JAISCR, 2023, Vol. 13, No. 4, pp. 273 – 287
10.2478/jaiscr-2023-0019

DISCRETE UNCERTAINTY QUANTIFICATION FOR OFFLINE REINFORCEMENT LEARNING

José Luis Pérez[1], Javier Corrochano[1], Javier García[2], Rubén Majadas[1], Cristina Ibañez-Llano[3], Sergio Pérez[3], Fernando Fernández[1,*]

[1]*Computer Science Department, Universidad Carlos III de Madrid, Spain*

[2]*Electronics and Computing Department, Universidad de Santiago de Compostela, Spain*

[3]*Repsol Technology Lab, Repsol, Spain*

[*]*E-mail: ffernand@inf.uc3m.es*

Submitted: 2th June 2023; Accepted: 7th October 2023

Abstract

In many Reinforcement Learning (RL) tasks, the classical online interaction of the learning agent with the environment is impractical, either because such interaction is expensive or dangerous. In these cases, previous gathered data can be used, arising what is typically called Offline RL. However, this type of learning faces a large number of challenges, mostly derived from the fact that exploration/exploitation trade-off is overshadowed. In addition, the historical data is usually biased by the way it was obtained, typically, a sub-optimal controller, producing a distributional shift from historical data and the one required to learn the optimal policy. In this paper, we present a novel approach to deal with the uncertainty risen by the absence or sparse presence of some state-action pairs in the learning data. Our approach is based on shaping the reward perceived from the environment to ensure the task is solved. We present the approach and show that combining it with classic online RL methods make them perform as good as state of the art Offline RL algorithms such as CQL and BCQ. Finally, we show that using our method on top of established offline learning algorithms can improve them.

Keywords: Off-line Reinforcement Learning, uncertainty quantification, Machine Learning

1 Introduction

Classical Reinforcement Learning (RL) tasks are solved by an online interaction between the learning agent and the environment. Sometimes, this is unfeasible, either because such interaction is very expensive or because it may produce catastrophic effects in the agent or its environment. In addition, even when an online interaction is feasible, we might prefer to use previously collected data, for example, to obtain a sub-optimal policy that can be used in a later fine-tuning process, minimizing the amount of new data required. This way of learning from a batch of experiences without exploration has been referred to as Batch RL, Offline RL, or data-driven RL [1, 2, 3].

Some of the most common methods within RL can learn from off-policy data, following an experience replay approach. However, methods that apply experience replay do not make it fully effective from offline data without adding some online interaction due to several factors. The main problem is that the classical exploration-exploitation trade-off that makes efficient and effective to most RL algorithms is, in Offline RL, broken due to some new challenges. A fundamental one is related to the distributional shift, also called out-of-distribution states and actions [4, 5, 1]. Distributional shift is typically defined as the difference between the distribution of the data on which our function approximator (policy, value function or model) has been trained and the distribution in which it will be evaluated. This is due both to the change in the states visited by the learned policy, and to the act of maximising the expected return.

In Offline RL, as well as in the scope of exploration and exploitation trade-off, distributional shift appears by the difference between the available data for learning the policy and the data required to learn the optimal policy. Basically, if some experiences are required to learn the optimal policy, but such experiences are never visited, there is no way to learn the optimal policy. Sometimes, even some state-actions pairs are visited, they are not done in an amount enough so the model acquired from it is accurate. The importance of this point grows with the stochasticity of the environment and its reward and/or state transition function. The more lack of data to learn the model, the more uncertainty about the model correctness.

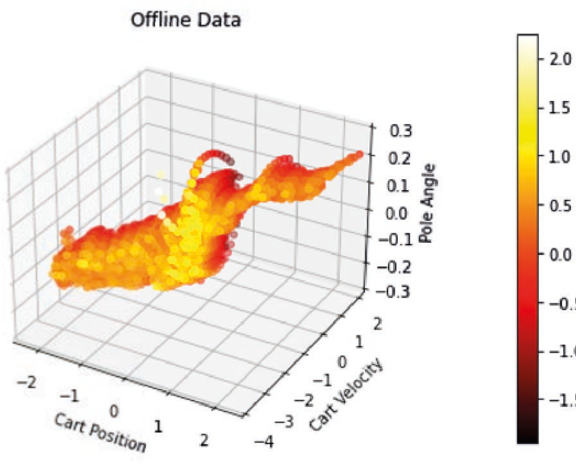
In this paper, we present a simple but effective approach to quantify the uncertainty that a dataset will generate based on how frequent the visited states are. This information is used to conservatively reshape the reward signal of the environment, therefore propagating it to the value function estimation. Therefore, state-actions pair that have been visited in low proportion to other states, should contribute also less to the computation of the optimal policy, or even be avoided since its outcome is quite uncertain.

As an example, Figure 1 shows two different data distributions of the Cartpole domain[1]. The first distribution, Figure 1a, is an exploration performed during training with offline data. The second one, Figure 1b, is a random exploration performed in the environment of the Cartpole domain. As seen in these images, the distribution corresponding to the exploration in the environment is wider than the distribution of the offline dataset, in other words, there are regions that are less known or even completely missing by the offline dataset. If an agent learns a policy using the offline dataset, the agent will behave blindly in all such unknown areas.

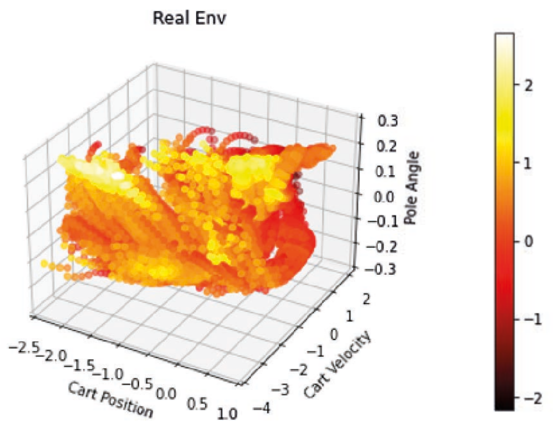
The main idea proposed in this paper is to weight the original reward with a measure of the uncertainty of that region of the space. The metric is based on the clustering of the visited state space. We use a well known bias of the *k*-means algorithm, which locates centroids or prototypes in all and only the known instance space, but in an unbalance way: it may locate only a few instances in one Voronoi region, and many in others, focusing only in the average distortion metric. Therefore, our measure is based on the number of instances that are located in the visited region or cluster, so the higher the number of instances, the better known is be transition function for such region. In case a region is well known the original reward is used as it is received from the environment. In the opposite case, if the number of instances is low, we understand that the region is less-known and the original reward is reduced proportionally. This measure also can be understood as an exploration quality indicator. The number of instances in less-known regions will be low, and this indicates that we are dealing with a poor exploration or that we would need additional exploration to ensure a correct learning of the value function.

