And finally, a discussion is provided with comments on the theoretical guarantees for the performance of the listed baselines and suggestions for further improvements in this field, like resorting to Generative Adversarial Networks (GANs) to better estimate the underlying distributions.

1 A Review of MDPs

An MDP is formally defined as a tuple $M \stackrel{\text{def}}{=} (S, A, T, r, \gamma, \mu_0)$ where S is the set of states, A the set of actions, $T : A \times S \times S \to [0, 1]$ is the state transition function defining the probability that dictates the evolution from $s \in S$ to $s' \in S$ after taking the action $a \in A$, $R : A \times S \to [R_{min}, R_{max}]$ with $R_{max}, R_{min} \in \mathrm{R}$ and

$R_{max} > R_{min}$ is the reward function that indicates what the agent gains when it selects action $a \in A$ and the system state is $s \in S$, $\gamma \in [0, 1)$ is called the discount factor and $\mu_0 : S \to [0, 1]$ is the initial probability distribution over states $s \in S$ at time $t = 0$. A policy is defined as a function that maps states to actions, such as $\pi : A \times S \to [0, 1]$; $\pi(a|s)$ can be interpreted as the probability of taking action $a \in A$ when being in the state $s \in S$. Time evolution is discrete and at every time step the agent observes the system, acts on the environment and earns a reward. The following definitions refer to an MDP with discrete A and S but they can be straightforwardly rearranged to address MDP with continuous states and actions spaces.

Solving an MDP amounts to finding a policy π^* which, $\forall s \in S$, maximizes the value function:

$$V_M^\pi(s) \stackrel{\text{def}}{=} \mathrm{E}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim T(\cdot|s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right].$$

The value function can also be expressed recursively as the fixed point of the Bellman operator:

$$V_M^\pi(s) = \sum_{a \in A} \pi(a|s) \left(R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a) V_M^\pi(s') \right).$$

We define also the Q-value function:

$$Q^\pi(s, a) \stackrel{\text{def}}{=} R(s, a) + \gamma \sum_{s' \in S} T\left(s' \mid s, a\right) V_M^\pi(s')$$

and we notice that $V_M^\pi(s) = \mathrm{E}_{a \sim \pi(\cdot|s)} \left[Q^\pi(s, a) \right]$.

Resolution Schemes

With basic planning algorithms like Value or Policy Iteration where the contraction property of the Bellman operator is exploited, one can compute a value V and a policy π that iteratively converge to V^* and π^*, respectively (Sutton and Barto 1998; Mausam and Kolobov 2012). These algorithms require to store in memory the whole state space. However, the application of the Bellman operator demands that all the functions that compose the MDP are known. What can we do, for instance, if the transition function is unknown?

Model-free Approaches In such scenarios, temporal difference (TD) schemes like *Q-learning* (Watkins and Dayan 1992) can be applied. In Q-learning the Q-function is computed iteratively by minimizing the TD error using sampled transitions (s, a, r, s'). Q-learning is a model free RL algorithm because 1) it does not require an a-priori knowledge of the model, 2) it exploits a growing batch of sampled experiences. Another popular model free approach is based on Policy Gradients (Williams 1992) which maximizes an estimate of the value function with respect to the policy where the expected value over the transition distribution is replaced by sampled transitions.

Policy Gradients methods serve as the base for the Actor-Critic architecture by which the variance of the gradient with respect to the policy is reduced either by replacing the cumulative reward with an estimate of the Q-value or by subtracting from it an estimate of the value function (Sutton et al. 1999). The module that compute Q-value and value function estimates is called the Critic and the one that computes π thanks to the Policy Gradient method is named the Actor.

Model-based Approaches Another route to follow is that of first using a batch of previously sampled experiences to obtain both $\hat{T}$ and $\hat{R}$ which are respectively estimates of the transition function T and of the reward function R of the unknown MDP. And then, to directly execute a planning algorithm or to use $\hat{T}$ as a generative model of new fictitious experiences (s, a, r, s') and subsequently apply a model-free technique using the new data set augmented with the artificial transitions. Such a scheme was first mentioned in the Dyna-Q algorithm (Sutton and Barto 1998), even though it was prescribed to be used in combination with periodical further explorations of the environment. Regarding the MDP planning literature, techniques which exploit heuristic guided trial-based solving have been created to address large finite state spaces. Amongst all *Upper Confidence bounds applied to Trees* (UCT) and more recently, PROST (Kocsis and Szepesvári 2006; Keller and Eyerich 2012). Such algorithms could be applied to the estimated MDP generative model, using $\hat{T}$ and $\hat{R}$.

Notice that when new data is generated the optimal policy of the MDP defined by $\hat{T}$ can be different from the one of the original MDP since they really define two different decision processes. The authors of the work (van Hasselt, Hessel, and Aslanides 2019) questioned the advantage of using a model to generate fictitious data over working directly on the batch with model free algorithms. Model-based techniques are normally more data efficient than model free competitors since probably the model learning stage can capture more easily the characteristics needed to estimate the Q-value and value function. However, in that paper it is empirically displayed that an appropriately fine-tuned model-free algorithm can achieve a superior data efficiency performance.

