Discrete Uncertainty Quantification Approach for Offline RL

Javier Corrochano[1] **Javier García**[2] **Rubén Majadas**[1] **Cristina Ibanez-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, Spain

Abstract

In many Reinforcement Learning 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 Reinforcement Learning. However, this type of learning faces a large number of challenges, mostly derived from the fact that exploration/exploitation trade-off is overshadowed. Instead, 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. Specifically, in this paper we present a new approach to deal with the uncertainty risen by the absence or sparse presence of some states in the data. Our approach is based on shaping the reward signal of 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 significantly.

1 Introduction

In many Reinforcement Learning (RL) tasks, the classical online interaction between the learning agent and the environment 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 refinement process. This way of learning from a batch of experiences without exploration in order to obtain the best possible policy given the data has been referred to as batch RL, offline RL, or data-driven RL [1, 2].

Although some of the most common methods within RL can learn from off-policy data, following an experience replay approach, they 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 off-line RL, broken due to some new challenges. A fundamental one is related to the distributional shift, also called out-of-distribution states and actions [3, 4, 1]. Distributional shift is 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.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021

Distributional shift issues can be addressed in several ways and can be classified into two groups:

- **Policy Constraint:** It mitigates distributional shift constraining the learned policy to be close to the behaviour policy [1].
- **Uncertainty based:** These solutions attempt to estimate the epistemic uncertainty of Q values an then use it to detect distributional shift [1].

In this paper, we present a simple but effective approach to quantify the uncertainty of the dataset 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. This idea comes from a previous work in which we implemented a model-based RL framework based on learning the Markov Decission Process (MDP) model through discretisation of the states and actions.

The paper is organized as follows. Section 2 present a brief review of previous work on offline RL. Then the new approach to deal with distributional shift of the dataset through reshaping of the reward function is presented in Section 3. After this, experiment section and conclusions are presented in Section 4 and 5. The experiments section shows the results obtained by introducing this reward reshaping with different algorithms in three different domains from the D4RL benchmark datasets.

2 Related work

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.

Online RL: Although online algorithms are not specifically designed for offline applications, they have been used in the past. A widely adopted solution in continuous control tasks is Soft Actor-Critic (SAC) [5]. SAC is a model-free and off-policy RL model that maximizes both the expected reward and entropy. The most concerning issue with online methods is that they are not devised for static datasets, thereby not addressing distributional shift. Our solution provides a simple and effective way of quantifying uncertainty for adapting them to offline scenarios.

Model-free offline RL: Fujimoto et al. [6] 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 introduce 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. Kumar et al. [7] propose the Conservative Q-learning (CQL) algorithm capable of learning from a fixed dataset and without further interaction. 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.

Model-based offline RL: Model-Based Policy Optimization (MBPO) [8] 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) [9], Model-based Offline Policy Optimization (MOPO) [4] and Conservative Offline Model-Based Policy Optimization (COMBO) [10].

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 MOPO and MoReL use a measure of uncertainty, 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 [7] into the model-based setting. 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 Discrete Uncertainty Quantification Approach for Offline RL (DUQ)

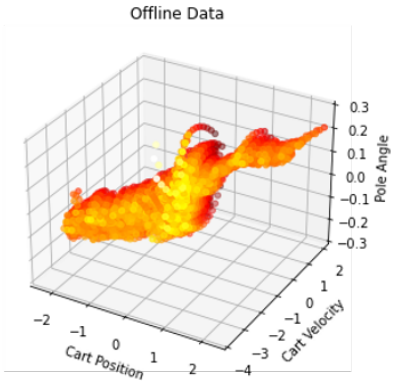
DUQ is a simple approach to quantify uncertainty using the discretization of states. This information is received by the agent through the reward and used in the learning process to avoid less-known regions. In the following section, we motivate this concept and include its formal definition

3.1 Motivation

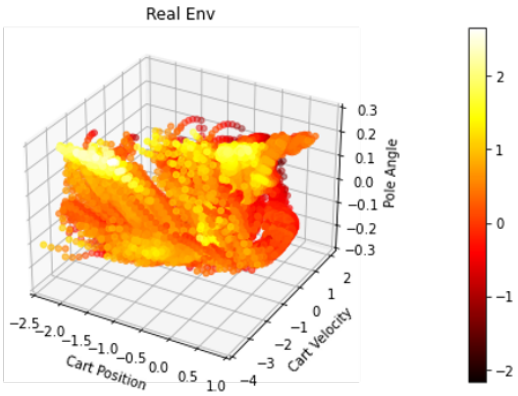
In Figure 1, we show two different data distributions of the Cartpole domain. The first distribution, Figure 3a, is an exploration done during training with offline data. The second one, Figure 3b, is a random exploration performed in the environment of the Cartpole domain. As seen in these images, the 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 little or not at all known by the offline dataset. If an agent learns a policy using the offline dataset, there will be less-known regions, and it will act blindly.

