Batch Reinforcement Learning from Crowds

Guoxi Zhang✉[1] and Hisashi Kashima[1,2]

[1] Graduate School of Informatics, Kyoto University
guoxi@ml.ist.i.kyoto-u.ac.jp, kashima@i.kyoto-u.ac.jp
[2] RIKEN Guardian Robot Project

Abstract. A shortcoming of batch reinforcement learning is its requirement for rewards in data, thus not applicable to tasks without reward functions. Existing settings for the lack of reward, such as behavioral cloning, rely on optimal demonstrations collected from humans. Unfortunately, extensive expertise is required for ensuring optimality, which hinder the acquisition of large-scale data for complex tasks. This paper addresses the lack of reward by learning a reward function from preferences between trajectories. Generating preferences only requires a basic understanding of a task, and it is faster than performing demonstrations. Thus, preferences can be collected at scale from non-expert humans using crowdsourcing. This paper tackles a critical challenge that emerged when collecting data from non-expert humans: the noise in preferences. A novel probabilistic model is proposed for modelling the reliability of labels, which utilizes labels collaboratively. Moreover, the proposed model smooths the estimation with a learned reward function. Evaluation on Atari datasets demonstrates the effectiveness of the proposed model, followed by an ablation study to analyze the relative importance of the proposed ideas.

Keywords: Preference-based Reinforcement Learning· Crowdsourcing

1 Introduction

Batch Reinforcement Learning (RL) [Lange et al., 2012] is a setting for RL that addresses its limitation of data acquisition. Online RL needs to generate new data during learning, either via simulation or physical interaction. However, simulators with high fidelity are not always available, and real-world interactions raise safety and ethical concerns. Batch RL instead reuses existing vastly available data, so it has received increasing attention in recent years [Pavse et al., 2020, Agarwal et al., 2020, Gelada and Bellemare, 2019, Kumar et al., 2019, Fujimoto et al., 2019].

In batch RL, the data consists of observations, actions, and rewards. For example, in recommender systems, the observations are user profiles, and the actions are items to be recommended. The rewards are scalars, evaluating actions for their consequences on achieving a given task. The mapping from observations and actions to rewards is called a reward function. Often, the sequence

of observations, actions, and rewards generated during interaction is called a trajectory.

While batch RL is a promising data-driven setting, its dependence on rewards can be problematic. Many tasks of interest lack a reward function. Consider an application to StarCraft, a famous real-time strategy game, for example. Typically, evaluative feedback is given for the final result of a series of action sequences (i.e., the entire trajectory), not for individual actions about their contributions to the result. In the RL literature, the lack of reward has been addressed by inverse RL [Abbeel and Ng, 2004] or Behavioral Cloning (BC) [Schaal, 1996], which eliminate the need for rewards by leveraging expert demonstrations. However, their assumption on demonstrations can be hard to satisfy. Optimal demonstrations require extensive expertise. In practice, the competency of human demonstrators differs [Mandlekar et al., 2019], causing RL and BC algorithms to fail [Mandlekar et al., 2021].

This paper addresses the lack of reward signals by learning a reward function from preferences. A *preference* is the outcome of a comparison between two trajectories for the extent the task is solved. Compared to demonstrations, providing preferences requires less human expertise. For example, demonstrating the moves of professional sports players is difficult, but with general knowledge a sports fan can still appreciate the moves in games. Hence, preferences can be collected at scale from a large group of non-expert humans, possibly via crowdsourcing [Vaughan, 2017], which is the use of the vast amount of non-expert human knowledge and labor that exists on the Internet for intellectual tasks. In the RL literature, learning from preferences is discussed as Preference-based RL (PbRL) [Wirth et al., 2017]. Recent advances show that agents can solve complex tasks using preferences in an online RL setting [Christiano et al., 2017, Ibarz et al., 2018]. This paper extends PbRL to a batch RL setting and focuses on the following challenge: How to learn a reward function from noisy preferences?

Denoising becomes a major requirement when preferences are collected using crowdsourcing. Crowd workers can make mistakes due to the lack of ability or motivation, thus data generated with crowdsourcing is often very noisy. Similar observation is made when collecting demonstrations from the crowd. Mandlekar et al. [2018, 2019] discovers that collected demonstrations hardly facilitate policy learning due to noise in demonstrations. However, this challenge has been overlooked by the PbRL community. The reason is that, in an online setting, preferences are often collected from recruited collaborators, so their quality is assured. Meanwhile, the present study assumes preferences are collected from non experts, and little is known or can be assured about the annotators.