Function Approximators

When the state or the action space has the cardinality of the continuum a tabular representation of policies and value functions is unfeasible. If the states are characterized by continuous feature vectors, planning algorithms are not applicable without a preliminary discretization. (Munos and Moore 2002) proposes a variable resolution discretization of S assuming that a perfect generative model is available. The latter enables to split recursively the feature space where more control is required preventing an unforgivable loss of resolution in the transition function of the aggregate MDP. Even though the variable resolution scheme provides a more efficient splitting criterion than an uniform grid, it does not manage to escape from the curse of dimensionality. Conversely, some promising attempts have been fulfilled in the case of a continuous action space and finite state space (Mansley, Weinstein, and Littman 2011). Resorting to the Universal Approximation Theorem (Csáji 2001) it has been found practical to use function approximators in order to estimate the policy and the value functions. The increase in computational power of the last decade gave birth to a rich

community of scientists and engineers who use function approximators with thousands of parameters such as neural networks. Model-free algorithms using neural networks are Deep Q-learning Networks (Mnih et al. 2015), Policy Gradients (Williams 1992; Sutton et al. 1999) and their subsequent improvements (Hessel et al. 2018) (Schulman et al. 2015; Mnih et al. 2016; Barth-Maron et al. 2018; Haarnoja et al. 2018) that led to development of agents which achieved better than human performances in games like Go (Silver et al. 2016), Chess (Silver et al. 2017), and also some video games of the ATARI suite.

In model-based settings, approximators usually need the specification of a prior distribution for T which is often chosen Gaussian since these algorithms are usually applied to problems driven by a deterministic dynamics or to problems whose intrinsic stochasticity can be thought being induced by a Gaussian distribution in some latent space (Deisenroth and Rasmussen 2011; Chua et al. 2018; Hafner et al. 2019; Kaiser et al. 2020; Hafner et al. 2020). The latter is a strong limitation of these approaches since, more often than not, taking decisions under uncertainty amounts to deal with multi-modal transition distributions that would be poorly described by a normal distribution.

2 Single Batch Learning

As we have stated in the introduction, learning from a single batch of collected experiences is a necessity of compelling importance for a safe, cost limited and data efficient development of intelligent agents. We will see that several algorithms which constrain the optimal policy obtained with RL or planning tools to one that does not drive the agent to regions of $S \times A$ that have been poorly sampled in the data set lead to more effective policies than the one used to collect the batch. Usually the results are also better than the one obtained with a policy that has been calculated by straightforwardly applying the schemes listed in Section 1.

The utilization of function approximators to estimate the value functions using a single batch requires theoretical delicacy since many convergence guarantees do not stand. In the paper (Chen and Jiang 2019) the authors realized that usually two fundamental assumptions are implicitly required in order for the following algorithms to work:

1. mild shifts between the distributions of the real world and the one inferred from the data in the batch,
2. conditions on the class of candidate value-functions stronger than just the membership of the optimal Q-value to this function class.

Related to those points, (Chen and Jiang 2019) explores the notion of *concentratability* coefficient (Munos 2003), which is hereafter recalled.

$\forall (s,a) \in S \times A$ and $\forall h \geq 0, \forall \pi$:

$$\frac{P\left(s_h = s, a_h = a | s_0 \sim \mu_0, \pi\right)}{\mu_{\mathcal{B}}(s,a)} \leq C,$$

where $\mu_{\mathcal{B}}$ is the probability distribution that generated the batch assuming that the transitions are independent and identically distributed. The existence of C ensures that any attainable distribution of state-action pairs is not too far away from $\mu_{\mathcal{B}}$. The main result reported in (Chen and Jiang 2019) is that not constraining C precludes sampled efficient learning even with "the most favourable" data distribution $\mu_{\mathcal{B}}$.

Rather than focusing on the practical implementation of the different methodologies, which as we will see is often approximate due to the intractability of the terms present in the derived theoretical bounds, we aim to perform a simple yet comprehensible adumbration of the ideas that support their development. With this in mind we are going to neglect implementation related technicalities and sketch the theoretical foundations of single batch learning algorithms.

Constraints for Model-free Algorithms

The first successful applications of offline learning for planning and control with function approximators are very recent (Fujimoto, Meger, and Precup 2019; Fujimoto et al. 2019). In these works the authors showed that performing Q-learning to solve a finite state MDP using a fixed batch $\mathcal{B}$ leads to the optimal policy $\pi^*_{\mathcal{B}}$ for the MDP $M_{\mathcal{B}}$ whose transition function is the most likely one with respect to the transitions $(s,a,r,s') \in \mathcal{B}$. More often than not, the optimal policy for $M_{\mathcal{B}}$ performs poorly in the true environment. The discrepancy between the transition function of the original process and the one learnt from the batch will be the key element of the following discussion.

