PEBL: Pessimistic Ensembles for Offline Deep Reinforcement Learning

Jordi Smit, Canmanie T. Ponnambalam, Matthijs T. J. Spaan, Frans A. Oliehoek
Delft University of Technology
j.smit-6@student.tudelft.nl, {c.t.ponnambalam, m.t.j.spaan, f.a.oliehoek}@tudelft.nl

Abstract

Offline reinforcement learning (RL), or learning from a fixed data set, is an attractive alternative to online RL. Offline RL promises to address the cost and safety implications of taking numerous random or bad actions online, a crucial aspect of traditional RL that makes it difficult to apply in real-world problems. However, when RL is naïvely applied to a fixed data set, the resulting policy may exhibit poor performance in the real environment. This happens due to over-estimation of the value of state-action pairs not sufficiently covered by the data set. A promising way to avoid this is by applying pessimism and acting according to a lower bound estimate on the value. In deep reinforcement learning, however, uncertainty estimation is highly non-trivial and development of effective uncertainty-based pessimistic algorithms remains an open question. This paper introduces two novel offline deep RL methods built on Double Deep Q-Learning and Soft Actor-Critic. We show how a multi-headed bootstrap approach to uncertainty estimation is used to calculate an effective pessimistic value penalty. Our approach is applied to benchmark offline deep RL domains, where we demonstrate that our methods can often beat the current state-of-the-art.

Introduction

Offline (batch) reinforcement learning (RL), addresses some of the key problems that make general reinforcement learning unsuitable for real-world applications such as robotics (Cabi et al. 2020), healthcare (Wang et al. 2018), recommender systems (Strehl et al. 2010), and chatbots (Pietquin et al. 2011). In offline RL, a particular core issue of standard RL is absent: the exploration-exploitation trade-off. Instead, a data set is available that contains transitions sampled from the environment that ideally provide useful information about its dynamics and the task we want to optimize. This suggests that we can find a policy that optimizes the task, given only the data set, and avoid taking additional exploratory actions online whose consequences are unknown. In safety-critical applications or anywhere ample offline data is available, offline RL is an attractive approach to automation. However, naïvely applying policy optimization to a fixed data set has been repeatedly shown in practice to produce a policy that performs very poorly on the true task (Levine et al. 2020; Fujimoto et al. 2019; Fu et al. 2020; Kim 2020; Hou et al. 2020). The fundamental issue arises in the likely case that the data has insufficient information to allow the transition and reward model to be adequately estimated, either explicitly or implicitly, in the model-free case. This issue arises from the strong bias introduced by the policies that generated the data set, rendering the data unrepresentative of the true MDP.

A theoretical analysis of the offline RL problem suggests that policies learned on fixed data will become over-optimistic, meaning they assign values to actions that are higher than the true value in areas with insufficient information (Buckman, Gelada, and Bellemare 2020). If the offline data is collected by selecting actions at random, it is more likely that the coverage of the problem space is sufficient to recover the transition and reward functions. In this case, over-estimation is less likely, but significantly more data is needed to uncover optimal behavior. In the other extreme, where data is collected according to expert demonstrations, the data is highly biased and will contain transitions from only a subset of the overall state-action space. The value attributed to parts of the state-action space that were not represented enough in the data set is likely to not only be incorrect but over-estimated. This effect occurs because over-estimations have a larger impact on the performance of the offline RL agent in the true environment (Buckman, Gelada, and Bellemare 2020). One way to bypass this issue is to constrain the learned policy to lie near the policy exhibited in the data. This has been shown to be effective (Levine et al. 2020; Fujimoto et al. 2019; Kim 2020; Fujimoto, Meger, and Precup 2019), but such an approach is limited in its ability to improve on the policy emergent in the data set. The ideal approach should be robust to many data collection strategies, ranging from random actions to expert demonstrations.

An alternative to penalizing actions that are under-represented in the data set is to penalize actions whose estimated value is highly uncertain. The penalty allows policies to deviate from the empirical policy in high information regions while closely imitating the empirical policy in low information regions. This principle, referred to as uncertainty-based pessimism, is a promising approach to producing offline RL algorithms that are robust to various data collection policies (Jin, Yang, and Wang 2020; Rashidinejad et al. 2021). The effectiveness of such an ap-

proach has been demonstrated in tabular problems, where a principled approach can be taken to uncertainty estimation (Buckman, Gelada, and Bellemare 2020). We aim to extend these findings to deep RL, where the continuous nature of the state-action space and use of function approximation means that count-based methods can no longer be used for uncertainty estimation. We propose a multi-headed bootstrap approach with randomized priors to estimating epistemic uncertainty in deep learning settings (Osband, Aslanides, and Cassirer 2018). With the inclusion of randomized priors, every ensemble member learns a different function in regions with no training data, resulting in better epistemic uncertainty estimation. With this, we derive a pessimistically-penalized offline deep reinforcement learning approach, which we call PEssimistic ensemBLe, or PEBL (pronounced "pebble"). This paper introduces two versions of PEBL, one built on Double Deep Q-Learning (Van Hasselt, Guez, and Silver 2016) and another variant based on Soft Actor-Critic (Haarnoja et al. 2018). Our implementation of these techniques is available in an open-source repository at: github.com/j0rd1smit/PEBL.

This paper first highlights related literature and describes how our work builds upon recent advances in offline deep RL. We then discuss uncertainty estimation in deep RL and the motivations for the approach used in our method. This is followed by the description of our two PEBL variants for offline deep RL, tackling both discrete and continuous action spaces. In experiments, we demonstrate that our uncertainty-based pessimistic methods achieve state-of-the-art performance, often improving on existing benchmarks. Finally, we conclude and suggest directions for future work in this area.

Background

In this section, we formalize established concepts that provide a foundation for our work.