The paper is organized as follows. Section 2 reviews the related background, with a brief introduction to Offline RL, distributional shift and uncertainty quantification. Section 3 present a brief review of previous and related works on Offline RL. Then, in Section 4, the new approach to deal with distributional shift of the dataset through the reshaping of the reward function is presented. After this, an extensive evaluation is reported in Section 5. The experiments shows the results obtained by introducing this reward reshaping with different al-

[1]This four-dimensional environment is represented by the *x* position of the cart, its velocity, the angle of the pole with respect to the cart and its pole angular velocity.



(a) States visited in the data available for Offline RL



(b) States visited following a ransom exploration strategy

Figure 1. Exploration comparison. The colour of the points represents the value of the fourth dimension of the state (pole angular velocity). Offline data is clearly biased to some areas that could prevent the correct learning of the optimal policy.

gorithms in three different domains from the D4RL [6] benchmark datasets. Last, Section 6 describes the main conclusions and future research.

2 Background

In this section, some concepts of RL, its Offline version and uncertainty quantification are reviewed.

2.1 Reinforcement Learning

The RL problem is formalized by a Markov Decision Process (MDP), which is represented by $\mathcal{M} = \{ S, A, R, P, \rho_0, \gamma \}$, where S is the state space, A is the action space, $R : A \times S \to \mathbb{R}$ is the reward function, $P : S \times A \times S' \to [0,1]$ is the transition function which returns the probability to reach state s' from state s when action a is performed, ρ_0 is the initial state distribution, and γ is the discount factor. The goal is to obtain an optimal policy π, which maximizes expected return according to the next formula, which estimates the discounted reward to be received:

$$J(\pi) = \sum_{k=0}^{K} \gamma^k r_k \tag{1}$$

Classic reinforcement learning is well-known for its online nature, agent's policy is learned via trial-and-error interaction with the environment. Much progress has been made in this area, in particular in the field known as Deep RL. Solutions such as Deep Q Networks (DQN) [7] or Actor Critic methods [8] have become a reference.

Exploitation vs. exploration problem is essential in classic reinforcement learning, where finding a trade-off between following greedy steps given by current policy (exploit) and including a random behaviour that allows reaching unknown regions of the state space (explore) is indispensable to achieve optimal policies. Traditionally, this issue has been addressed through different strategies, like ε-greedy, that allow the above-mentioned balance to be maintained.

Some techniques have incorporated batch fashion approaches, which is known as experience replay. Instead of learning iteratively and update the Q-values every time step, the agent holds a memory or buffer of the last N experience tuples to be taken into account in the future. As a summary, the basic idea behind experience replay is to store past experiences and then use these experiences to update the Q-values, rather than using just the single most recent experience. However, methods that apply experience replay do not make it fully effective from offline data without adding some online interaction due to the distributional shift, as will be discussed below.

2.2 Offline Reinforcement Learning

In the Offline RL setting, any interaction with the environment is unfeasible. Instead, a fixed dataset of transitions is available which is formalised as $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{N}$. Experience tuples are assumed to be collected by a logging (behavioral) policy π_β which might not be optimal.

The Offline approach seeks improvements beyond imitation learning naive techniques. Behavioural Cloning (BC) replicates logging policy π_β by learning from batch experience in a supervised manner. Offline RL tries to achieve a good generalization and improve knowledge transfer without being constrained by the data.

Experience tuples dataset may not contain some high reward regions, which are essential for optimal learning. One of the main problems in Offline Reinforcement Learning relies on the constraint of learning through data, without any interaction with the environment. Hence, learning is restricted to the exploitation of available data. The difference between logging policy π_β and optimal policy π^* which could visit unknown or out-of-distribution states is one of the open problems in this research field called distributional shift, as will be discussed below.

This paradigm would allow us to apply RL to domains where it is currently infeasible or impractical to collect data online, such as healthcare (e.g.,medical diagnosis), robotics (e.g., robotics manipulation), inventory management, and autonomous driving [3, 1]. Existing works and projects which solve distributional shift and implements reinforcement learning in an offline manner will be reviewed in Section 3.

2.3 Distributional Shift

A Machine Learning model employs training data to learn the underlying distribution of that dataset with the goal of getting accurate predictions for unseen data. When the model is able to generate accurate predictions for unseen data and performs well, it is said that model generalizes. These concepts are addressed in the supervised learning scope. However, the assumption that unseen data comes from the same distribution as training data is erroneous, the model might not generalize well. Distributional or Data Distribution Shift is the name of that common Machine Learning issue [9].

In the Offline Reinforcement Learning approach, a function approximator (e.g. policy, value function, or model) might be trained under one distribution and evaluated on a different distribution, due both to the change in visited states for the new policy and, more subtly, by the act of maximizing the expected return [1].

Distributional shift issues can be addressed in several ways and can be classified into three groups. The first one, policy constraint, mitigates distributional shift by constraining the learned policy to be close to the behaviour policy [1]. Uncertainty based solutions attempt to estimate the epistemic uncertainty of Q values, and then use this to detect distributional shift [1]. Lastly, regularization methods are used when we want to impose behaviors on our learned policy that do not depend on behaviour policy. Regularization is a powerful tool that allows us to tune our learned function by adding a penalty term. [3].

2.4 Uncertainty Quantification

Our approach is based on estimating the uncertainty of the state space in an unsupervised manner, thus some concepts about uncertainty quantification are reviewed in the following lines.