This paper proposes a probabilistic model, named Deep Crowd-BT (DCBT), for learning a reward function from noisy preferences. DCBT assumes that each preference label is potentially unreliable, and it estimates the label reliability from data. As shown in Figure 1, the idea behind DCBT is to model the correlation of the reliability of a label with its annotator, other labels for the same query, and the estimated reward function. The conditional dependency on the annotator models the fact that unreliable annotators tend to give unreliable la-

bels. Meanwhile, as it is a common practice to solicit multiple labels for the same query, the proposed model collaboratively utilizes labels from different annotators for the same query. Yet in practice each query can only be labelled by a small group of annotators given a fixed labelling budget, so DCBT also utilizes the estimated reward function effectively smooths the label reliability.

A set of experiments on large scale offline datasets [Agarwal et al., 2020] for Atari 2600 games verifies the effectiveness of DCBT. The results show that DCBT facilitates fast convergence of policy learning and outperforms reward learning algorithms that ignore the noise. Furthermore, an ablation study is also performed to analyze the relative importance of collaboration and smoothing. The contributions of this paper are summarized as follows:

- This paper addresses the lack of reward in batch RL setting by learning a reward function from noisy preferences.
- A probabilistic model is proposed to handle the noise in preferences, which collaboratively models the reliability of labels and smooths it with the estimated reward function.
- Experiments on Atari games, accompanied by an ablation study, verify the efficacy of the proposed model.

2 Related Work

Batch RL is a sub-field of RL that learns to solve tasks using pre-collected data instead of online interaction. Efforts have been dedicated to issues that emerge when learning offline, such as the covariate shift [Gelada and Bellemare, 2019] and the overestimation of Q function [Kumar et al., 2019]. This paper addresses the lack of reward in batch RL setting, which complements policy learning algorithms.

In the literature of RL, the lack of reward is canonically addressed by either inverse RL [Abbeel and Ng, 2004] or BC [Schaal, 1996]. Inverse RL does not apply as it requires online interactions. BC suffers from the inefficiency of data acquisition and the imperfectness of collected demonstrations. While data acquisition can be scaled up using crowdsourced platforms such as the RoboTurk [Mandlekar et al., 2018, 2019], the imperfectness of demonstrations remains an issue for BC [Mandlekar et al., 2021]. Recent results show that BC can work on mixtures of optimal and imperfect demonstrations via collecting confidence scores for trajectories [Wu et al., 2019] or learning an ensemble model for behavioral policies [Sasaki and Yamashina, 2021]. These methods still require large amount of optimal trajectories.

Instead, this paper addresses the lack of reward by collecting preferences from humans. As preferences require less expertise than optimal demonstrations, they can be collected using methods such as crowdsourcing [Vaughan, 2017]. In the literature of RL, PbRL is shown to be successful for complex discrete and continuous control tasks [Christiano et al., 2017, Ibarz et al., 2018]. Interested readers may refer to the detailed survey from Wirth et al. [2017]. However, the existing work on PbRL is restricted to clean data in the online RL setting,

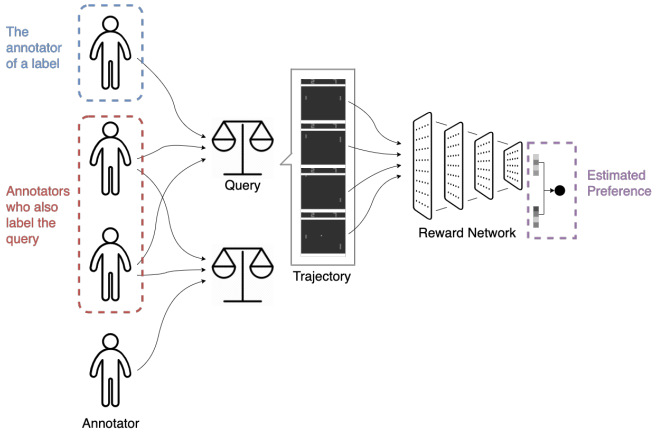

Fig. 1. This diagram illustrates information utilized by the proposed DCBT model. An arrow pointing to a query means that the annotator gives a label for the query. When determining the reliability of a label, the DCBT model utilizes: (a) the ID of the annotator of the label (the blue box), (b) the IDs of annotators who also label the query and the labels they give (the red box), and (c) an estimate for preference computed using the learned reward function. Part (b) is our idea of utilizing other labels for the same query to assist in modeling label reliability. Meanwhile, an estimate for preference can be computed with (c), which is also helpful in determining label reliability.