Reinforcement Learning

In reinforcement learning, the environment or task is modeled as a Markov decision process (MDP). Formally, an MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$ containing state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, transition function $\mathcal{T}$, and a discount factor γ. An agent acting in an MDP at time step t receives a state s_t and then executes an action a_t which causes the environment to transition according to $\mathcal{T}(s_t, a_t, s_{t+1})$, a function that maps state-action pairs to a distribution over next states $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. The agent enters the next state s_{t+1}, and receives a reward r_t according to reward function $\mathcal{R}(s_t, a_t, s_{t+1})$, where $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. The solution to an MDP is a policy π that maps states to actions. The value of a policy $V^{\pi}_{s_0=s_i}$ is the sum of expected discounted rewards when following policy π from initial state s_i. An optimal policy π^* maximizes this value for every state. The action-conditioned value is referred to as $Q(s, a)$, which returns the value for a particular action in a given state.

In reinforcement learning, the transition and reward dynamics of the MDP are unknown. An agent must therefore interact with the environment in order to learn to optimize rewards. In Q-learning, a common model-free approach for tabular problems, this is done by continuously applying the Bellman update on experience sampled from the MDP (s, a, r, s') (collected by the agent) until Q converges with learning rate α (Watkins and Dayan 1992):

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a')) + (1 - \alpha)Q(s, a)$$

Deep Reinforcement Learning

In continuous state-space environments, the Q-function can no longer be represented in a table, and function approximation is required. In deep reinforcement learning, a multi-layered neural network is used to approximate Q. The most basic method applying this is Deep Q-Learning (DQN) (Mnih et al. 2013, 2015), where the TD-target y_t is calculated by the neural network (with parameters θ) is:

$$y_t = r_t + \gamma \max_{a'} Q(s', a' : \theta)$$

Double Deep Q-Learning (DDQN) Naïve applications of DQN has been known to exhibit instability due to over-estimation, thus DDQN (which originated in tabular settings) was posed as a solution to this problem (Van Hasselt, Guez, and Silver 2016). In DDQN, two Q-functions are maintained, one of which is used for action selection (parameterized by θ) and the other for evaluation (parameterized by θ'). The original DDQN algorithm calculates the target using the formula:

$$y_t = r_t + (1 - d_t) \cdot \gamma Q(s_{t+1}, \underset{a}{\operatorname{argmax}}\{Q(s_{t+1}, a; \theta)\}; \theta')$$

In this equation, r_t is the reward, and d_t is a flag indicating whether the episode terminated or not at the t-th time step.

Soft Actor-Critic (SAC) Both DQN and DDQN are suitable for discrete-action problems. Soft Actor-Critic is an algorithm that can also be applied to problems with continuous action spaces. It builds on the popular Actor-Critic approach, which splits learning the policy from the value function. In this approach, the actor learns an action distribution that maximizes the expected reward based on the feedback from the critic, while the critic learns to approximates the value function (Sutton and Barto 2018). Actor-Critic approaches are typically used in on-policy settings. However, using importance sampling, it is also possible to use this approach in an off-policy setting (Haarnoja et al. 2018). In SAC, the goal is to optimize for a trade-off of expected return and entropy of the policy. The TD-target y_t is calculated using the formula:

$$y_t = r_t + (1 - d_t) \cdot \gamma \min_{j=1,2} \{Q(s_{t+1}, \tilde{a}_{t+1}; \theta'_j)\} - \alpha \log \pi_\phi(\tilde{a}_{t+1} | s_{t+1})$$

In this equation, $\tilde{a}_{t+1}$ is sampled from the learned policy $\pi_\phi(.|s_{t+1})$, and α is the entropy trade-off parameter. Similar to DDQN, SAC tries to minimize instability issues due to over-estimations caused by the function approximation. It does this by learning two Q-value functions, parameterized by θ_1 and θ_2. It uses the lowerest Q-value estimation for its policy losses, and it is TD-target to prevent over-estimations.

Uncertainty-Aware Offline Reinforcement Learning

Buckman, Gelada, and Bellemare describe two types of offline reinforcement learning algorithms that apply pessimism: proximal (which we call policy-constrained) pessimistic algorithms and uncertainty-aware pessimistic algorithms (2020). In their experiments, the performance of four algorithmic families was demonstrated: naïve (simply applies standard reinforcement learning on the given data set), behavior cloning (copies the empirical policy followed in the data set), policy-constrained pessimistic (penalizes actions not well-represented in the data set), and uncertainty-aware pessimistic (penalizes actions with high uncertainty in the data set). A behavior cloning-based policy can only perform as well as the data collection policy. The policy-constrained pessimistic family can improve upon the data collection policy, but only slightly, as these algorithms are constrained to stay close to it. The naïve algorithmic family performs well if the data set contains sufficient exploratory data, such as in a huge randomly generated data set. However, its performance quickly deteriorates when the data set is more biased (such as that generated by expert demonstrations). Theoretically, and then in tabular experiments, it was shown that the uncertainty-aware pessimistic approach was the most robust to different types of data sets.

Related work

Uncertainty-aware approaches to reinforcement learning have been used in the traditional online setting, notably in the robust MDP framework which aims to build policies that are robust to modeling errors (Zhou et al. 1996; Givan, Leach, and Dean 2000; Nilim and El Ghaoui 2005; Wiesemann, Kuhn, and Rustem 2013). Recently, the theoretical advantages of uncertainty-aware algorithms in the offline setting have been demonstrated in tabular experiments (Buckman, Gelada, and Bellemare 2020; Rashidinejad et al. 2021) and linear function approximation experiments (Jin, Yang, and Wang 2020). We are concerned with the application of uncertainty-aware pessimism in offline deep reinforcement learning, where quantifying uncertainty is particularly difficult (Buckman, Gelada, and Bellemare 2020; Abdar et al. 2020). Recently, deep learning approaches for offline reinforcement learning have received considerable attention. Several of these advances can be viewed as members of the policy-constrained pessimistic family (Kim 2020; Kumar et al. 2020; Hou et al. 2020), where the primary differences between approaches are the choice of constraint (Buckman, Gelada, and Bellemare 2020). For example, both Batch Constrained Q-Learning (Fujimoto et al. 2019) and Pessimistic Offline Policy Optimization (POPO) (Hou et al. 2020) can be viewed as pessimistic algorithms because they are pessimistic with respect to Q-values for state-action pairs not covered by the behavior policy. However, because this pessimism is based on the observed behavior policy, they belong to the policy-constrained pessimistic family, whereas we are interested in the uncertainty-based pessimistic family and its theoretical advantages over the alternative.