Sources of uncertainty arise when the test and training data are mismatched, while data uncertainty occurs because of class overlap or due to the presence of noise in the data, however, estimating knowledge uncertainty is significantly more difficult than estimating data uncertainty [10]. Potential causes of uncertainty are noise in observations or an incomplete coverage of the domain.

There are two main types of uncertainty. The irreducible uncertainty in data that gives rise to uncertainty in predictions is aleatoric uncertainty (also known as data uncertainty). This type of uncertainty is not a property of the model, but rather is an inherent property of the data distribution, and hence, it is irreducible [10]. The second one is epistemic uncertainty, which arises when the model has incomplete knowledge. The main reason of that type of uncertainty is the lack and poor quality of training data. Offline RL methods based on uncertainty quantification try to measure it.

Numerous techniques are being developed in uncertainty quantification research field. Some of them include bayesian approaches [10], model ensemble approach (common in model based Offline RL)[10, 11, 5] and unsupervised learning methods [12] such as the one presented in this article.

3 Related Work on Offline RL

Offline RL is an emerging field that has gained momentum over the past few years. Most of its works try to deal with the problem of distributional shift using different techniques. In the literature, there exist model-free and model-based methods as in online RL. We give a brief overview of them alongside their online counterpart, discussing their most prominent algorithms and their relation with policy constraint and uncertainty-based methods.

3.1 Online RL

Although online algorithms are not specifically designed for offline applications, they have been used in the past. Off-policy online reinforcement learning algorithms allow learning from fixed collections of data. However, those techniques failed when training data is uncorrelated to the distribution under the current policy [13]. This problem was reviewed in the previous section.

A widely adopted solution in continuous control tasks in an online manner is Soft Actor-Critic (SAC) [14]. SAC is a model-free and off-policy RL model that maximizes both the expected reward and entropy. Our approach, DUQ, provides a simple and effective way of quantifying uncertainty for adapting SAC and other online off-policy algorithms to offline scenarios, as will be shown below.

3.2 Model-free Offline RL

Fujimoto et al. [13] presented the first continuous control Deep RL algorithm, Batch-Constrained Deep Q-learning (BCQ), which can learn effectively from a fixed batch of data. They introduced a novel class of off-policy algorithms, which restricts the actions space in order to force the agent to behave close to the policy with respect to a subset of the given data. BEAR [15] and BRAC [16] follow the ideas introduced by BCQ, arguing policy constraint methods. Kumar et al. BCQ has been used in practical cases, as energy management optimization for connected hybrid electric vehicle[17]. The Conservative Q-learning (CQL) algorithm is capable of learning from a fixed dataset and without further interaction[18]. It aims to address the limitations caused by the distributional shift by learning a conservative Q-function such that the expected value of a policy under this Q-function lower-bounds its true value. This paper provides an easy-to-use mechanism to further reduce out-of-distribution states and actions problems in these solutions with little modification. Related with uncertainty quantification, Agarwal et. al. [19] presented REM, an ensemble of K Q-functions trained by disjoints subsets of the dataset [3].

Model-free algorithms are computationally more efficient than model-based solutions because they do not need to generate a model and sample long rollouts over it. However, they offer lower returns by exploiting data, as noted by the Siemens Technology Team in their research [20].

3.3 Model-based Offline RL

Model-Based Policy Optimization (MBPO) [21] is a model-based RL algorithm that, if it is properly tuned, it can yield better results than model-free approaches in the offline setting. This method utilizes a predictive model of the transition distribution from the dataset. It updates the policy using data sampled both from the dataset and model. Other state-of-the-art works propose conservative model-based RL algorithms, such as Model-Based Offline RL (MOReL) [11], Model-based Offline Policy Optimization (MOPO) [5] and Conservative Offline Model-Based Policy Optimization (COMBO) [22].

They use conservative value estimates by modifying the MDP model learned from data to induce conservative behaviours. Their main idea is to give the policy a penalty for visiting states in which the trained model is highly unlikely to perform well. Both MOReL and MOPO learn an ensemble of N dynamics models with each model trained independently via maximum likelihood. With this ensemble, MOReL and MOPO quantify the uncertainty by defining an error measure, which changes for each selected state-action pair. On the one hand, MOPO utilizes this measure as a soft penalty in the reward function. On the other hand, MOReL constructs the MDP with terminal states based on a threshold of this measure. Finally, COMBO extends CQL [18] into the model-based setting. COMBO overcomes uncertainty quantification step of the previous methods which can be unreliable. It is similar to MOPO, although it penalizes the Q-values directly instead of through the reinforcement function. While model-based approaches offer great performance, they are usually harder to fit due to their added complexity. The simplicity of our solution helps to decrease distributional shift using a easy-to-use technique with few parameters.

3.4 Datasets and metrics

Datasets used in Offline RL are usually collected by generating rollouts of a final policy π_β (optimal or not) or extracting data from the replay buffer used by online algorithms in their training phase. Work with realistic problems and data also is open to managing handcrafted policies and experience tuples which can break the Markovian property. It is therefore ideal to be able to work with real-world data to be aware of these kinds of challenges.

The first aspect to consider when choosing a dataset is the nature of the data: discrete or continuous. Dealing with continuous data is difficult and requires complex solutions that take advantage of the computational power of neural networks. All state-of-the-art solutions developed since the release of BCQ [13] work on this line.

Another aspect to consider is the stochastic dynamics in the environment because of the connection to real world problems (e.g., economics, healthcare, education, etc.). However common suites and benchmarks have a lack of this kind of data (despite the Atari domains, which are well-known stochastic simulators [23]).

Data distribution and action-state space coverage have play an important role in dataset configuration and how this can affect the final outcome. Batch data collected by expert policies will be characterized by containing full rollouts leading to good solutions but the space may not be well covered. On the other side, random or suboptimal logging policies provides more coverage but it does not have to be complete. Design datasets with risky biases is important due to if a dangerous region is covered it will evaluated and the it will be known by the agent.

D4RL [6] and RL Unplugged [24] are two of the most extended benchmark in literature. Both provide fixed datasets of different tasks with different logging configurations. In D4RL we find MuJoCo datasets, CARLA simulator data and maze environments among others. RL Unplugged, on its side, contains Atari games data, locomotion task and real world applications.