Figure 1: Exploration comparison.

(a) Exploration in Discrete Environment



(b) Exploration in Continuous Environment



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 the all 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 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 should be the region and therefore the original reward is reduced little or not at all. 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.

3.2 Definitions - DUQ Metric

Given a dataset $D = \{d_1, ..., d_m\}$, given a set of centroids $C = \{C_1, ..., C_n\}$, and given the Voronoi regions generated by the set C over the dataset D, $R = \{r1, ..., r_n\}$ being $r_i = \{d_j | dist(d_j, c_i) \leq dist(d_j, c_k), k \neq i\}$. Then, DUQ metric can be defined as Equation 1 shows:

$$DUQ(d_j) = |r_i|, d_j \in r_i \tag{1}$$

In different words, the DUQ of an instance is the number of instances of its corresponding Voronoi region.

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 2. Finally, the reshaped reward is defined in Equation 2.

$$r_{DUQ}(s, a, s') = r_{original}(s, a, s') \cdot W_p^{DUQ}(s') \tag{2}$$

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

3.3 Implementation

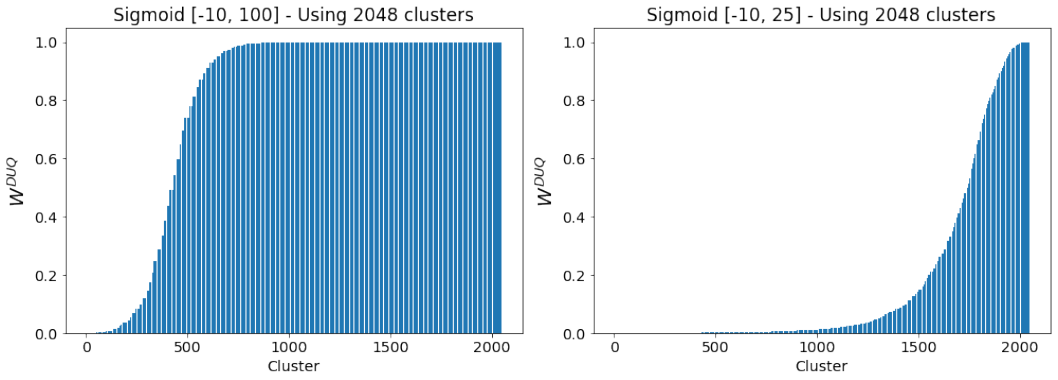
Below, we describe the steps we follow to apply our approach.

1. The first step is the **data acquisition**. Any dataset from a OpenAI Gym domain (or any other RL task) could be valid, but we have chosen those from D4RL [11]. They are widely adopted in RL publications, allowing the community to standardize benchmarks.
2. To calculate the DUQ metric is necessary to **discretize the state space** where frequency of states can be computed. Therefore the second step consists on 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* [12]. 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.
3. 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 2, 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 that figures, the number of instances by cluster have been ordered from highest to lowest. 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.

Figure 2: Uncertainty measure by cluster ready to use in the training process.



3.4 DUQ Algorithm

At this point, we can **launch the training process**. We have tested our approach using well known algorithms, such as CQL, BCQ, or SAC, comparing the results of using the original reward function and the DUQ reward. We show the results in the experimentation (see Section 4).

During the training, the agent visits one state $s \in S$, selects an action $a \in A$ and goes to another state $s' \in S$ at each step. Our approach modifies the reward, using the previously defined method the new reward as defined in Equation 2.

4 Experimentation

In this section, we show the different aspects to consider for experimentation. On the one hand, the domains and datasets (see Section 4.1), on the other hand, the experimental parameters, which are defined in the experimental results (see Section 4.2).