Like our method, many other offline deep RL algorithms also use an ensemble of Q-functions to prevent overestimating the Bellman backup. For example, the methods BEAR (Kim 2020), POPO (Hou et al. 2020), and BRAC (Wu, Tucker, and Nachum 2019) use ensembles to do this by picking either the Q-function with the lowest value or they pick a weighted average of the current highest and lowest Q-value in the ensemble. This technique reduces the over-estimation propagation caused by the target network and the max operator in the Bellman equation. This works well in online RL (Haarnoja et al. 2018; Fujimoto, Hoof, and Meger 2018), but in offline RL, insufficient information and insufficient coverage in the data set also causes over-estimations which are not addressed by this technique (Rashidinejad et al. 2021; Jin, Yang, and Wang 2020). Methods like BEAR, POPO, and BRAC have to resort to policy-constrained techniques such as the maximum mean discrepancy, KL divergence, and Wasserstein Distance to counteract this secondary source of over-estimations (Kim 2020; Hou et al. 2020; Wu, Tucker, and Nachum 2019). In contrast, our method addresses the over-estimations caused by insufficient information by penalizing the Q-values based on the epistemic uncertainty.

Another interesting approach is Conservative Q-learning (Kumar et al. 2020) and its model-based version Conservative Offline Model-Based Policy Optimization (COMBO) (Yu et al. 2021). These methods regularize the Q-values by simultaneously minimizing all the Q-values and maximizing the Q-values in the data set, finding a lower bound on the Q-value. While it appears that this algorithm aims to avoid low-information regions similar to the algorithms in the uncertainty-aware pessimistic family, this is not the case because it uses data concentration as a proxy for information. In reality, it belongs to the policy-constrained pessimistic algorithmic family because it aims to stay close to the empirical policy, especially in very noisy or small data sets. This constraint is much looser than the previously mentioned policy constraints in practice, allowing for larger improvements. In contrast, our uncertainty-aware pessimistic algorithm is not constrained by any empirical policy, which theoretically allows for even larger improvements upon the empirical policy in a wider range of data sets (Rashidinejad et al. 2021; Jin, Yang, and Wang 2020).

Currently, Model-based Offline Policy Optimization (MOPO) (Yu et al. 2020) and Model-Based Offline Reinforcement Learning (MOReL) (Kidambi et al. 2020) are the most similar to our model-free algorithm. These model-based methods use an ensemble of models to estimate the uncertainty in the MDP. This uncertainty estimate is used to construct a pessimistic version of the MDP by subtracting the state uncertainty from the modeled rewards. The resulting pessimistic MDP is then used as input to a model-free method to learn a pessimistic policy. The major disadvantage of these methods is that they do not learn a pessimistic policy using an end-to-end based technique, as our method does. This is potentially an issue because when a Bellman update is performed on a function approximated Q-network, the value of a particular state is impacted by generalizations from other states, which potentially confounds the pessimistic penalties (Buckman, Gelada, and Bellemare

2020). We conclude that there is a gap in the offline deep RL literature for an end-to-end model-free uncertainty-aware pessimistic algorithm, which we aim to address using our method PEBL.

Uncertainty-Based Pessimism in Deep Reinforcement Learning

The uncertainty in reinforcement learning can be decomposed into two types of uncertainty, aleatoric and epistemic (Abdar et al. 2020; Kendall and Gal 2017). *Aleatoric* uncertainty arises from the stochasticity that is naturally present in the observations. Its key property is that it cannot be reduced by adding more data. A typical example of aleatoric uncertainty is a random or noisy reward function; adding more data from this function will *not* remove the noise in the observations. *Epistemic* uncertainty describes what the model does not know due to limitations in the observed data. This type of uncertainty has the key property that it *can* be reduced by adding more data. In this work, we are mainly concerned with epistemic uncertainty.

We seek to apply uncertainty-based pessimism to address two key challenges emergent in offline deep reinforcement learning. The first issue arises when a naïve algorithm, such as an online version of DDQN or SAC, is applied to an offline data set. In this scenario, a wide range of literature has observed that the Q-values can increase continuously until the algorithm diverges (Levine et al. 2020; Fujimoto et al. 2019; Kim 2020). This effect is most apparent in data sets with low state-action space coverage, such as small data sets and expertly generated data sets. The issue is caused by overoptimistic value estimations of the next state. These overestimations can be small initially, but they compound over time due to the Bellman equation, resulting in divergent behavior. One way to apply pessimism to counter this compounding effect is by penalizing the estimated Q-value in the Bellman backup of the target network by a factor of the uncertainty (estimated by the standard deviation) (Levine et al. 2020):

$$y_t = \overline{\mu}_Q(s_{t+1}, a; \theta') - \overline{\sigma}_Q(s_{t+1}, a; \theta') \tag{1}$$

In this equation, $\overline{\mu}_Q$ is the mean, $\overline{\sigma}_Q$ is the standard deviation of the Q-value approximations, and θ' is the weights of the target network.

Another issue that must be addressed is the trade-off between minimizing uncertainty and maximizing the Q-values in the learned policy. Uncertainty-aware offline RL algorithms do this by learning a policy that maximizes the uncertainty lower bound of the Q-values:

$$\pi_p = \underset{a}{\operatorname{argmax}}\{\overline{\mu}_Q(s_t, a; \theta) - \overline{\sigma}_Q(s_t, a; \theta)\} \tag{2}$$

In the remainder of this section, we will discuss how this estimate $\overline{\sigma}_Q$ can be obtained.