4 Discrete Uncertainty Quantification for Offline RL (DUQ)

DUQ is a simple approach to quantify uncertainty using the discretization of the state space. This information is received by the agent through the reward and used in the learning process to avoid less-known regions. In the following section, we motivate this concept and include its formal definition by firstly introduce some definitions and concepts from vector quantization literature, as Voronoi regions.

4.1 DUQ Metric

A vector quantizier Q of dimension k and size N is a mapping from a vector in k-dimensional Euclidean space, $\mathbb{R}^k$, into a finite set C containing N output or reproduction points, called code vectors or centroids [25]. Thus,

$$Q : \mathbb{R}^k \rightarrow C \tag{2}$$

where $C = \{c_1, c_2, ..., c_n\}$ and $c_i \in \mathcal{R}^k$.

Associated with every centroid c_i is a partition or cell, R_i [25], defined by:

$$R_i = \{x \in \mathbb{R}^k : Q(x) = c_i\} \tag{3}$$

i.e., R_i is composed of all points x belonging to cluster c_i. In turn, each R_i consists of all points x which have less distorsion when encoded with centroid c_i than with any other [25]:

$$R_i = \{x : dist(x, c_i) \leq dist(x, c_j), i \neq j\} \tag{4}$$

where $dist(x,y)$ is computed as $dist(x,y) = ||x - y||^2$. The Voronoi regions R_i can be obtained with clustering techniques such as k-means or mini batch k-means. Inside these algorithms, the well-known Lloyd's Iteration is computed [26]. Given the set of centroids C and their Voronoi regions R (both could be random or the result of a previous iteration), centroids are iteratively recalculated by:

$$centroid(R_i) = \frac{1}{|R_i|} \sum_{x \in \mathbb{R}} x \tag{5}$$

The main objective is to minimize the mean distortion for all centroids of set C.

We now have all the ingredients for the definition for the proposed DUQ metric. DUQ metric can be defined as Equation 6 shows:

$$DUQ(x) = |R_i|, x \in R_i \tag{6}$$

i.e., the DUQ of a state x is the number of instances $|R_i|$ of its corresponding Voronoi region R_i. To use the metric for training requires some additional transformations which are described next.

4.2 Apply DUQ to training

The above-formalized metric is applied to the training process through the reward function. A reward reshape is performed that takes into account the quantified uncertainty. First, DUQ metric is calculated and yields a result like the one shown in Figure 2. It shows the number of instances of each Voronoi region, for a specific dataset.

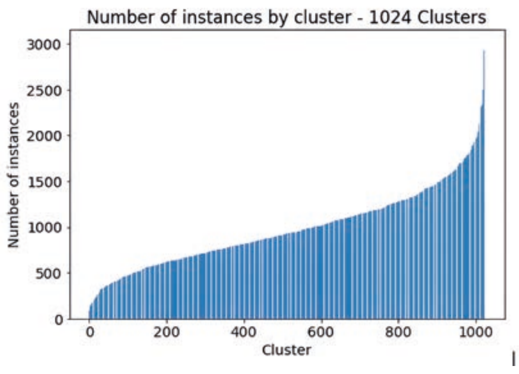


Figure 2. DUQ Metric without reshape.

This metric is not homogeneous: its values depend on the number of available data points and the number of clusters chosen. Specifically, in Figure 2, the number of clusters chosen is 1024 and the metric is approximately in the range [0-3000].

In order to obtain an homogeneous result, we add a transformation τ_p to weight and reshape the DUQ metric. This allows us to experiment with different shapes of the distribution $W_p^{DUQ} : \tau_p \cdot DUQ$. The transformation τ_p consists on a sigmoidal function in which the parameter p defines the input range of the function; $\tau_p(d_i) = \frac{1}{1+e^{-d_i}(p)}$. Some examples of this functions are shown later in Figure 3.

The last step is to use the reshaped DUQ metric in the reward function of the learning agent to ensure that the states with a low DUQ are avoided. This new reward function is defined in Equation 7.

$$r_{DUQ}(s,a,s') = r_{original}(s,a,s') \cdot W_p^{DUQ}(s') \tag{7}$$

where $W_p^{DUQ}(s) = \tau_p(s) \cdot DUQ(s)$

4.3 DUQ Algorithm

Given the previous definitions, the steps necessary to obtain the collection of centroids and their Voronoi regions follow Algorithm 1. The clustering process is applied on the state space, in this case, on the set of states s. K-means or mini batch K-means can be used as a clustering technique.

Algorithm 1 Learn discretization

```
Require: $D$ (experience tuples), $k$ (number of clusters)
1: $n = 0$, $D' = \emptyset$
2: while $n < |D|$ do
3:     Take $s_n$ from $(s_n, a_n, s_{n+1}, r_n) \in D$
4:     $D' = D' \cup s_n$
5:     $n = n + 1$
6: end while
7: Learn discretization $C = \{c_1, c_2, ..., c_k\}$ and partition $R = \{r_1, r_2, ..., r_k\}$ from $D'$ using k-means (see Section 4.1)
8: Return $C$
```

Once the clusters and their discretizations have been obtained, the training data can be transformed according to Algorithm 2. For each tuple in the data, it is queried to which region $v = R$ the state s_n belongs. When this value is obtained, $W_p^{DUQ}(s_n)$ is applied to s_n. $DUQ(s_n)$ is the count $|v|$ of instances in the region and $\tau_p(s_n)$ is the sigmoidal reshape. The original tuple reward is multiplied by $W_p^{DUQ}(s_n)$ as indicated in Section 4.2. The new dataset D' can be used to train with any Offline RL algorithm.

Algorithm 2 Apply DUQ to dataset

```
Require: $D$ (experience tuples), $C$ (centroids)
1: $n = 0, D' = \emptyset$
2: while $n < |D|$ do
3:     Take $t = (s_n, a_n, s_n + 1, r_n) \in D$
4:     Compute region $v = R(C(s_n))$
5:     $W_p^{DUQ}(s_n) = \tau_p(s_n) \cdot DUQ(s_n)$ with $DUQ(s_n) = |v|$
6:     $r_{DUQ} = r_n \cdot W_p^{DUQ}(s_n)$
7:     $D' = D' \cup (s_n, a_n, s_{n+1}, r_{DUQ})$
8: $n = n + 1$
9: end while
10: Return $D'$
```

4.4 Implementation

Below, we describe the steps we follow to code our approach.