Indeed, the *extrapolation error* for a given policy π:

$$\epsilon(s,a) \stackrel{\text{def}}{=} Q^{\pi}(s,a) - Q^{\pi}_{\mathcal{B}}(s,a)$$

defined as the difference between the Q-value function of the real MDP and the Q-value function of the most likely MDP learnt from the batch could be computed with an operator similar to the Bellman's one:

$$\begin{aligned} \epsilon(s,a) \quad = \quad & \gamma \sum_{s' \in S} \Big[\big(T(s'|s,a) - T_{\mathcal{B}}(s'|s,a) \big) V^{\pi}_{\mathcal{B}}(s') \\ & + T(s'|s,a) \sum_{a' \in A} \pi(a'|s') \epsilon(s',a') \Big] \end{aligned}$$

The authors noticed that the extrapolation error is a function of divergence between the true transition distribution and the one estimated from the batch along with the error at succeeding states. Their idea is then to minimize the error by constraining the policy to visit regions of $S \times A$ where the transition distributions are similar. Henceforth, they modified Q-Learning and Deep Q-Learning algorithms to force the new "optimal" policy to be *not so distant* from the one that was used during the collection of the batch. They train a generator network that gets as an input a state s to estimate the batch generating policy and then allow for a small perturbation around it. The magnitude of the perturbation is an hyperparameter. In this way, they obtain a policy that always achieves better performance than the one used during the batch collection. This algorithm is called Batch Constrained Q-Learning (BCQ).

In a subsequent work, it has been shown that the error in the estimation of the Q-value with neural networks is generated by the back-up of poor estimates of the Q-value that

comes from regions $S \times A$ that were badly sampled in $\mathcal{B}$ (Kumar et al. 2019). To contrast the accumulation of the error, the authors developed the Bootstrapping Error Accumulation Reduction (BEAR) algorithm which, exploiting the notion of distribution concentratability, manages to constrain the improved policy to the support of the one that generated the batch. Strictly speaking, they blame the back-up of Q-value estimates of states with Out Of Distribution (OOD) actions for increasing the extrapolation error. They should blame for OOD state-action transitions, but in offline Q-learning the Q function is computed only at states that are in the replay buffer. This constraint is softer than the one imposed by BCQ and it has been showed to provide better results.

When BEAR and BCQ are applied on batches generated with a random policy, they can eventually perform worse than Deep Q-learning naively applied using the batch as a fixed replay buffer. In these cases, if the data set is big enough, there are not many OOD actions (Kumar et al. 2019). Probably, enforcing a constraint as done in BEAR and BCQ will provide a too little window for policy improvement.

In the same year, yet another inspiring paper about offline reinforcement learning was published (Wu, Tucker, and Nachum 2019). The authors of the latter showed that any policy constraining approach like BCQ, BEAR and KL-Control (Jaques et al. 2019) can be obtained as a special case of their Behaviour Regularized Actor Critic (BRAC) algorithm.

The general idea is to either 1) penalize the value function estimated by the actor or 2) regularize the policy generated by the critic by a distance in probability space between the batch collector policy $\pi_{\mathcal{B}}$ and the currently evaluated one, as:

$$V_D^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t \mathrm{E}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim T(\cdot|s_t, a_t)}} \Big[r(s_t, a_t) - \alpha D\big(\pi(\cdot|s_t), \pi_{\mathcal{B}}(\cdot|s_t)\big) | s = s_0 \Big]$$

where D is a distance function in probability space (e.g. Kernel MMD, Kullback-Leibler, Total Variation, Wasserstein, etc) and α is an hyperparameter. While the policy regularized learning objective of the actor maximizes the following criterion:

$$\mathop{\mathrm{E}}_{(s,a,r,s') \sim \mathcal{B}} \left[\mathop{\mathrm{E}}_{a \sim \pi(\cdot|s)} [Q(s,a)] - \alpha D\big(\pi(\cdot|s), \pi_{\mathcal{B}}(\cdot|s)\big) \right]$$

Their results showed that overall value penalization works better than policy regularization and the distance D that provides the best performing policy is the Kullback-Leibler divergence.

In their practical implementation both BCQ and BEAR use an average Q-value over an ensemble of Q-value networks to reduce the prediction error. In BRAC, the minimum Q-value over an ensemble of Q-value networks is used.

Extrapolation Error Reduction with Random Ensembles of Q-value Networks

In parallel to the previous studies, the authors of (Agarwal, Schuurmans, and Norouzi 2019) empirically demonstrated that the stabilization of Deep Q-learning networks using a single data set can be achieved by training at the same time a multitude of different Deep Q-value Networks with their weights differently initialized. During training the final estimate of the Q-value will be a normalized *random* linear combination of the output of the intermediate Q-functions, while in the end they will just consider as the final Q-value estimate their average. The linear combination step is equivalent to a Dropout layer in a neural network. By doing so the final output will be stabler and if a network will be more affected than another by an OOD action back-up, the final average over the random ensemble will likely mitigate this error. The authors called this neural network architecture Random Ensemble Mixture (REM).

Generative model learning for Model-based approaches

Steps forward in the development of model based RL using a single batch have been done respectively in MOPO and MOReL (Yu et al. 2020; Kidambi et al. 2020). In both cases, a generative model is first learnt from the batch and then used to create new transitions. On the augmented data set then a model-free algorithm is applied. In this fashion, since we can use the generative model to "explore" the $S \times A$ space, the error in the Bellman back-up will not be induced directly by ill sampled regions but by the *epistemic error* of the model.