while crowdsourced data are often noisy [Zheng et al., 2017, Rodrigues and Pereira, 2018]. While being detrimental to reward learning [Ibarz et al., 2018], noise in preferences remains a overlooked issue. This paper extends PbRL to a batch setting and overcomes the issue of noise in preferences. It extends the propabilistic model proposed by Chen et al. [2013] to effectively model label reliability while learning a reward functions from preferences.

3 Problem Setting

3.1 Markov Decision Process

A sequential decision-making problem is modeled as interactions between an agent and an environment, described as a discounted infinite-horizon MDP: $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$, where $\mathcal{S}$ refers to the state space (we interchangeably use *states* and *observations* in this paper), $\mathcal{A}$ is a finite and discrete set of actions, and $R(s, a)$ is the reward function. $P(s'|s, a)$ is the transition probability that characterizes how states transits depending on the action, which is not revealed to the agent. $\gamma \in \mathbb{R}$ is the discount factor.

Interactions roll out in discrete time steps. At step t, the agent observes $s_t \in \mathcal{S}$ and selects action a_t according to a policy $\pi : \mathcal{S} \to \mathcal{A}$. Based on (s_t, a_t), the environment decides next state s_{t+1} according to $P(s_{t+1}|s_t, a_t)$, and the reward

$r_t \in \mathbb{R}$ is determined according to $R(s, a)$. The objective of policy learning is to find a policy π that maximizes $\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, a_t)$, the discounted sum of the rewards.

3.2 Reward Learning Problem

In this paper, trajectories are assumed to be missing from trajectories. A trajectory can be written as $\eta = (s_1, a_1, \ldots, s_{T_c}, a_{T_c})$, where T_c is the length of this trajectory. A learning agent is provided with a set of trajectories and preferences over these trajectories generated by a group of annotators. The i^{th} sample can be written as a four tuple: $(\eta_{i,1}, \eta_{i,2}, y_i, w_i)$. The pair $(\eta_{i,1}, \eta_{i,2})$ is the preference query, and y_i is the preference label. $y_i =$ "$\succ$" if in $\eta_{i,1}$ the task is solved better than in $\eta_{i,2}$, $y_i =$ "$\approx$" if the two clips are equally good, and $y_i =$ "$\prec$" if in $\eta_{i,1}$ the task is solved worse than in $\eta_{i,2}$. w_i is the ID of the annotator who gave y_i. Let N be the number of preferences, and let M be the number of annotators.

The preferences are assumed to be noisy, as the annotators may lack of expertise of commitment. Yet no information other than annotator IDs is revealed to the learning agent. From the preferences, the learning agent aims at learning a reward function $\hat{R}$. The learning problem is summarized as follows:

- Input: A set of noisy preferences $D = \{(\eta_{i,1}, \eta_{i,2}, y_i, w_i)\}_{i=1}^{N}$, where $\eta_{i,1}$ and $\eta_{i,2}$ are two trajectories, w_i is the identity of the annotator, and $y_i \in \{$"$\succ$","$\approx$","$\prec$"$\}$ is the preference label.
- Output: An estimated reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

After learning R, the trajectories are augmented with estimated rewards given by R. They are now in the standard format for off-policy policy learning, and in principle any algorithm of the kind is applicable.

4 Proposed Method

This section presents the proposed Deep Crowd-BT (DCBT) model. A diagram for DCBT is shown in Figure 1, and a pseudocode for reward learning with DCBT is provided in Algorithm 1.

4.1 Modeling Preferences

Given a sample $(\eta_{i,1}, \eta_{i,2}, y_i, w_i)$, DCBT first computes the rewards of $\eta_{i,1}$ and $\eta_{i,2}$. For each state-action pair (s, a) in $\eta_{i,1}$ and $\eta_{i,2}$, the reward network outputs a scalar $R(s, a)$. This is done by the reward network shown in the upper part of Figure 1. Let θ_R be the parameters of R. For image input, R is parameterized with convolutional neural networks followed by feedforward networks.