Estimating Uncertainty in Deep Reinforcement Learning

We seek a penalty that represents the epistemic uncertainty, not the aleatoric uncertainty, which means it should be inversely proportional to the density in the data set as well as our confidence. The deep learning literature proposes multiple ways to estimate the uncertainty in the predictions of a neural network (Osband et al. 2016; Osband, Aslanides, and Cassirer 2018; Gal and Ghahramani 2016; Blundell et al. 2015; Kendall and Gal 2017; Liu et al. 2020). One of the best-known uncertainty estimation techniques methods is Monte Carlo dropout (Gal and Ghahramani 2016). This method measures the uncertainty as the sample standard deviation over N Monte Carlo samples of slightly differing network configurations. Dropout has been shown to successfully capture uncertainty in several applications (Wang et al. 2019; Nair et al. 2020; Do et al. 2020). However, it mainly focuses on aleatoric uncertainty because its Bayesian posterior does not concentrate on the observed data and it cannot propagate its uncertainty through the Bellman fixed point (Osband, Aslanides, and Cassirer 2018). These properties make Monte Carlo dropout an unsuitable candidate for our problem.

Another well-known uncertainty estimation technique is the ensemble (Pearce, Leibfried, and Brintrup 2020; Osband, Aslanides, and Cassirer 2018; Lakshminarayanan, Pritzel, and Blundell 2017). An ensemble measures the uncertainty as the sample standard deviation between the prediction of its members. Ensembles are very good at estimating epistemic uncertainty as long as every member learns a different function in low data density areas. Another advantage of the ensemble is that its uncertainty measure is function-dependent and can propagate uncertainties through the Bellman fixed-point by definition (Osband, Aslanides, and Cassirer 2018).

Our method uses the multi-headed bootstrap ensemble with random priors to ensure that the ensemble members are as diverse as possible (Osband, Aslanides, and Cassirer 2018). This approach is similar to the traditional deep learning ensemble, but it improves upon it in two ways. First, it adds a different parallel prior to the prediction of each head (Figure 1). This prior is a randomly initialized but frozen network, meaning its weights will not change during the training process. Due to this strategy, each head has an input-dependent prior, while the mapping remains constant throughout the training process. Thus, in high data concentration areas, each head learns to ignore its prior and approaches a similar function, while in the low data concentration areas, each head is biased by its prior and there are more significant disagreements in the learned function between heads. This results in better epistemic uncertainty estimation for out-of-distribution data. The second improvement is the addition of the bootstrap. By training each ensemble member on a slightly different subset of the data, we ensure that it learns a different function where data is sparse, which results in improved epistemic uncertainty estimation at the edges of dense data concentration.

There is one notable disadvantage to our proposed uncertainty estimation technique: the number of weights and compute of this method increases linearly with the number of heads used. However, it is possible to share a feature extractor such as a convolutional neural network (CNN). In theory, this shared encoder can reduce the diversity in the ensemble, but it has been observed empirically that this negative ef-

fect is minimal, making this a valid computational trade-off (Osband et al. 2016; Osband, Aslanides, and Cassirer 2018; Sedlmeier et al. 2019). In practice, the parallel heads and their priors contain only one or two fully connected layers making the memory and performance requirements manageable.

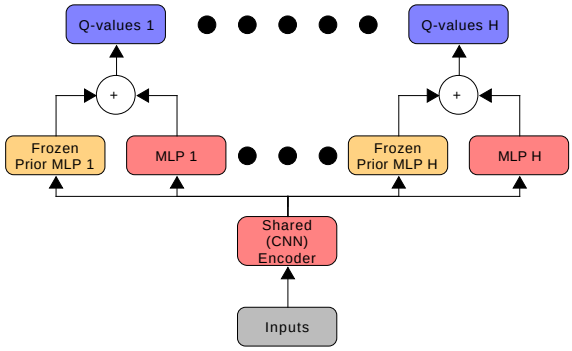


Figure 1: The multi-headed bootstrap ensemble with random priors network architecture. The architecture is similar to traditional ensembles but adds the output of a frozen prior network to each head's output to create a state depended prior.

Pessimistic Ensembles for Offline Deep Reinforcement Learning

We propose two new PEssimistic ensemBLe (PEBL) algorithms that use a multi-headed bootstrap with priors architecture to approximate the Q-function. The first algorithm is a pessimistic version of DDQN (Van Hasselt, Guez, and Silver 2016) aimed at problems with discrete action spaces. The use of two Q-networks helps avoid over-estimations in the function approximation, offering more stable performance. The second algorithm is a pessimistic version of SAC (Haarnoja et al. 2018) and works in both continuous and discrete action-space cases. Although SAC can also be applied to discrete action spaces, DDQN is often preferred in this setting due to its lower memory and computation costs.

Pessimistic Ensemble DDQN

Our pessimistic version of DDQN, which we call PEBL DDQN changes the TD-target of DDQN to:

$$y_t = r_t + (1 - d_t) * \gamma \overline{\mu}_Q(s_t, a^*; \theta') - \overline{\sigma}_Q(s_t, a^*; \theta') \quad (3)$$

$$a^* = \underset{a}{\text{argmax}}\{\overline{\mu}_Q(s_t, a; \theta)\overline{\sigma}_Q(s_t, a^*; \theta)\}$$

In this equation, $\overline{\mu}_Q(s_t, a; \theta)$ and $\overline{\sigma}_Q(s_t, a; \theta)$ are the sample mean and sample standard deviation over the different heads:

$$\overline{\mu}_Q(s_t, a; \theta) = \frac{1}{h}\sum_{i=0}^{h-1} Q_i(s_t, a; \theta_i) \quad (4)$$

$$\overline{\sigma}_Q(s_t, a; \theta) = \sqrt{\frac{\sum_{i=0}^{h-1}(Q_i(s_t, a; \theta_i) - \overline{\mu}_Q(s_t, a; \theta_i))^2}{h - 1}} \quad (5)$$