Intuitively, the more chaotic is the underlying system, the greater will a trajectory generated by the learnt model diverge from a real one given the very same starting state distribution and an identical sequence of actions to apply. Broadly speaking, as described below, the two methods build a penalized (MOPO) and pessimistic (MOReL) MDP whose optimal policies are encouraged to visit regions of $S \times A$ where the epistemic error is expected to be little (MOPO), or areas that would be likely to be sampled by the same distribution dynamics that generated the batch (MOReL).

Model Error Penalized MDP In MOPO, defining as $\eta_M[\pi] \stackrel{\text{def}}{=} \mathrm{E}_{s \sim \mu_0}\left[V_M^{\pi}(s)\right]$ as the performance of a policy for the MDP M, a theoretical bound for $\eta_M[\pi] - \eta_{\hat{M}}[\pi]$ is recovered. In particular, they show that:

$$\eta_M[\pi] \geq \mathrm{E}_{(s,a) \sim \rho_{\hat{M}}^{\pi}} \Big[r(s,a) - \frac{\gamma}{1-\gamma} \max_{s'} \left(V_M^{\pi}(s')\right) \cdot D_{TV}\left(T(\cdot|s,a), \hat{T}(\cdot|s,a)\right) \Big] = \eta_{\tilde{M}}[\pi] \quad (1)$$

where $\rho_{\hat{M}}^{\pi}$ is the discounted state-action distribution of transitions along the Markov Chain induced by $\hat{T}$ and π starting from the initial state distribution μ_0.

The right hand side is the performance of the MDP $\tilde{M}$ whose dynamics is driven by $\hat{T}$, but with reward function penalized by a term which is directly proportional to the

total variation distance between the true and the inferred transition functions. Since both D_{TV} and $\max_{s'} V^{\pi}_{M}(s')$ are unknown, in the practical implementation the penalty is replaced by $\lambda u(s,a)$ where λ is an hyperparameter and $u : S \times A \rightarrow [0, +\infty)$ such that $u(s,a) \geq D_{TV}(T(\cdot|s,a), \hat{T}(\cdot|s,a)) \; \forall (s,a) \in S \times A$. Therefore finding the optimal policy for the penalized MDP amounts to obtaining the policy that maximizes the lower bound on η_M.

Notwithstanding, we believe that their bound is greatly dependent on the choice of a proper hyperparameter λ and function u, which is not trivial for stochastic MDPs while it can be appropriately approximated by the covariance of a Gaussian Process for deterministic environments like the one used as test-cases in their paper. Moreover, if the penalization is too big the lower threshold will be likely of little use. Imagine the extreme situation where $\forall (s,a) \in S \times A$, $r > 0$ and $r - \lambda u < 0$. In this case $\eta_M[\pi] > 0 \; \forall \pi$ trivially, while, calling $\tilde{M}$ the reward penalized MDP with dynamics driven by $\hat{T}$, $\eta_{\tilde{M}}[\pi] < 0$. Therefore, maximizing the bound will not necessarily lead to a policy that works better than chance on the real MDP.

In (Yu et al. 2020) the performance of a policy is defined as the expected value of V^{π}_{M} over the starting state distribution μ_0. This starting state distribution is then interpreted as the distribution of states in the batch. However, the latter could be much different from the true starting state distribution if the batch is of modest size.

Therefore a more robust definition could be $\eta_M[\mu^{\pi}_{M}, \pi] \overset{\text{def}}{=} \mathrm{E}_{s \sim \mu^{\pi}_{M}}[V^{\pi}_{M}(s)]$, where μ^{π}_{M} is the stationary distribution of states (if it exists) for the MDP M with dynamics dictated by the policy π. Using this new definition, the lower bound on $\eta_M[\mu_M, \pi]$ acquires an extra term dependent on the difference $\Delta^{\pi}_{\hat{M},M}(s) = \mu^{\pi}_{\hat{M}}(s) - \mu^{\pi}_{M}(s)$. Indeed, the performance of a policy in the true MDP can be expressed as:

$$\begin{aligned} \eta_M\left[\mu^{\pi}_{M}, \pi\right] &= \mathrm{E}_{s \sim \mu^{\pi}_{M}}\left[V^{\pi}_{M}(s)\right] \\ &= \mathrm{E}_{\mu^{\pi}_{M}}\left[V^{\pi}_{M}(s)\right] - \int_S d\mu(s) \Delta^{\pi}_{\hat{M},M}(s) V^{\pi}_{M}(s) \end{aligned}$$

where, $d\mu(s)$ is a measure over the state space. It is then possible to obtain the same bound of Equation (1) but with the extra term dependent on the integral over the state space of the discrepancy between the stationary distributions:

$$\begin{aligned} \eta_{\hat{M}}\left[\mu^{\pi}_{\hat{M}}, \pi\right] - \eta_M\left[\mu^{\pi}_{M}, \pi\right] = \\ = \mathrm{E}_{s \sim \mu^{\pi}_{\hat{M}}}\left[V^{\pi}_{\hat{M}}(s)\right] - \mathrm{E}_{s \sim \mu^{\pi}_{M}}\left[V^{\pi}_{M}(s)\right] = \\ = \mathrm{E}_{s \sim \mu^{\pi}_{\hat{M}}}\left[V^{\pi}_{\hat{M}}(s) - V^{\pi}_{M}(s)\right] + \int_S d\mu(s) \Delta^{\pi}_{\hat{M},M}(s) V^{\pi}_{M}(s) \end{aligned}$$