Then the probability of the event "$\eta_{i,1} \succ \eta_{i,2}$" is modeled with the Bradly-Terry model (BT) [Bradley and Terry, 1952]:

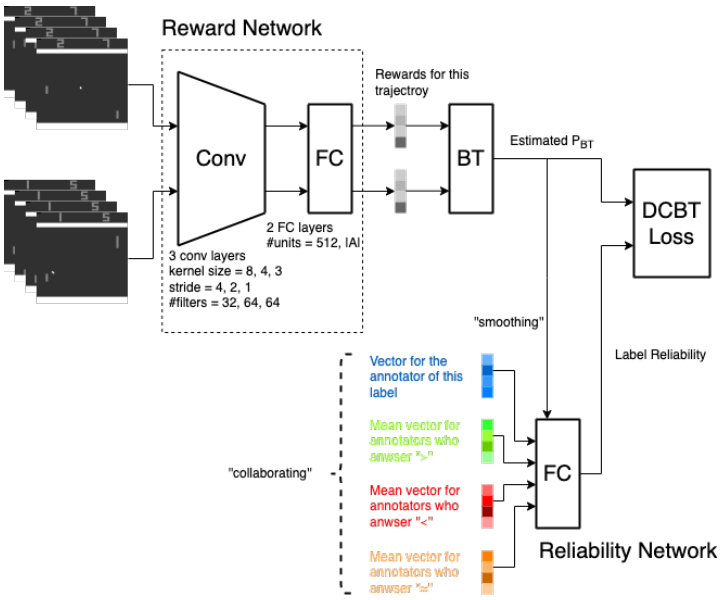

Fig. 2. This figure illustrates the architecture for learning reward function with the proposed DCBT model. After learning, the reward function (shown in dashed box) can be used to infer rewards using states and actions of trajectories.

$$\begin{aligned} \mathrm{P}_{\mathrm{BT}}\left(\eta_{i,1} \succ \eta_{i,2}\right) &= \frac{\exp\left(G(\eta_{i,1})\right)}{\exp\left(G(\eta_{i,1})\right) + \exp\left(G(\eta_{i,2})\right)}, \\ G(\eta_{i,1}) &= \frac{1}{T_c} \sum_{(s,a)\in\eta_{i,1}} R(s,a), \\ G(\eta_{i,2}) &= \frac{1}{T_c} \sum_{(s,a)\in\eta_{i,2}} R(s,a). \end{aligned} \tag{1}$$

The larger $G(\eta_{i,1})$ is, the larger $\mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2})$ becomes. $\mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2}) > 0.5$ if $G(\eta_{i,1}) > G(\eta_{i,2})$, that is, the trajectory with a larger sum of rewards is preferred. If the preferences are consistent with the task of interest, then maximizing the loglikelihood of $\mathrm{P}_{\mathrm{BT}}\left(\eta_{i,1} \succ \eta_{i,2}\right)$ will generate reward function that facilitate learning of competitive policies [Christiano et al., 2017, Ibarz et al., 2018].

4.2 Handling Noise in Preferences

When the preferences are noisy, preference labels are not entirely consistent with the underlying task. Ibarz et al. [2018] point out that policies degenerate for noisy preference data. In literature, the Crowd-BT model [Chen et al., 2013]

Algorithm 1: Reward Learning with DCBT

Input: D, a noisy preference dataset
T_{INIT}, the number of graident steps for the initialization phase
T_{TOTAL}, the total number of gradient steps
T_{ALT}, the period of alternative optimization
βT_{ALT} is the number of steps to train θ_R
Initialize θ_R, θ_W and θ_α randomly
for $t = 1, 2, \ldots, T_{\text{TOTAL}}$ **do**
  Sample a batch of preference data
  if $t \leq T_{\text{INIT}}$ **then**
    update $\theta_R, \theta_W, \theta_\alpha$ by minimizing $L_{\text{DCBT}-\text{INIT}} + \lambda_1 L_{\text{IDF}} + \lambda_2 L_{\ell_1,\ell_2}$
  end
  else
    if t mod $T_{\text{ALT}} < \beta T_{\text{ALT}}$ **then**
      update θ_R by minimizing Equation 8.
    end
    else
      update θ_W, θ_α by minimizing Equation 8.
    end
  end
end