1. To calculate the DUQ metric is necessary to **discretize the state space** where frequency of states can be computed. Therefore, this step consists of the normalization and discretization of that data. As introduced above, we have chosen to apply k-means or mini-batch k-means, depending on the dataset size, implemented in *scikit-learn* [27]. The appropriate number of clusters depends on the dataset used. In our experiments, we have obtained several combinations based on the distortion observed in the discretization process.

2. Next, the **DUQ metric** is computed for all the datasets based on the number of instances by cluster. Since this measure will be used as a weight to reshape the reward, it has to be normalized in the range [0,1]. Moreover, to make it even smoother and avoid abrupt changes between different clusters, a transformation with a sigmoidal function is applied to the number of instances of each cluster, as defined above.

   The range of the DUQ metric, i.e. the minimum and maximum value of the distribution, modifies the shape of the final distribution obtained after applying the sigmoidal function. This range is used as a parameter to experiment with different shapes. To define this parameter, it is necessary to select a minimum and maximum value to scale the distribution. Once the transformation has been applied, we get the previously defined W^{DUQ}.

   In Figure 3, we show a couple of examples of the uncertainty measure by cluster processed and ready to use in the training process, i.e. the W^{DUQ}. In these figures, the number of instances by cluster have been ordered from lowest to highest. In this examples we use different range for the DUQ metric. The first example is much smoother than the second since in most of the clusters the original reward is preserved ($W^{DUQ} = 1$). However, in the second one, it is penalized more frequently. This form has usually shown the best results, as described in the following section.

5 Evaluation

In this section, we summary the evaluation performed to evaluate the performance of DUQ when compared with other approaches. First, the domains and datasets (see Section 5.1). Second, the experimental setup where relevant hyperparameters are described and how the experimentation was carried out (see Section 5.2). Finally, the results are reviewed providing the learning curves and the parameters used (see Section 5.2).

5.1 Domains and datasets

There are many domains modeled as environments under the Gym [28] library, but only a part of them have good offline datasets to perform Offline Reinforcement Learning. In data-driven deep RL, the D4RL datasets [6] are widely employed by the research community in numerous recently published papers. Although we have run experiments in other classic domains, such as CartPole or Mountain Car[28], we present the results of those from D4RL to seamlessly compare with other works.

Specifically, we have obtained results from the MuJoCo environments Hopper, HalfCheetah, Walker2D and Ant which have available datasets in D4RL. The space and action spaces in these environments are continuous, so getting good results is more challenging than in discrete classic environments. D4RL offers different types of datasets for the same environments. In our case, we have chosen the option called *Medium*. In this option, the data is generated by first training a policy using SAC, then early-stopping the training, and finally collecting 1M samples from this partially-trained policy. The reason we use this type of data is that we want to achieve good results from experience tuples obtained from a poor policy, which is more similar to what happens in real-world problem-solving. If very good policies are available, other approaches as behavioural cloning have more sense.

5.2 Experimental setup

The experiments have been performed using the current state-of-the-art algorithms CQL, BCQ and SAC in an offline manner, and modifying their parameters. We have decided to compare our approach with CQL and BCQ since they are amongst the first well-known data-driven methods for Of-

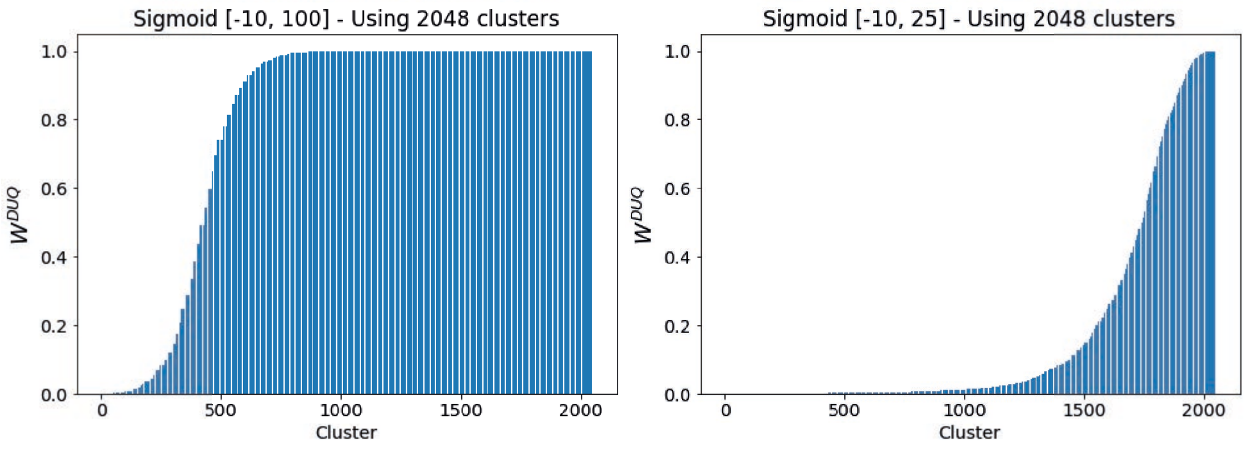


Figure 3. Uncertainty measure by cluster ready to use in the training process.

fline RL. There are many parameters to consider; those related to our approach are:

1. **Number of clusters:** It is the amount of clusters used to group the data and calculate the uncertainty measure (k parameter of k-means algorithm); i.e. then number of Voronoi Regions.

2. **Input range of the sigmoidal:** It is the range used to normalize the number of instances by cluster to calculate the uncertainty measure, as shown in Equation 6.

3. **Common parameters:** It is also necessary to experiment with other parameters, such as learning rate, batch size, or number of episodes, to name a few. Parameters values used in our experiments are described in the following section (see Section 5.2).

In order to run the experiments, the implementation of the algorithms used has been provided by d3rlpy [29]. This library is an Offline Deep Reinforcement Learning API for researches which implements state-of-the-art algorithms such as CQL or BCQ.