In this equation, Q_i is the Q-value prediction of the i-th ensemble head. This definition is different from the original definition from the bootstrapped DQN (Osband et al. 2016; Osband, Aslanides, and Cassirer 2018), which trained each head on its own target head. This is because we need to subtract an uncertainty penalty; if the penalty was subtracted from each head directly, an unstable feedback loop is created, resulting in potentially very large negative Q-values. Finally, we add the bootstrapping masks as described in the efficient implementation of bootstrapped DQN (Sedlmeier et al. 2019), giving the following definition for the TD-error for head i:

$$TD_i = m_{i,j} \cdot (Q_i(s_t, a; \theta_i) - y_t)$$

In this equation, $m_{i,j}$ is a boolean mask that has been sampled for each training sample j from a Bernoulli distribution with $p = 0.8$ as suggested by (Pearce, Leibfried, and Brintrup 2020). Note that the $m_{i,j}$ remains constant through the entire training process. The final policy selects the action with the highest pessimistic q-value.

$$\pi_t = \underset{a}{\text{argmax}}\{\overline{\mu}_Q(s_t, a; \theta) - C_\pi \cdot \overline{\sigma}_Q(s_t, a; \theta)\}$$

Pessimistic Ensemble SAC

Our pessimistic version of Soft Actor-Critic, called PEBL SAC, changes the target to:

$$y_t = r_t + (1 - d_t) \cdot \gamma \cdot (\overline{\mu}_Q(s, a; \theta_1, \theta_2) - \overline{\sigma}_Q(s, a; \theta_1, \theta_2)) - \alpha \log \pi_\phi(\tilde{a}_{t+1}|s_{t+1}) \quad (6)$$

where $\tilde{a}_{t+1}$ is sampled from the learned policy $\pi_\phi(.|s_{t+1})$. We define $\overline{\mu}_Q(s, a; \theta_1, \theta_2)$ and $\overline{\sigma}_Q(s, a; \theta_1, \theta_2)$ as:

$$\overline{\mu}_Q(s, a; \theta_1, \theta_2) = \frac{1}{h}\sum_{i=0}^{h-1} \min_{j=1,2} Q(s, a; \theta_{j,i}) \quad (7)$$

$$\overline{\sigma}_Q(s, a; \theta_1, \theta_2) = \sqrt{\frac{1}{h-1}\sum_{i=0}^{h-1}(\overline{\mu}_Q(s, a; \theta_1, \theta_2) - \min_{j=1,2} Q(s, a; \theta_{j,i}))^2} \quad (8)$$

Using these targets, we can calculate the TD-errors of the PEBL SAC algorithm in the same way we calculated the TD-errors for the PEBL DDQN algorithm with head i and mask m_i:

$$TD_{i,j} = m_i \cdot (Q_i(s_t, a; \theta_i) - y_t)$$

The original SAC algorithm calculates the policy loss using the formula:

$$\mathcal{L} = \frac{1}{|B|}\sum_{s \in B}(\alpha \log \pi_\phi(\tilde{a}_t, s_t) - \min_{i=1,2}\{Q(s_t, \tilde{a}_t; \theta_i)\})$$

Our pessimistic version of this algorithm changes the loss policy function to:

$$\mathcal{L} = \frac{1}{|B|} \sum_{s \in B} (\alpha \log \pi_\phi(\tilde{a}_t, s_t) - (\overline{\mu}_Q(s, a; \theta_1, \theta_2) - C_\pi \cdot \overline{\sigma}_Q(s, a; \theta_1, \theta_2)) \quad (9)$$

In this equation, we also introduce the uncertainty weight trade-off parameter C_π. This parameter controls the trade-off between minimizing uncertainty and maximizing the Q-values in the learned policy. This trade-off parameter is needed because the policy loss in SAC is similar to a white box adversarial attack on the Q-function due to the re-parameterization trick (Akhtar and Mian 2018). This adversarial formulation of the policy loss is not a problem in deep online RL because it forces the agent to learn about the flaws in its Q-function, which helps with exploration. However, in offline RL, this adversarial formulation is a problem because the agent can no longer collect counterexamples in the environment. Therefore, selecting the right parameter for C_π is crucial. Empirically, we found that it can be difficult to find the right value for C_π because it depends on many factors such as the size of the data set, the state-action space coverage, or the difficulty of modeling the true MDP. However, we can tune this parameter online using dual gradient descent, a technique that is increasingly common in deep reinforcement learning (Haarnoja et al. 2018; Kumar et al. 2020). We apply dual gradient descent and transform C_π into a Lagrangian multiplier by adding the constraint that the average uncertainty of the actions selected by the learned pessimistic policy π should be equal to the average uncertainty observed in the actions for the data set:

$$\frac{1}{n} \sum_{s_t, a_t}^{\mathcal{D}} \overline{\sigma}_Q(s_t, a_t; \theta) = \frac{1}{n} \sum_{s_t}^{\mathcal{D}} \overline{\sigma}_Q(s_t, a_p; \theta), a_p \in \pi_p \quad (10)$$

This constraint is possible because we are only interested in avoiding epistemic uncertainty, which is high in areas where the network cannot model the function well. Thus, we allow the selection of out-of-distribution actions as long as we avoid areas with above-average epistemic uncertainty. The constraint in Equation 10 captures this property, which results in a higher value of C_π if π_p chooses actions with above-average epistemic uncertainty. It results in a lower value for C_π if π_p chooses actions with below-average epistemic uncertainty.

Experimental Results

In this section, we discuss the results of several empirical evaluations of our PEBL methods. The main purpose of our experiments is to demonstrate that uncertainty-based pessimistic algorithms can be applied to offline deep RL problems and achieve high performance and that they can do well for a wide range of provided data distributions. In all experiments, the trade-off parameter C_π in the SAC version of our algorithm is learned using Lagrangian dual gradient descent (Equation 10). For an open-source implementation, please refer to the code in the accompanying GitHub repository: github.com/j0rd1smit/PEBL.