is a probabilistic model for modeling noisy pairwise comparisons. It assumes each annotator makes errors with an annotator-specific probability. Denote by $\alpha_w = \mathrm{P}(\eta_{i,1} \succ_w \eta_{i,2} \mid \eta_{i,1} \succ \eta_{i,2})$ the probability that annotator w gives "$\succ$" for $(\eta_{i,1}, \eta_{i,2})$, when the groundtruth is $\eta_{i,1} \succ \eta_{i,2}$. With the Crowd-BT model,

$$\mathrm{P}_{\text{Crowd}-\text{BT}}(y_i = \text{“}\succ\text{"}) = \alpha_{w_i}\mathrm{P}_{\text{BT}}(\eta_{i,1} \succ \eta_{i,2}) + (1 - \alpha_{w_i})(1 - \mathrm{P}_{\text{BT}}(\eta_{i,1} \succ \eta_{i,2})). \tag{2}$$

In other words, the Crowd-BT model assumes that the reliability of labels from the same annotator is the same and fixed, in regardless of queries being compared. This assumption, however, is inadequate in our case and suffers from at least two reasons. In our case, each query is labeled by multiple annotators. The other labels for the same query and the credibility of their annotators are informative when modeling the reliability of a label, but they are not utilized in the Crowd-BT model. Meanwhile, in practice rich coverage of possible trajectories are required to ensure the generalization of R. Thus, under limited labeling budget, each preference query is labeled by a tiny group of annotators, which incurs high variance in labels.

The proposed DCBT extends Crowd-BT by explicitly overcoming the abovementioned issues. Instead of per-annotator reliability parameter, it learns a per-sample reliability defined as:

$$\alpha_i = \mathrm{P}(y_i = \text{“}\succ\text{"} \mid w_i, C_i, \mathrm{P}_{\text{BT}}(\eta_{i,1} \succ \eta_{i,2})). \tag{3}$$

The probability of $y_i =$ "$\succ$" can be expressed as

$$\mathrm{P}_{\mathrm{DCBT}}(y_i = \text{“}\succ\text{"}) = \alpha_i \mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2}) + (1 - \alpha_i)(1 - \mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2})). \quad (4)$$

For each sample, the reliability network shown in the bottom part of Figure 1 outputs label reliabilityα_i. This network is parameterized with the fully-connected layer with sigmoidal activation. Let the parameter of this network be θ_α. This network addresses the drawbacks of the Crowd-BT model by taking as input w_i, C_i and $\mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2})$.

C_i is a set that contains labels and their annotators for the same preference query $(\eta_{i,1}, \eta_{i,2})$. Specifically,

$$C_i = \{(y_j, w_j) \mid \eta_{j,1} = \eta_{i,1}, \eta_{j,2} = \eta_{i,2}, j \neq i, (\eta_{j,1}, \eta_{j,2}, y_j, w_j) \in D\}. \quad (5)$$

Using C_i, the reliability network utilizes crowdsourced preferences *collaboratively*. The intuition is that, for the same query, labels from other annotators and the credibility of these annotators provide useful information for modeling α_i. A label might be reliable if it is consistent with other labels, especially with those from credible annotators. Moreover, the label values matter. For example, suppose ("$\prec$", w_j) $\in C_i$, and w_j is a a credible annotator. Then while both "$\succ$" and "$\approx$" are inconsistent with "$\prec$", the latter should be a more reliable one. The reliability network relies on a worker embedding matrix $\theta_W \subset \mathbb{R}^d$. It takes as input the embedding vector of w_i. For C_i, it groups the label-annotator pairs into three groups by labels values. Then it computes the mean vector of worker embedding vectors in each group and concatenates these mean vectors together. For example, the red vector in the bottom of Figure 1 corresponds to the mean vector of annotators who answer "$\prec$". Zero vector is used when a group does not contain any annotators.