The experimentation carried out consists of replicating the execution of CQL, BCQ and SAC in an offline manner alongside the execution using the modifications produced by DUQ. Thus we evaluate DUQ solution against the true performance of these algorithms in the common D4RL benchmark. Reproducing the baseline gives us a better understanding of the performance of the algorithms and allows us to have more references than the results published in the source papers.

The results will be formatted as D4RL has established by normalizing final scores between 0 (performance given by a random policy) and 100 (performance given by an expert policy). These two values are provided by the benchmark. The results are going to be the average performance of two independent executions (random seeds).

In addition to numerical value reports, we plot the average performance of the policy over training time. Every point drawn on the plots is the average accumulated reward earned over 10 test episodes performed in an online manner following the decisions of the policy that is being trained.

In order to facilitate the reproducibility of the experiments, we show the parameters used for the final experiments in Appendix 6.

The **actor and critic learning rate** and the **batch size** are not listed in this table since we use the same values for each algorithm. CQL uses 0.0001 for actor and 0.0003 for critic learning rate, BCQ uses 0.001 for actor and critic learning rate and SAC uses 0.0003 for actor and critic learning rate. We used a batch size of 256 for all experiments.

The number of clusters and the sigmoidal range for a specific task are obtained through experimentation. The negative value of the range controls the lower bound of the metric (a larger negative value means more regions scored as 0), while on the other hand, the positive value controls the upper bound (larger values mean more regions marked as 1).

Table 1. Experimental results

Algorithm	Reward	**(Mean ± SD)**			
		hopper-medium	halfcheetah-medium	walker2d-medium	ant-medium
CQL	original	79.84 ± 4.98	24.38 ± 1.69	75.73 ± 0.92	4.77 ± 0.16
	DUQ	30.01 ± 0.03	**40.21 ± 5.42**	**81.35 ± 0.26**	**18.33 ± 3.25**
BCQ	original	74.48 ± 2.95	41.09 ± 0.07	69.55 ± 2.9	75.99 ± 3.46
	DUQ	30.25 ± 0.05	**43.29 ± 0.02**	**72.17 ± 1.70**	**90.35 ± 0.43**
SAC	original	0.71 ± 0.04	10.96 ± 3.03	1.89 ± 1.61	-46.72 ± 0.17
	DUQ	0.75 ± 0.02	**41.63 ± 1.64**	1.61 ± 1.42	-46.62 ± 0.13

The results have been normalized according to the D4RL paper [6]. We highlight the experiments in which our approach obtains better results.

5.3 Experimental results

In Table 1, we show the numeric results obtained by executing the CQL, BCQ and SAC algorithms in the four different domains, both using the original reward and our modified reward to account for uncertainty of the dataset. We have obtained the score and standard deviation values from two independent executions. As it can be seen, the modified reward (DUQ) improves the final results in 3 of the 4 domains tested in most cases. Moreover, in a couple of tests, the improvement is quite significant compared with the original result.

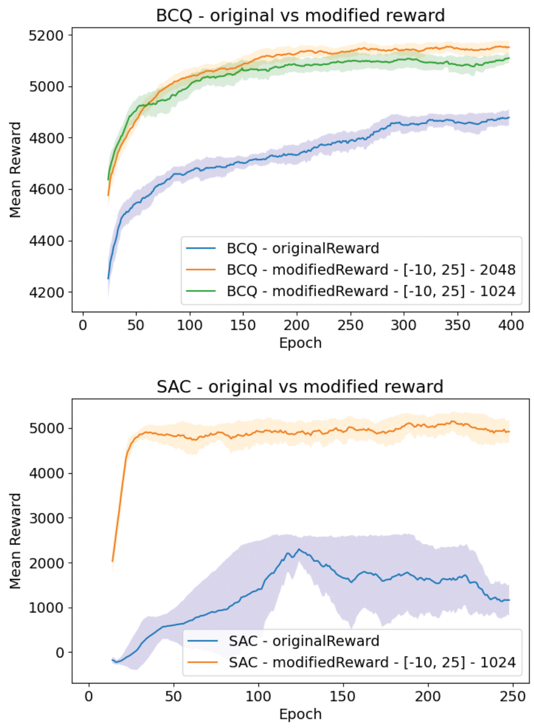
For a more detailed view of the behavior of the experiments performed, we present Figures 4, 5 and 6. These graphs show the evolution of the reward obtained during the agent's learning. The horizontal axis represents the training epochs, with an epoch being each time the algorithm has worked with the entire dataset. On the other hand, the vertical axis represents the average accumulated reward earned over 10 test episodes. In order to smooth the noise in the learning curves and improve visualization a moving average is applied. Therefore, the learning evolution curves that are being trained with different dataset setups are drawn on plots. Original Reward refers to the original execution of the algorithm without changes in the dataset and, on the other side, modified reward refers to executions that are using datasets modified by DUQ. Check the above section for more details about the plots.

From these graphs, we can draw several conclusions. The first finding is that, with varying degrees of success, our modified reward consistently beats the original reward. The second one is that our modification reaches high and stable scores more rapidly than the unmodified baseline. The following sections discuss the results obtained in each of the environments.

Hopper environment is the only one in which the DUQ technique is not able to improve previous results. Mujoco Hopper is the simplest of the environments studied (fewer articulations and dimensions in the action space). One of the reasons for these results may be the simplicity of the environment leading to the identification of unknown regions worsening performance rather than enhancing it.

A common issue in the experiments carried out and also, in the results given by D4RL, is the poor performance of SAC in an offline manner. This may be due to the aforementioned concept that algorithms of an online nature fail when working with batch data.