where the expected value over the distribution $\mu^{\pi}_{\hat{M}}$ is similar to the one computed in (Yu et al. 2020) but with $\mu_0 = \mu^{\pi}_{\hat{M}}$. Therefore the right hand side can be bounded from below:

$$\begin{aligned} \eta_{\hat{M}}\left[\mu^{\pi}_{\hat{M}}, \pi\right] - \eta_M\left[\mu^{\pi}_{M}, \pi\right] \leq \frac{\gamma}{1-\gamma} \max_{s'}\left(V^{\pi}_{M}(s')\right) \\ \cdot \mathrm{E}_{(s,a) \sim \rho^{\pi}_{\hat{M}}}\left[D_{TV}\left(T(\cdot|s,a), \hat{T}(\cdot|s,a)\right)\right] \\ + \int_S d\mu(s) \Delta^{\pi}_{\hat{M},M}(s) V^{\pi}_{M}(s) \end{aligned}$$

where $\rho^{\pi}_{\hat{M}}$ now is the discounted state-action distribution of transitions along the Markov Chain induced by $\hat{T}$ and π starting from the stationary distribution $\mu^{\pi}_{\hat{M}}$.
Exploiting the definition of penalized MDP $\tilde{M}$:

$$\eta_M\left[\mu^{\pi}_{M}, \pi\right] \geq \eta_{\tilde{M}}\left[\mu^{\pi}_{\hat{M}}, \pi\right] - \int_S d\mu(s) \Delta^{\pi}_{\hat{M},M}(s) V^{\pi}_{M}(s)$$

The latter can again be bounded from above by plugging in the absolute value of Δ and the max of V over the state space:

$$\begin{aligned} \eta_M[\mu^{\pi}_{M}, \pi] \quad \geq \quad & \eta_{\tilde{M}}[\mu^{\pi}_{\hat{M}}, \pi] \\ & - \max_{s'} V^{\pi}_{M}(s') \int_S d\mu(s) |\Delta^{\pi}_{\hat{M},M}(s)| \end{aligned}$$

It is remarkable that now the optimal policy for $\tilde{M}$ does not necessarily maximizes the bound. Assuming that our algorithm is monotonically improving the policy, it could be then convenient to stop it earlier and settle for a sub-optimal policy which in turn maximizes the bound. It's all about balancing the trade-off between the optimality condition for $\tilde{M}$ and the discrepancy within the stationary distributions. The newly added term is unfortunately intractable due to the lack of knowledge about the MDP.

In the implementation of MOPO new trajectories are generated starting from states already present in the batch up to h following time steps. Ablation experiments have shown that the roll-out horizon h is indeed required to obtain good results. We suspect that the state distributional shift that was neglected is to be blamed for the occurrence of the behaviour. Generating data that are not so far away from ones in the batch prevents the accumulation of model error, but this theoretical aspect, even if already mentioned in (Janner et al. 2019), should not affect the bound that aims to be valid on any uncertainty penalized MDP independently of other factors.

Pessimistic MDP The authors of MOReL define an MDP with an extra absorbing state y. The state space of the pessimistic MDP is $\tilde{S} = S \cup \{y\}$, while the transition function

$$\tilde{T}(s'|s,a) = \begin{cases} \delta_{s',y} & \text{if } D_{TV}\left(\hat{T}(\cdot|s,a), T(\cdot|s,a)\right) > \theta, \\ \delta_{s',y} & \text{else if } s = y, \\ \hat{T}(s'|s,a) & \text{otherwise.} \end{cases}$$

The reward function R is identical to the original one except for y: $R(y,a) = -\kappa \; \forall a \in A$.
θ is a freely chosen threshold and $\kappa >> 0$ is a penalty. Essentially if the model error is greater than θ the agent will end up for sure in the strongly penalized absorbing state. Therefore any optimal policy for the pessimistic MDP will try to avoid transitions for which the model error is high. The optimal policy $\hat{\pi}$ when applied on the real MDP bounds from above the performance of the optimal policy of the true MDP.

$$\begin{aligned} \frac{4R_{max}}{1-\gamma}\left(\zeta(\mu_0, \hat{\mu}_0, T, \hat{T}) + \mathrm{E}\left[\gamma^{T^{\pi^*}_{U}}\right]\right) \\ \geq \eta_M[\mu_0, \pi^*] - \eta_M[\mu_0, \hat{\pi}] \end{aligned}$$

with,

$$\zeta(\mu_0, \hat{\mu}_0, T, \hat{T}) = D_{TV}\left(\mu_0, \hat{\mu}_0\right) + \frac{\gamma}{1-\gamma} \max_{s,a} D_{TV}(\hat{T}, T)$$

The two performances are similar if the D_{TV} between the real starting state distribution and the one inferred from the batch is negligible, if the maximum model error is little, and also, if the expected value of the first hitting time of the absorbing state while applying the policy π^* in the pessimistic model $\gamma^{T_{\mathcal{U}}^{\pi^*}}$ is small.