MinAtar

We first aimed to reproduce the results presented by Buckman, Gelada, and Bellemare in their paper (2020). In continuous experiments, they compared only three algorithmic families, excluding the uncertainty-aware pessimistic and leaving it as an open research question. We use their same experimental set-up and additionally assess the performance of our uncertainty-aware pessimistic algorithm PEBL.

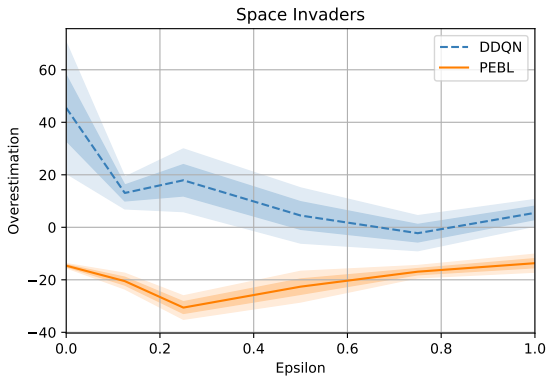
The two environments used in this experiment are from MinAtar, which contains several miniaturized versions of Atari 2600 games (Young and Tian 2019). As is common in offline deep RL experiments, we first train an online agent to obtain an optimal policy for each of the environments (Buckman, Gelada, and Bellemare 2020; Fujimoto et al. 2019; Fu et al. 2020; Agarwal, Schuurmans, and Norouzi 2020). We then use this policy to generate several data sets with different expert/random ratios by modifying epsilon in an epsilon-greedy strategy. The generated data sets used in this experiment contain 50000 transitions. We chose this slightly smaller data set size because it highlights the advantages of the uncertainty-based method. Using this experimental setup, we evaluate our method in two ways. Firstly, we validate that our method does not overestimate the expected discounted return. We do this by comparing the return our method obtains in the real environment with its prediction for the discounted return at the start of the episode. Secondly, we evaluate how well our method adheres to the theoretically predicted property of uncertainty-aware algorithms by measuring the final performance of the learned policy for varying values of epsilon.

Baselines We compare PEBL to methods representing the aforementioned algorithmic families. The first is the behavior clone, referred to as BC. The policy-based pessimistic family is represented by Batch-Constrained Q-Learning (BCQ) (Fujimoto, Meger, and Precup 2019). The third category is the naïve algorithmic family, represented by DDQN (Van Hasselt, Guez, and Silver 2016).

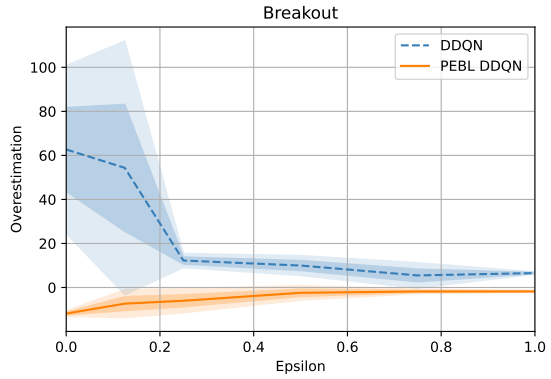
Results The outcomes of our MinAtar experiments are pictured in Figures 2 and 3. The results of Figure 2 show that the value function learned by our PEBL DDQN method does not overestimate the expected discounted return for any epsilon value in both the Space Invaders and Breakout environment. In contrast, the baseline DDQN method overestimates the expected discounted return for every epsilon value, and this overestimation gets worse when the amount of information in the data decreases.

In Figure 3, we see that PEBL DDQN performs best for a wider range of epsilon-greedy policy data sets than the other algorithmic families, in line with the theoretical results we aimed to replicate (Buckman, Gelada, and Bellemare 2020). However, our method does not follow the theoretically predicted behavior for uncertainty-aware pessimistic algorithms in the expert data regime. We expected to perform as well as or similarly to the behavior cloning method but fall notably short. This suggests that our epistemic uncertainty estimation techniques are not expressive enough for highly biased data. This result is interesting because Figure

2 shows that our method does not overestimate the expected discounted return in these highly biased expert data sets.



(a) Space invaders



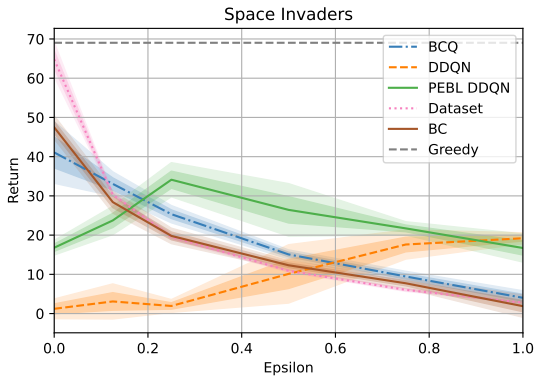
(b) Breakout

Figure 2: The over-estimations in our MinAtar experiment. We define the overestimation as the obtained return minus the predicted Q-value in the initial state. Our PEBL DDQN method does not over-estimate the expected discounted return for any ϵ in both the Space Invaders and Breakout environment. In contrast, the naive DDQN method overestimates for every ϵ.

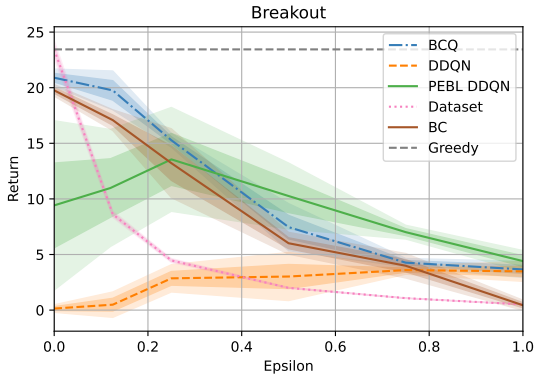
Maze-2D: Uncertainty Visualization

We applied PEBL SAC to the Maze-2D environment (Figure 4a) from the D4RL benchmark (Fu et al. 2020). A motivation for using this 2-dimensional domain is that we can visualize the influence of pessimism on what our algorithm learns. In this task, the agent is shown a data set with unrelated paths through the maze containing only x and y positions and velocities. The agent has to stitch these transitions together to find a path to a goal location in the maze. The challenge of this environment is that the agent never observes the optimal path and has to learn the maze's layout through the data set.