Meanwhile, the reliability network utilizes $\mathrm{P}_{\mathrm{BT}}(\eta_1 \succ \eta_2)$ to address the variance in labels. The present study claims that label reliability also depends on the difficulty of queries. A label $y_i =$ "$\approx$" is less reliable if η_1 is significantly better than η_2, when compared to the case in which η_1 is only slightly better than η_2. Thus the reliability network utilizes $\mathrm{P}_{\mathrm{BT}}(\eta_1 \succ \eta_2)$ in determining the reliability of y_i. As the R summarizes information from all annotators, utilizing $\mathrm{P}_{\mathrm{BT}}(\eta_1 \succ \eta_2)$ effectively *smooths* the labels collected for each queries.

4.3 Learning

The parameter θ_R, θ_W, and θ_α can be learned by minimizing the following objective function:

$$L_{\mathrm{DCBT}}(\theta_R, \theta_W, \theta_\alpha) = -\frac{1}{N} \sum_{i=1}^{N} \left[\tilde{y_i} \log(\mathrm{P}_{\mathrm{DCBT}}) + (1 - \tilde{y_i}) \log(1 - \mathrm{P}_{\mathrm{DCBT}})\right], \quad (6)$$

where $\mathrm{P}_{\mathrm{DCBT}}$ is a short-hand notation for $\mathrm{P}_{\mathrm{DCBT}}(y_i =$ "$\succ$"). $\tilde{y_i}$ equals to 1 if $y_i =$ "$\succ$", 0.5 if $y_i =$ "$\approx$" and 0 if $y_i =$ "$\prec$". Note that $\mathrm{P}_{\mathrm{DCBT}}$ is a function of

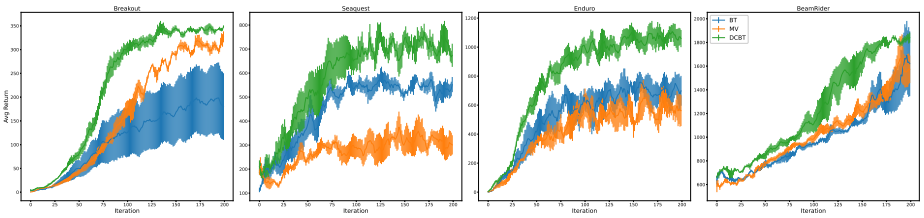

Fig. 3. Performance comparison among the proposed DCBT model, BT model, and MV. The curves are for the average returns obtained during training a QR-DQN agent. The QR-DQN agents trained with the proposed method performs better than those trained with BT and MV for game *Seaquest* and *Enduro*. Moreover, for all of the games, the proposed DCBT model enables faster policy learning convergence, which is also desirable in practice.

θ_R, θ_W and θ_α for a fixed dataset D. This dependency is omitted in notations for simplicity.

As mentioned by Chen et al. [2013], there is an identifiability issue for learning from pairwise comparisons. For a query $(\eta_{i,1}, \eta_{i,2})$, $\mathrm{P}_{\mathrm{BT}}(\eta_{i,1} \succ \eta_{i,2})$ does not change when adding an arbitrary constant $C \in \mathbb{R}$ to $G(\eta_{i,1})$ and $G(\eta_{i,2})$. Preliminary experiments show that reward network tends to output large values, a similar situation with the over-fitting problem in supervised learning. Following Chen et al. [2013], our learning algorithm utilizes a regularization term L_{reg} to restrict reward values around zero. Moreover, ℓ_1 and ℓ_2 regularization are also helpful to reduce over-fitting.

$$L_{\mathrm{reg}}(\theta_R) = -\frac{1}{2N}\sum_{i=1}^{N}\sum_{k=1}^{2}\left[\log\left(\frac{\exp\left(G(\eta_{i,k})\right)}{\exp\left(G(\eta_{i,k})\right)+1}\right)+\log\left(\frac{1}{\exp\left(G(\eta_{i,k})\right)+1}\right)\right]. \tag{7}$$

The overall objective function can be compactly written as

$$L(\theta_R, \theta_W, \theta_\alpha) = L_{\mathrm{DCBT}} + \lambda_1 L_{\mathrm{reg}} + \lambda_2 L_{\ell_1,\ell_2}, \tag{8}$$