Again, in a practical implementation the choice of good estimators of the epistemic error, of the distributional distance between starting states, and also, of the first hitting time is of crucial importance. In the large batch regime the authors neglect the first two terms and focus only on the expectation of the first hitting time which can be bounded from the above by the *discounted* distribution of visits to $(s, a) \in \mathcal{U}$, where $\mathcal{U}$ represents the *unknown* state-action pairs that lead to the absorbing state, when applying $a \sim \pi^*$.The late distribution can be in turn bounded by a term proportional to the support mismatch of the distribution of states that were never sampled in the original data set.

3 Discussion

Theoretical bounds and function approximators

The theoretical bounds which justify the creation of the previously listed algorithms rely either on a penalty or on a regularization term proportional to a sort of uncertainty that obnubilates our knowledge about the underlying system. Sometimes the penalty is expressed as an estimate of the epistemic model error, other times as a difference between starting or stationary state distributions, finally it can be quantified as Out Of Distributions (OOD) state-action pairs with respect to the policy used during the collection of the data set.

The penalty or regularization term is often proportional to an upper bound of the value function or to a free hyperparameter. As we have seen the latter statement implies that when this constant is too big the intractable performance of a policy on the real MDP is bounded by a tractable term which unfortunately will be of little use.

Only REM stabilizes the accumulation of the error in Q-learning thanks to a constraintless weighted random ensemble average. Despite its nicety, the stabilization is not properly a goal-oriented correction to the deviation of the optimal Q-value estimated using a single batch from the real one.

Model-based approaches also learn a function $\hat{R}$, however there is no term linked to the uncertainty in the evaluation of the reward in the batch penalized resolution scheme for offline learning algorithms. We believe that such a term proportional to the reward error is not truly necessary since we expect it to be stemmed from the same regions of $S \times A$ that are badly sampled in the data set $\mathcal{B}$ and considering that the penalization is already applied on the reward or value functions.

Finding a proper estimator of the *errors* is not trivial. The algorithms were often tested on deterministic environments where a reasonable estimator of the model error can be achieved by the maximal variance in between an ensemble of different Gaussian models. Since it's reasonable to expect that the model error will be high in regions that were ill-sampled in the batch, another way to measure it could be getting an estimate of the probability that a given (s, a) could have been generated by the same process that gave birth to the batch. Therefore estimating the probability distribution of (s, a, s') in the true MDP with policy $\pi_{\mathcal{B}}$ is a priority.

The most practical way of learning a probability distribution function without a prior could be to use a GAN (Goodfellow et al. 2014). A GAN is comprised of a Generator and a Discriminator. The first is a neural network which receives random noise as an input and generates an output with the same shape of the data in a training set. The second gets an input with the correct shape and provides as output a real number. While training the Discriminator tries to identify which data was present in the training batch between samples that really populate it and the output of the generator. The higher the output of the Discriminator on a sample will be, the most likely that sample will be in the batch if the Discriminator is well trained. At the same time the goal of the Generator will be that of fooling the Discriminator. The loss functions minimized during the training are peculiar of a min-max game. Sophisticated GANs architecture can use a well trained Generator to build fake samples that could fool even a human. A striking example is StyleGAN2 by NVIDIA (Karras et al. 2019) which can generate high quality dimensional pictures of people that do not really exist.

We believe that the use of a GAN's Discriminator trained on $\mathcal{B}$ to obtain an estimate of the log-likelihood of a transition (s, a, r, s') with respect to the unknown transition distribution should be a promising venue for penalizing the reward and/or the value functions with a more pertinent estimator of the distance between the true distribution of data and the one we can infer from a single batch. In this way we may be able to recover an informative quantity about the distributional shifts in a non parametric way that is independent of any possible prior and might, in principle, also work for systems driven by a stochastic time evolution. Doing so we would drop off the Gaussian assumption that has been so far used in almost all of the model-based techniques.

However, GANs have some weak spots: the training is unstable because the loss function is not convex, the procedure takes time, and they suffer from mode collapse. The latter is maybe the most problematic issue since when the unknown latent distributions is multi-modal the Generator may focus on building up samples that benefits from characteristics that are typical only of a little slice of the whole set. Since the Discriminator is trained alongside the Generator, it will learn to recognize samples that are typical of that specific mode. Several approaches to mitigate mode collapse (Ghosh et al. 2018) and training instability (Arjovsky, Chintala, and Bottou 2017) have been attempted so far, but the issues can be still considered unsolved.

Off-policy Evaluation

It is necessary to find statistically robust methods which are able of estimating how well an algorithm will run in the real

world without interacting with it. Off-policy evaluation is an active field which would require a summary of its own. Recent approaches utilize optimized versions of Importance Sampling to estimate the unknown ratio between stationary distributions of states under dynamics driven by different policies. Recent works propose to create a sort of Discriminator and optimize a min-max loss function to serve this purpose (Liu et al. 2018; Zhang et al. 2020).

The application of a min-max optimization loss function in the field strengthens our intuition that the implementation of a GAN anyhow in the estimation of the distributional shift might be useful.