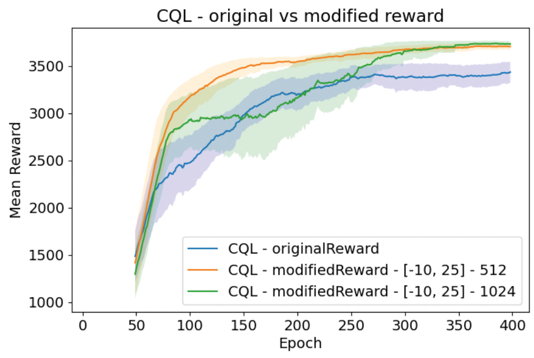
In Halfcheetah environment, DUQ modifications improve the performance given by the three techniques tested. Particularly interesting is the large improvement in the performance of the SAC algorithm. Figure 4 depicts the reward evolution during training in the HalfCheetah domain using BCQ and SAC algorithms. In the same plot, we see the experiment that utilizes the original and the modified reward

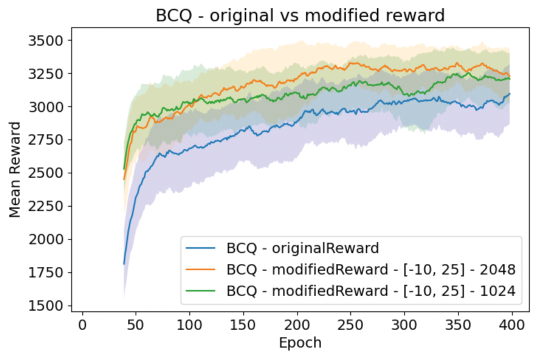


Nomenclature of experiments in legend: Algorithm (BCQ) - type of experiment (modifiedReward) - Input range of the sigmoidal ([-10,25]) - Number of clusters (1024)

Figure 4. HalfCheetah Medium Dataset. BCQ and SAC. Original and modified reward (DUQ) comparison.

In the experiments run on the Walker2D environment, it is relevant to note that the performance of CQL and BCQ is improved with DUQ, and the results are good and close to the optimal value provided by D4RL. Figure 5 shows the reward evolution during training in the Walker2D domain using CQL and BCQ algorithms.

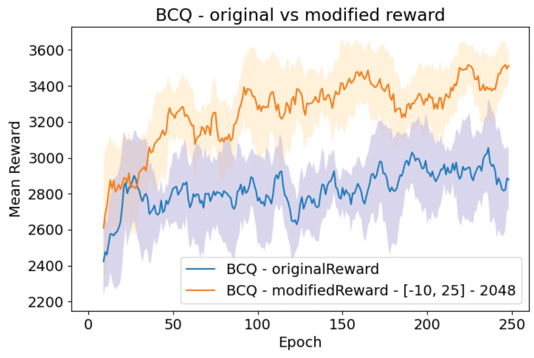




Nomenclature of experiments in legend: Algorithm (CQL) - type of experiment (modifiedReward) - Input range of the sigmoidal ([-10,25]) - Number of clusters (512)

Figure 5. Walker 2D Medium Dataset. CQL and BCQ. Original and modified reward (DUQ) comparison.

Working with Ant is complicated and the literature does not usually offer results with this environment. However, testing DUQ on data from this environment provides more evidence and allows the mechanism to be evaluated in more complicated situations. The Ant's state space consists of 27 continuous variables that describe the angles, velocities, and positions of Ant joints. DUQ improves performance over the original executions of CQL and BCQ. Figure 6 shows the training evolution in the Ant domain using BCQ.



Nomenclature of experiments in legend: Algorithm (BCQ) - type of experiment (modifiedReward) - Input range of the sigmoidal ([-10,25]) - Number of clusters (2048)

Figure 6. Ant Medium Dataset. BCQ. Original and modified reward (DUQ) comparison.

6 Conclusions

In this paper, we present Discrete Uncertainty Quantification Approach for Offline RL (DUQ), which provides a simple approach to partially mitigate the distributional shift through an effortless uncertainty quantification method. Additionally, we have also shown how it is easily applicable on top of Offline RL algorithms. Our experiments show that our approach improves the performance in several MuJoCo environments compared to standard benchmarks from D4RL. We believe that DUQ can be used extensively as an additional technique to any other, helping to achieve better and more stable results.

As future work, we consider the option of calculating the uncertainty measure by discretizing not only the states, but also the actions, so that state-action pairs are considered. In this way, it would be possible to evaluate which actions are more or less known in a state. Additionally. the team is working in incorporate DUQ main ideas in model based Offline RL techniques, applying the approach in the detection of state space unknown regions.

Acknowledgement

This work has been performed under contract agreement between Universidad Carlos III de Madrid and Repsol. Fernando Fernández is partially funded by *Programa del Ministerio de Universidades para estancias de movilidad de profesores e investigadores en centros extranjeros de enseñanza superior e investigación*. It has also been supported by AEI/FEDER (European Union) Grant number PID2020-119367RB-I00.

References

[1] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

[2] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In Reinforcement learning, pages 45–73. Springer, 2012.

[3] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. arXiv preprint arXiv:2203.01387, 2022.

[4] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

[5] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. arXiv preprint arXiv:2005.13239, 2020.

[6] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. Deep Learning Workshop NIPS 2013, 2013.

[8] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6):1291–1307, 2012.

[9] Chip Huyen. Data distribution shifts and monitoring.

[10] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion, 76:243–297, 2021.

[11] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. arXiv preprint arXiv:2005.05951, 2020.

[12] Katiana Kontolati, Dimitrios Loukrezis, Dimitris Giovanis, Lohit Vandanapu, and Michael Shields. A survey of unsupervised learning methods for high-dimensional uncertainty quantification in black-box-type problems. Journal of Computational Physics, 464:111313, 05 2022.

[13] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. pages 2052–2062, 2019.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[15] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, page 11784–11794, 06 2019.

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

[17] Hongwen He, Zegong Niu, Yong Wang, Ruchen Huang, and Yiwen Shou. Energy management optimization for connected hybrid electric vehicle using offline reinforcement learning. Journal of Energy Storage, 72:108517, 2023.

[18] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2006.04779, 2020.

[19] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. ICML'20: Proceedings of the 37th International Conference on Machine Learning, pages 104–114, 2019.

[20] Phillip Swazinna, Steffen Udluft, Daniel Hein, and Thomas Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning. IFAC-PapersOnLine, 55(15):19–26, 2022.

[21] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. arXiv preprint arXiv:1906.08253, 2019.

[22] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. arXiv preprint arXiv:2102.08363, 2021.

[23] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. 2020.

[24] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando deFreitas. Rl unplugged: Benchmarks for offline reinforcement learning, 2020.