Besides, an initialization phase is required before optimizing Equation 8. This phases initializes R by maximizing the loglikelihood of the BT model. The intuition is that by regarding all labels as correct, the reward network can attain intermediate ability in modeling preferences. It also initializes θ_α and θ_W by minimizing the cross entropy between α_i and a Bernoulli distribution with parameter $\bar{\alpha}$. This follows the initialization procedure for the Crowd-BT model. Formally, the objective function for initialization phase, $L_{\mathrm{DCBT-INIT}}$, is defined as:

$$L_{\mathrm{DCBT-INIT}}(\theta_R, \theta_W, \theta_\alpha) = -\frac{1}{N}\sum_{i}^{N}[\tilde{y_i}\log \mathrm{P}_{\mathrm{BT}} + (1-\tilde{y_i})\log(1-\mathrm{P}_{\mathrm{BT}}) + \bar{\alpha}\log(\alpha_i) + (1-\bar{\alpha})\log(1-\alpha_i)], \tag{9}$$

where P_{BT} is a short-hand notation for $\mathrm{P}_{\mathrm{BT}}(y_i = \text{“}\succ\text{"})$. Note that it is a function of θ_R, and α_i is a function of θ_R, θ_W and θ_α. $\bar{\alpha}$ is a hyper-parameter, which is set to 0.99. The gradients from the latter two terms of $L_{\mathrm{DCBT-INIT}}$ to θ_R are blocked, as they use pseudo labels instead of true labels.

After initialization, the objective function in Equation 8 is minimized with an alternative scheme, similar to the method described by Chen et al. [2013]. At each round, θ_R is first optimized for several gradient steps while keeping θ_W and θ_α fixed. Then, θ_W and θ_α are optimized for several steps while θ_R is fixed. The entire algorithm is given in Algorithm 1.

5 Evaluation

5.1 Setup

The present study utilizes the benchmark datasets published by Agarwal et al. [2020] for experimental evaluations; they contain trajectories for Atari 2600 games collected during training a DQN agent. Due to limited computation resources, four of the games used in existing work for online PbRL [Ibarz et al., 2018] are selected. In practice, trajectories are too long to be processed, so they are truncated to clips of length 30 ($T_c = 30$). From each game 50,000 clips of trajectories are randomly sampled. Queries are sampled randomly from these clips.

To have precise control for error rates, preferences are generated by 2,500 simulated annotators. Each annotator generates a correct preference label with a fixed probability sampled from $\mathrm{Beta}(7, 3)$. When making mistakes, an annotator selects one of the two incorrect labels uniformly at random. Each annotator labels at most 20 queries, which means the number of preferences does not exceed 50, 000. In our experiments, each query is annotated by at most ten different annotators.

Algorithms for reward learning are evaluated using the performance of the same policy policy-learning algorithm. Reward functions are learned using different reward learning algorithms on the same set of noisy preferences. Then, the reward functions are utilized to compute rewards for learning policies using the same policy-learning algorithm, which means that the performance of the policies reflects the performance of the reward-learning algorithms. For the policy learning algorithm, the quantile-regression DQN algorithm (QR-DQN) [Dabney et al., 2018] is adopted due to its superior performance. A recent empirical analysis shows that this algorithm yields the state-of-the-art performance in batch RL settings [Agarwal et al., 2020]. Obtained policies are evaluated in terms of the average return obtained per episode. Experiments are repeated three times on three different sets of trajectories of a game. The mean values of returns and their standard error are reported.

5.2 Implementation Details

Figure 2 shows the network structure used for reward learning using the DCBT model. The other reward learning algorithms utilize the same reward network.