Batch Quality and Size Scalability

It is of compelling necessity to develop and test the baselines in common environments using the same batches to shine a light onto the change in performance of the different paradigms when the quality (and the variety) and the size of the available data increase. D4RL (Datasets for Deep Data-Driven Reinforcement Learning) is a collection of data sets recorded using policies of different qualities (random, medium, expert) on the typical benchmarking environments used by the RL community (OpenGym, MuJoCo, Atari etc.) (Fu et al. 2020). However, the offline learning community has yet to settle to the use of a common pipeline for benchmarks. The results achieved by MOReL using D4RL are reported in Table 1, for comparison with different baselines examine the reference (Kidambi et al. 2020). Independently of the quality of the batch we notice an improvement in the performance, expressed as the average cumulative reward over a sequence of trajectories, of the optimal policy for the pessimistic MDP when evaluated in the true environment.

The results achieved with MOPO are reported in Table 2. MOPO performs better than all previous baselines on randomly generated batches and on data sets which consist of the full replay buffer of a Soft-Actor Critic (SAC) trained partially up to an environmental specific performance threshold. Surprisingly, on batches generated with the sub-optimal trained SAC the best baselines are BRAC with value function penalty and BEAR. The main difference between the last two types of data sets is that while the latter is generated with a fixed policy, the previous one is a collection of transitions gathered with a mixture of differently performing policies. When the sub-optimal policies are not so bad, it seems reasonable to just slightly modify them to obtain better results, hence BEAR and BRAC looks like viable methods. However, when the overall batch policy is not so good, constraining the reward with respect to the model error (and the transitions close to the ones present in the batch up to a roll-out horizon) can be more fulfilling.

As mentioned also by the authors of BRAC, their algorithm when applied to small data sets becomes more susceptible to the choice of the hyperparameters. This is probably because on small data set the distributional shift / model error can become significant. It is crystal clear that the field needs better theoretical foundations and better algorithms in order to learn more safe and performing policies from small batches collected with strategies of any quality, even uniform random ones.

Environment	Pure-Random	Pure-Partial
Hopper-v2	2354 ± 443 (20)	3642 ± 54 (1376)
HalfCheetah-v2	2698 ± 230 (-638)	6028 ± 192 (4198)
Walker2d-v2	1290 ± 325 (-7)	3709 ± 159 (1463)
Ant-v2	1001 ± 3 (-263)	3663 ± 247 (1154)

Table 1: Average cumulative return of the policy obtained with MOReL as reported in (Kidambi et al. 2020). A Pure-Partial policy is a partially trained suboptimal policy. The number between parentheses is the average cumulative reward with the batch collecting policy. All results are averaged over 5 random seeds.

4 Conclusions

In this paper we have examined the state-of-the-art RL and planning algorithms motivated by the necessity to exploit their application to improve offline learning using a single batch of collected experiences. This is challenging problem of crucial importance for the development of intelligent agents. In particular, when the interaction of such agents with the environment is expensive, risky or unpractical. Our goal was that of providing to the reader a self-contained summary of the general ideas that flow behind the main topic. For simplicity, we focused on MDPs but once the listed difficulties will be addressed we aim to extend the discussion to Partially Observable MDPs which are a more appropriate object to describe the interaction of agents in partial observable environments. We started with a recap of MDPs and resolution schemes. Then we presented the single batch learning and planning problem. Our main contribution is an outline of model-free and model-based batch RL algorithms while providing comments on size scalability, efficiency and on the usefulness of theoretical bounds. In particular, we proposed an improvement of the definition of performance of the value function following a specific policy that led us to believing that a sub-optimal policy for a reward uncertainty penalized MDP can be better than the optimal one when applied in the true environment. Secondly, we analyzed the penalization introduced in all sorts of offline-learning algorithms. We showed that if the coefficients multiplying the distributional shift estimator are too big then the theoretical threshold which bounds the performance of the policy applied in the real world is always respected, and therefore of little practical utility. We also advised the future implementation of GANs for a better estimate of distributional shifts and model errors. Indeed, estimators that optimizes a min-max loss function give hint that this might be a viable solution.

References

Agarwal, R.; Schuurmans, D.; and Norouzi, M. 2019. An optimistic perspective on offline reinforcement learning. *arXiv: Learning*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *ArXiv* abs/1701.07875.

Baek, S.; Kwon, H.; Yoder, J.; and Pack, D. 2013. Optimal path planning of a target-following fixed-wing uav using sequential decision processes. 2955–2962.

Data set type	Environment	Batch Mean	Batch Max	SAC	BEAR	BRAC-vp	MBPO	MOPO
random	halfcheetah	-303.2	-0.1	3502.0	2885.6	3207.3	3533.0	**3679.8**
random	hopper	299.26	365.9	347.7	289.5	370.5	126.6	**412.8**
random	walker2d	0.9	57.3	192.0	307.6	23.9	395.9	**596.3**
medium	halfcheetah	3953.0	4410.7	-808.6	4508.7	**5365.3**	3230.0	4706.9
medium	hopper	1021.7	3254.3	5.7	**1527.9**	1030.0	137.8	840.9
medium	walker2d	498.4	3752.7	44.2	1526.7	**3734.3**	582.5	645.5
mixed	halfcheetah	2300.6	4834.2	-581.3	4211.3	5413.8	5598.4	**6418.3**
mixed	hopper	470.5	1377.9	93.3	802.7	5.3	1599.2	**2988.7**
mixed	walker2d	358.4	1956.5	87.8	495.3	44.5	1021.8	**1540.7**
med-expert	halfcheetah	8074.9	12940.2	-55.7	6132.5	5342.4	926.6	**6913.5**
med-expert	hopper	1850.5	3760.5	32.9	109.8	5.1	**1803.6**	1663.5
med-expert	walker2d	1062.3	5408.6	-5.1	1193.6	**3058.0**	351.7	2527.1