Baselines We compare our method to the results provided in the benchmark paper for SAC, Behavior Cloning (BC), and Continuous Batch-Constrained Q-Learning (CBCQ) (Fujimoto, Hoof, and Meger 2018; Fu et al. 2020).



(a) Space invaders



(b) Breakout

Figure 3: The performance of different representatives of each offline RL algorithmic families compared to our algorithm. Measurements are obtained by taking the average performance of the final policy over 100 episodes averaged over 3 random seeds. Each random seed has its own data set containing 50000 transitions sampled with an ϵ-greedy policy, where the greedy policy is an expert DDQN agent.

Results The performance of the algorithms in the Maze-2D experiment are displayed in Table 1. PEBL SAC is able to deviate further from the observed behavior policy than the other baselines. This allows it to solve all mazes, including the large maze, which has not yet been solved by any existing policy-based pessimistic method (Fu et al. 2020). Figure 4b visualizes the value of the uncertainty-based penalty and the penalized Q-values. We see that the result of our pessimistic penalty is that the agent assigns low Q-values to areas which it is uncertain about, such as positions occupied by walls.

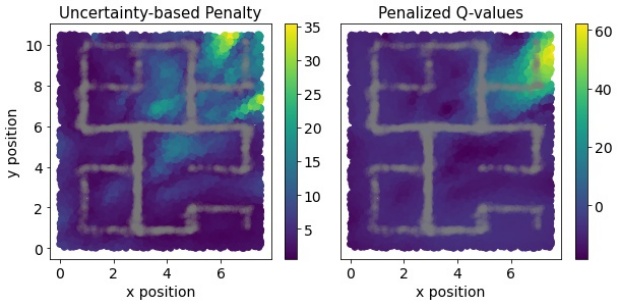
D4RL: MuJoCo gym

In this group of experiments, we compare PEBL SAC to some of the more difficult continuous-action tasks in the D4RL benchmark.

Maze type	SAC	BC	CBCQ	PEBL SAC
U	88.2	3.8	12.8	**151.1**
Medium	26.1	30.3	8.3	**146.6**
Large	-1.9	5.0	6.2	**129.5**

Table 1: Results for D4RL's Maze-2D benchmark. Each score is the average normalized reward over 100 runs at the last iteration of training.

(a) Maze-2D environment



(b) The uncertainty-based penalty and the penalized Q-values per location on the map

Figure 4: A visualization of the learned Q-function and its uncertainty for the large Maze-2D environment (Fu et al. 2020). Note that the agent only observes (x, y, v_x, v_y). Figure b) shows the uncertainty-based penalty $C_\pi \cdot \overline{\sigma}_Q(s, a; \phi_1, \phi_2)$ where C_π was learned and converged to $C_\pi = 12.53$ and the penalized Q-values, where the grey areas are data points provided in the data set.

Baselines Similarly to the first experiment, we compare our methods to representatives of each algorithmic family. The methods are SAC (Haarnoja et al. 2018) and Continuous Batch-Constrained Q-Learning (CBCQ) (Fujimoto, Meger, and Precup 2019) for the naïve and the policy-based pessimistic algorithmic family, respectively. We also compare our method to Conservative Q-learning (CQL) (Kumar et al. 2020) and Conservative Offline Model-Based Policy Optimization (COMBO) (Yu et al. 2021). Both methods are considered state-of-the-art offline RL methods and members of the policy-based pessimistic algorithmic family (Buckman, Gelada, and Bellemare 2020). Compared to other policy-based pessimistic algorithms, these methods have a significantly looser constraint, allowing for larger improvements upon the data collection policies.

Results The results are shown in Table 2, where the performance of SAC, BC, and CBCQ is given as reported in (Fujimoto, Meger, and Precup 2019), and for CQL and COMBO are the number reported in (Kumar et al. 2020) and (Yu et al. 2021), respectively. On the data sets generated using non-expert data policies, marked *Random* and *Medium*, PEBL SAC performs on par or exceeds the best prior methods. Interestingly, it even out-performs a model-based method, while PEBL SAC is a model-free method itself. Similar to the PEBL DDQN algorithm, we are unable to match the performance of behavior cloning in the expert data regime.

Data set type	SAC	BC	CBCQ	CQL	COMBO	PEBL SAC
Random	30.5	2.1	2.2	35.4	**38.8**	35.7
Medium-Replay	1.9	38.4	38.2	44.4	54.2	**57.5**
Medium	-4.3	36.1	40.7	46.2	55.1	**61.5**
Expert	-1.9	**107.0**	-	104.0	-	3.6

Table 2: Results for the D4RL benchmark. Each score is the average normalized reward over 100 runs at the last iteration of training as described in the D4RL benchmark.

Conclusion and Future Work

We introduced the multi-headed bootstrap with randomized priors approach to measuring epistemic uncertainty. Using this ensemble-based method, we penalized the value function to produce two new pessimistic offline RL algorithms, called PEBL DDQN and PEBL SAC. These algorithms are a step towards robust uncertainty-aware pessimistic offline RL algorithms in the deep learning setting. We have shown that our methods are able to perform well on a wide range of data set distributions compared to algorithms from the naïve and policy-based pessimistic families. However, in experiments, we also showed that our methods do not perform well on expert data sets, even though the theory predicts they would. We expect that our epistemic uncertainty estimation techniques are not expressive enough for highly biased data. A more suitable uncertainty measure may address this issue. Thus we hope to continue to investigate appropriate measures of uncertainty for these problems. Some promising methods for future work are Bayesian neural networks (Blundell et al. 2015) and deep Gaussian processes (Liu et al. 2020; van Amersfoort et al. 2021). Both methods are more accurate in their estimation of epistemic uncertainty but are more difficult to optimize in practice. We would also like to examine whether the epistemic uncertainty estimation of our method can be improved using self-supervised representation learning (Stooke et al. 2020; Laskin, Srinivas, and Abbeel 2020; Laskin et al. 2020). This technique has been shown to improve existing offline RL methods in various ways (Sinha and Garg 2021). It may prove beneficial for our method as the improved representations might make out-of-distribution detection easier. In general, we hope that our work inspires further investigation into the development of uncertainty-aware pessimistic algorithms that adhere to the theoretical support of their abilities.