4.1 Domains and datasets

There are many domains modeled as environments under the OpenAI Gym [13] library, but only a part of them have good offline datasets to perform Offline Reinforcement Learning. It is crucial to share standard datasets to evaluate algorithms. In data-driven deep RL, the D4RL datasets [11] are widely employed by the research community in numerous recently published papers. Although, we have run experiments in other classic domains [13], such as CartPole or Mountain Car, 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, and Walker2D, which have available datasets in D4RL. 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 are generated by first training a policy using SAC, then early-stopping the training, and finally collecting 1M samples from this partially-trained policy.

4.2 Experimental results

The experiments have been performed using the current state of the art algorithms CQL, BCQ, and SAC, and modifying their parameters. We have decided to compare our approach with CQL and BCQ since they are among the first well-known data-driven methods for offline 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).
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 1.
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.

In Table 1, we show the numeric results obtained by executing the CQL, BCQ, and SAC algorithms in three 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 2 of the 3 domains tested in most cases. Moreover, in a couple of tests, the improvement is quite significant compared with the original result.

In order to facilitate the reproducibility of the experiments, we show the parameters used for the final experiments in Table 2. 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.

For a more detailed view of the behavior of the experiments performed, we present Figures 3 and 4. These graphs show the evolution of the reward obtained during the agent's learning. Figure 3 depicts

Table 1: Experimental results

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

The results have been normalized according to the D4RL paper [11]. We highlight the experiments in which our approach obtains better results.

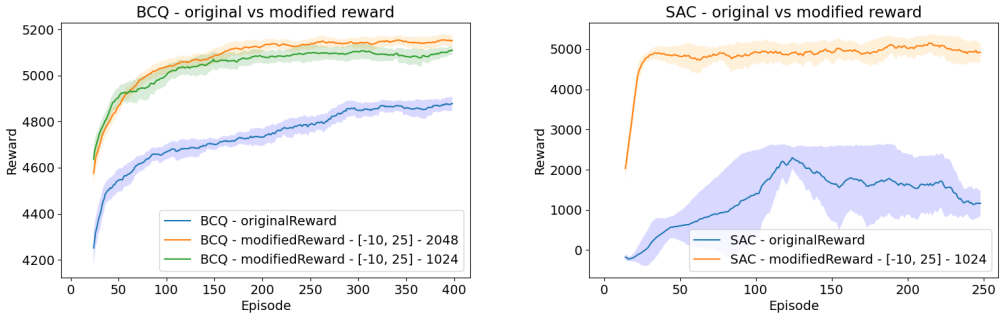
Table 2: Experimental parameters.

Domain	Reward	Algorithm	epochs	Nº 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]

Nº Clusters*: Number of Clusters,* ***Sigm. range****: Input range of the sigmoidal function*

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. In the same way, Figure 4 shows the reward evolution during training in the Walker2D domain using CQL and BCQ algorithms. 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.

Figure 3: HalfCheetah Medium Dataset. BCQ and SAC. Original and modified reward (DUQ) comparison.



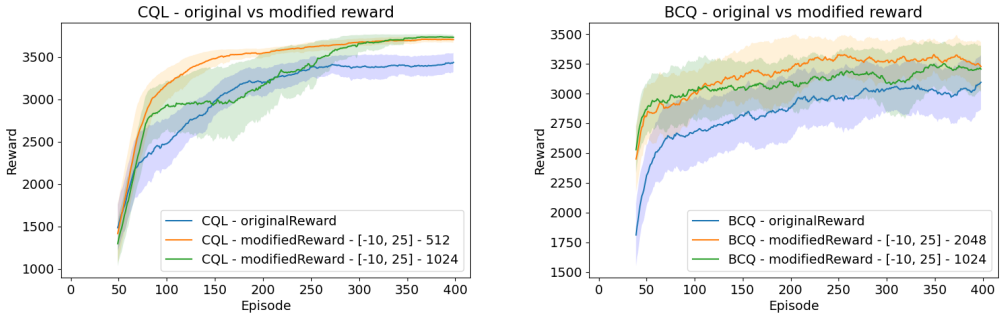
Nomenclature of experiments in legend: Algorithm (BCQ) - type of experiment (modifiedReward) - Input range of the sigmoidal ([-10,25]) - Number of clusters (1024)

5 Conclusions

In this paper, we present DUQ (Discrete Uncertainty Quantification Approach for Offline RL), 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.

Figure 4: Walker 2D Medium Dataset. CQL and BCQ. Original and modified reward (DUQ) comparison.



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

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

[4] 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.

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

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

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

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

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

[10] 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.

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

[12] 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.

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