Table 2: Results for D4RL datasets as reported in (Yu et al. 2020). Each number is the average undiscounted return of the policy at the last iteration of training, averaged over 3 random seeds. Data set types depend on the policy used to collect the batch: random (random policy), medium (suboptimally trained agent with a Soft Actor-Critic), mixed (adoperate as batch the replay buffer use to train a Soft-Actor Critic until an envirnomental specific threshold is reached), medium-expert (mix of an optimal policy and a random or a partially trained one). SAC column stands for a Soft Actor Critic (model-free) agent. BRAC-vp is the version of BRAC with the value penalty. MBPO is the vanilla model-based algorithm described in (Janner et al. 2019). Check the reference (Yu et al. 2020) for details about the hyperparameters.

Barth-Maron, G.; Hoffman, M. W.; Budden, D.; Dabney, W.; Horgan, D.; Dhruva, T.; Muldal, A.; Heess, N. M. O.; and Lillicrap, T. P. 2018. Distributed distributional deterministic policy gradients. *ArXiv* abs/1804.08617.

Chanel, C. P. C.; Roy, R. N.; Dehais, F.; and Drougard, N. 2020. Towards mixed-initiative human–robot interaction: Assessment of discriminative physiological and behavioral features for performance prediction. *Sensors* 20:296.

Chen, J., and Jiang, N. 2019. Information-theoretic considerations in batch reinforcement learning. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 1042–1051. PMLR.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *ArXiv* abs/1805.12114.

Csáji, B. C. 2001. Approximation with artificial neural networks.

Deisenroth, M. P., and Rasmussen, C. E. 2011. Pilco: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv* abs/2004.07219.

Fujimoto, S.; Conti, E.; Ghavamzadeh, M.; and Pineau, J. 2019. Benchmarking batch deep reinforcement learning algorithms. *ArXiv* abs/1910.01708.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. volume 97 of *Proceedings of Machine Learning Research*, 2052–2062. Long Beach, California, USA: PMLR.

Ghosh, A.; Kulharia, V.; Namboodiri, V. P.; Torr, P. H. S.; and Dokania, P. K. 2018. Multi-agent diverse generative adversarial networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 8513–8521.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*.

Hafner, D.; Lillicrap, T. P.; Fischer, I. S.; Villegas, R.; Ha, D. R.; Lee, H.; and Davidson, J. 2019. Learning latent dynamics for planning from pixels. In *ICML*.

Hafner, D.; Lillicrap, T. P.; Ba, J.; and Norouzi, M. 2020. Dream to control: Learning behaviors by latent imagination. *ArXiv* abs/1912.01603.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. *ArXiv* abs/1710.02298.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. In *NeurIPS*.

Jaques, N.; Ghandeharioun, A.; Shen, J. H.; Ferguson, C.; Lapedriza, À.; Jones, N. J.; Gu, S.; and Picard, R. W. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *ArXiv* abs/1907.00456.

Jonsson, A. 2018. Deep reinforcement learning in medicine. *Kidney Diseases* 5:1–5.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; Sepassi, R.; Tucker, G.; and Michalewski, H. 2020. Model-based reinforcement learning for atari. *ArXiv* abs/1903.00374.

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2019. Analyzing and improving the image quality of stylegan. *ArXiv* abs/1912.04958.

Keller, T., and Eyerich, P. 2012. Prost: Probabilistic planning based on uct. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *ICAPS*. AAAI.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. Morel. *ArXiv* abs/2005.05951.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. volume 2006, 282–293.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 11784–11794.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv* abs/2005.01643.

Liu, Q.; Li, L.; Tang, Z.; and Zhou, D. 2018. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 5356–5366.

Mansley, C.; Weinstein, A.; and Littman, M. 2011. Sample-based planning for continuous action markov decision processes. In *Twenty-First International Conference on Automated Planning and Scheduling*.

Mausam, and Kolobov, A. 2012. *Planning with Markov Decision Processes: An AI Perspective*.

Mirchevska, B.; Pek, C.; Werling, M.; Althoff, M.; and Boedecker, J. 2018. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2156–2162.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518:529–533.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

Munos, R., and Moore, A. 2002. Variable resolution discretization in optimal control. *Machine Learning* 49, Numbers:291–.

Munos, R. 2003. Error bounds for approximate policy iteration. In *ICML*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–503.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv* abs/1712.01815.

Sutton, R. S., and Barto, A. G. 1998. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks* 16:285–286.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

van Hasselt, H. P.; Hessel, M.; and Aslanides, J. 2019. When to use parametric models in reinforcement learning? In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 14322–14333.

Watkins, C., and Dayan, P. 1992. Q-learning. *Machine Learning* 8:279–292.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8(3–4):229–256.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *ArXiv* abs/1911.11361.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*.

Zhang, R.; Dai, B.; Li, L.; and Schuurmans, D. 2020. Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*.