Acknowledgments and Disclosure of Funding

This work received support as part of the research programme Physical Sciences TOP-2 project number 612.001.602, which is financed by the Dutch Research Council (NWO).

References

Abdar, M.; Pourpanah, F.; Hussain, S.; Rezazadegan, D.; Liu, L.; Ghavamzadeh, M.; Fieguth, P.; Khosravi, A.; Acharya, U. R.; Makarenkov, V.; et al. 2020. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *arXiv preprint:2011.06225* .

Agarwal, R.; Schuurmans, D.; and Norouzi, M. 2020. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 104–114. PMLR.

Akhtar, N.; and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* 6: 14410–14430.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 1613–1622. PMLR.

Buckman, J.; Gelada, C.; and Bellemare, M. G. 2020. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint:2009.06799* .

Cabi, S.; Gómez, S.; Novikov, A.; Konyushova, K.; Reed, S.; Jeong, R.; Zołna, K.; Aytar, Y.; Budden, D.; Vecerik, M.; Sushkov, O.; Barker, D.; Scholz, J.; Denil, M.; Freitas, N.; and Wang, Z. 2020. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems*. doi:10.15607/RSS.2020.XVI.076.

Do, H. P.; Guo, Y.; Yoon, A. J.; and Nayak, K. S. 2020. Accuracy, uncertainty, and adaptability of automatic myocardial ASL segmentation using deep CNN. *Magnetic resonance in medicine* 83(5): 1863–1874.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *CoRR* abs/2004.07219.

Fujimoto, S.; Conti, E.; Ghavamzadeh, M.; and Pineau, J. 2019. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint:1910.01708* .

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2052–2062. PMLR.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *Artificial Intelligence* 122(1-2): 71–109.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint:1812.05905* .

Hou, Q. H.; et al. 2020. POPO: Pessimistic Offline Policy Optimization. *arXiv preprint:2012.13682* .

Jin, Y.; Yang, Z.; and Wang, Z. 2020. Is Pessimism Provably Efficient for Offline RL? *arXiv preprint:2012.15085* .

Kendall, A.; and Gal, Y. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MOReL: Model-Based Offline Reinforcement Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21810–21823. Curran Associates, Inc.

Kim, S. 2020. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. Submitted to NeurIPS 2019 Reproducibility Challenge.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *arXiv preprint:2006.04779* .

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; and Srinivas, A. 2020. Reinforcement Learning with Augmented Data. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 19884–19895. Curran Associates, Inc.

Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 5639–5650. PMLR.

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

Liu, J. Z.; Lin, Z.; Padhy, S.; Tran, D.; Bedrax-Weiss, T.; and Lakshminarayanan, B. 2020. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint:2006.10108* .

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint:1312.5602* .

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Nair, T.; Precup, D.; Arnold, D. L.; and Arbel, T. 2020. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. *Medical image analysis* 59: 101557.

Nilim, A.; and El Ghaoui, L. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53(5): 780–798.

Osband, I.; Aslanides, J.; and Cassirer, A. 2018. Randomized prior functions for deep reinforcement learning. *arXiv preprint:1806.03335* .

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep Exploration via Bootstrapped DQN. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Pearce, T.; Leibfried, F.; and Brintrup, A. 2020. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, 234–244. PMLR.

Pietquin, O.; Geist, M.; Chandramohan, S.; and Frezza-Buet, H. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)* 7(3): 1–21.

Rashidinejad, P.; Zhu, B.; Ma, C.; Jiao, J.; and Russell, S. 2021. Bridging Offline Reinforcement Learning and Imitation Learning: A Tale of Pessimism. *arXiv preprint:2103.12021* .

Sedlmeier, A.; Gabor, T.; Phan, T.; Belzner, L.; and Linnhoff-Popien, C. 2019. Uncertainty-based out-of-distribution classification in deep reinforcement learning. *arXiv preprint:2001.00496* .

Sinha, S.; and Garg, A. 2021. S4RL: Surprisingly Simple Self-Supervision for Offline Reinforcement Learning. *arXiv preprint:2103.06326* .

Stooke, A.; Lee, K.; Abbeel, P.; and Laskin, M. 2020. Decoupling representation learning from reinforcement learning. *arXiv preprint:2009.08319* .

Strehl, A.; Langford, J.; Li, L.; and Kakade, S. M. 2010. Learning from Logged Implicit Exploration Data. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

van Amersfoort, J.; Smith, L.; Jesson, A.; Key, O.; and Gal, Y. 2021. Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression. *arXiv preprint:2102.11409* .

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Wang, G.; Li, W.; Aertsen, M.; Deprest, J.; Ourselin, S.; and Vercauteren, T. 2019. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing* 338: 34–45.

Wang, L.; Zhang, W.; He, X.; and Zha, H. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2447–2456.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4): 279–292.

Wiesemann, W.; Kuhn, D.; and Rustem, B. 2013. Robust Markov decision processes. *Mathematics of Operations Research* 38(1): 153–183.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint:1911.11361* .

Young, K.; and Tian, T. 2019. MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. *arXiv preprint:1903.03176* .

Yu, T.; Kumar, A.; Rafailov, R.; Rajeswaran, A.; Levine, S.; and Finn, C. 2021. Combo: Conservative offline model-based policy optimization. *arXiv preprint:2102.08363* .

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 14129–14142. Curran Associates, Inc.

Zhou, K.; Doyle, J. C.; Glover, K.; et al. 1996. *Robust and optimal control*, volume 40. Prentice hall New Jersey.