[25] Allen Gersho. Vector quantization and signal compression. Kluwer international series in engineering and computer science. Communications and information theory. : Kluwer Academic, Boston, 1992.

[26] S. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, 28(2):129–137, 1982.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

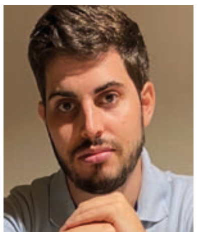
[28] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

[29] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In NeurIPS 2021 Offline Reinforcement Learning Workshop, December 2021.

Sergio Pérez is a Research Scientist at Repsol Technology Lab. He received his M.Sc. in Telecommunication Engineering from UPM in 2020. His research interests include Deep Learning and Reinforcement Learning applied to real-world industrial applications. https://orcid.org/0000-0002-2191-4974

Javier Corrochano Jiménez is an MSc in Computer Science Engineering. During his tenure at the Planning and Learning Group at Universidad Carlos III de Madrid, he contributed to the research, development, and implementation of algorithms in the field of offline reinforcement learning. Currently, as a Computer Vision Engineer, he is actively involved in cutting-edge research projects for major infrastructure companies.
https://orcid.org/0000-0003-3687-700X

Javier García is an associate professor of the Electronics and Computer Science Department at the Universidad de Santiago de Compostela, since september 2023. He obtained a PhD that received the distinguished Thesis Award at the Computer Science Department of Universidad Carlos III de Madrid in 2013. He was also a postdoctoral researcher under the Talent Attraction Program from the Comunidad de Madrid since 2016 until 2020 in the Computer Science Department of the Universidad Carlos III de Madrid. He has more than 40 journal and conference papers, mainly in the field of machine learning, automated planning and robotics. Currently, he is associate editor of the journal IEEE Robotics and Automation Letters and member of the editorial board of the journal Machine Learning.
https://orcid.org/0000-0002-5638-5240



Ruben Majadas graduated from the University Carlos III of Madrid. He is currently a PhD student, focusing on adversarial reinforcement learning methods from both attack and defense perspectives for his doctoral dissertation. His research interests encompass machine learning, particularly reinforcement learning, optimization, and adversarial networks.
https://orcid.org/0000-0002-3395-7330

Cristina Ibanez-Llano received an BSc degree in Mathematics from Universidad Autonoma de Madrid in 2002, a MSc in Mathematical Engineering in 2003 from Universidad Complutense de Madrid and a PhD in Industrial Engineering in 2010 from Universidad Pontificia Comillas. She has been working in the fields of energy both in the public and private sectors for over 20 years in many different areas (nuclear safety, oil and gas and industrial processes). Her main areas of interest are related with all different families of optimization techniques including stochastic optimization, mathematical and dynamic programming, reinforcement learning and derivative free methods.
https://orcid.org/0009-0003-1917-3382

José Luis Pérez Torres is a graduate of the master's and bachelor's degree in computer engineering from the Universidad Carlos III de Madrid. He worked in the Machine Learning and Planning department at the same university, known as PLG, where he focused on studying the field of Offline Reinforcement Learning. During his stay, he conducted research to address existing limitations in the field, such as distributional shift, uncertainty quantification, optimal space exploration and model-based approximations, among others. This marks the researcher's inaugural article publication, with him serving as the lead author.
https://orcid.org/0009-0002-8623-3595

Fernando Fernández is an associate professor of the Computer Science Department at Universidad Carlos III de Madrid, since october 2005. He received his Ph.D. degree in Computer Science from University Carlos III of Madrid (UC3M) in 2003. In the fall of 2000, Fernando was a visiting student at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). He was also a postdoctoral fellow at the Computer Science Department of Carnegie Mellon University since october 2004 until december 2005 and Visiting Researcher at Texas Robotis, University of Texas at Austin in 2022-2023. He is the recipient of a pre-doctoral FPU fellowship award from Spanish Ministry of Education (MEC), a Doctoral Prize from UC3M, and a MEC-Fulbright postdoctoral Fellowship as well as the 2020 and 2021 JPMorgan AI Research Awards. He has more than 80 journal and conference papers, mainly in the field of machine learning, automated planning and robotics. He is interested in intelligent systems that operate in continuous and stochastic domains. He is the Director of the Planning and Learning Group of the Computer Science Department at UC3M. He is also co-founder and CSO of Inrobocs Social Robotics, where social assistive robots are deployed in rehabilitation hospitals.
https://orcid.org/0000-0003-3801-6801

Apendix A Experimental parameters

Table 2. Experimental parameters

Domain	Reward	Algorithm	epochs	N^o Clusters	Sigm. range
hopper-medium	original	CQL	400	-	-
hopper-medium	original	BCQ	400	-	-
hopper-medium	original	SAC	250	-	-
hopper-medium	DUQ	CQL	400	1024	[-10,25]
hopper-medium	DUQ	BCQ	400	1024	[-10,25]
hopper-medium	DUQ	SAC	250	1024	[-10,25]
halfchetah-medium	original	CQL	400	-	-
halfchetah-medium	original	BCQ	400	-	-
halfchetah-medium	original	SAC	250	-	-
halfchetah-medium	DUQ	CQL	400	1024	[-10,25]
halfchetah-medium	DUQ	BCQ	400	2048	[-10,25]
halfchetah-medium	DUQ	SAC	250	1024	[-10,25]
walker2d-medium	original	CQL	400	-	-
walker2d-medium	original	BCQ	400	-	-
walker2d-medium	original	SAC	250	-	-
walker2d-medium	DUQ	CQL	400	512	[-10,25]
walker2d-medium	DUQ	BCQ	400	2048	[-10,25]
walker2d-medium	DUQ	SAC	250	1024	[-10,25]
ant-medium	original	CQL	-	-	-
ant-medium	original	BCQ	-	-	-
ant-medium	original	SAC	-	-	-
ant-medium	DUQ	CQL	250	4096	[-10,25]
ant-medium	DUQ	BCQ	250	1024	[-10,25]
ant-medium	DUQ	SAC	250	1024	[-10,25]

***N^o Clusters**: Number of Clusters, **Sigm. range**: Input range of the sigmoidal function*