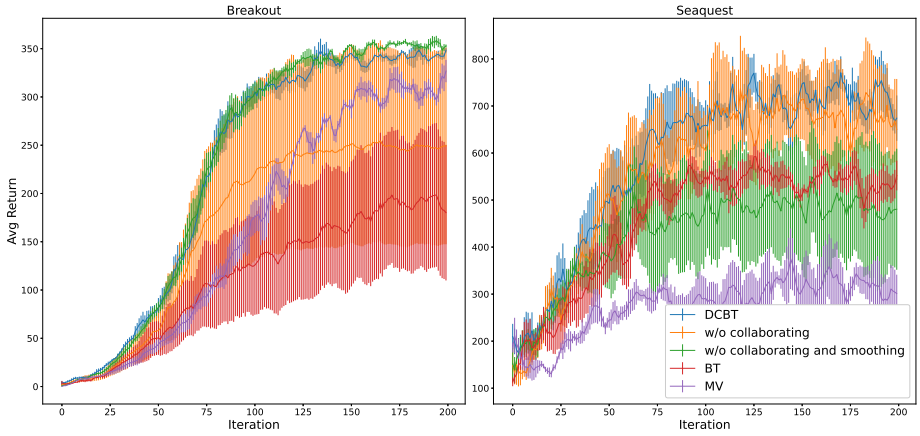

Fig. 4. The results of the ablation study for DCBT. For *Breakout*, removing annotator collaborating decreases performance, but further removing smoothing improves performance. For *Seaquest*, removing collaborating results in little difference, but further removing smoothing decreases performance. These results show that combining the two ideas can effectively handle noise in preferences and overcome drawbacks of the two ideas.

The convolutional network part of this architecture is the same as that of the QR-DQN agent released by [Agarwal et al., 2020].

The QR-DQN agent is trained for 200 training iteration. In each training iteration, to speed up training, the agent is continuously trained for 62,500 **gradient steps**. In its original implementation [Agarwal et al., 2020], agents are trained for 250,000 **environment steps**, during which parameters are updated every four environment steps. Therefore, in our evaluations for each iteration, agents are trained for the same number of gradient steps as [Agarwal et al., 2020] performed. Except for this difference, the other hyper-parameters are not altered.

5.3 Alternative Methods

For all of the four games, the proposed DCBT model is compared with the following two baseline methods.

BT Model. This method regards all preferences as correct ones, and utilizes the BT model to learn $\hat{R}$. It is the method used in the recent work for online PbRL setting [Christiano et al., 2017, Ibarz et al., 2018].

Majority Voting (MV). This method counts the occurrence of different labels for the same query. The label with the maximum count is chosen as the estimated label. Ties are broken randomly. Using the estimated labels, $\hat{R}$ is learned with

the BT model. The estimated labels generated by this method still contain noise, but they are less noisy than the original labels used by the BT method.

An ablation study is carried out to analyze the effect of annotator collaborating and smoothing in modeling label reliability. For game *Breakout* and *Seaquest*, the DCBT model is compared with the following two methods.

w/o collaborating. This is a variant of the proposed method that ignores other annotators who also label the same query.

w/o collaborating and smoothing. Not only *w/o collaborating*, the estimated rewards are also ignored. Note that this method only considers the identity of the annotator of a query, so it is equivalent to the Crowd-BT model.

5.4 Results

Figure 3 shows the results for all the four games. For the two games, *Seaquest* and *Enduro*, the proposed DCBT achieves the best final performance, which means that DCBT successfully generates reward functions that align with the tasks of interest. Meanwhile, for the other two games, *Breakout* and *BeamRider*, using MV results in slower convergence but close final performance. Thus, for these two games, the MV method can generate reward functions aligned with the tasks of interest, but such reward functions hinder fast convergence. Only for *BeamRider*, the BT method has a similar final performance as the DCBT model. In addition to improved final performance, faster convergence is also a desirable property in practice, especially in scenarios with limited resources. From this perspective, the proposed DCBT model also outperforms its alternatives.

Figure 4 shows the ablation study results for DCBT on *Breakout* and *Seaquest*. For *Breakout*, removing the annotator collaborating (shown in orange) decreases its performance, although it is still better than the BT method (shown in red). Hence for this game, label collaborating plays an important roll for DCBT. Interestingly, further removing the smoothing (shown in green) boosts the performance, which is even slightly better than the DCBT model. Since this is a rather simple model compared to DCBT, the increase in performance might be due to less overfitting.

For *Seaquest*, removing annotator collaborating hardly affects the performance. Furthermore, removing smoothing significantly decreases the performance. So for this game, the idea of smoothing might be important for the performance of the proposed DCBT model.

Our experimental results show that, the efficacy of annotator collaborating and smoothing might be task specific, but their drawbacks can be overcome by combining them together. Further investigation of the reasons is left as future work.

6 Conclusion

This paper address the lack of reward function in batch RL setting. Existing settings for this problem rely on optimal demonstrations provided by humans,

which is unrealistic for complex tasks. So even though data acquisition is scaled up with crowdsourcing, effective policy learning is still challenging. This paper tackles this problem by learning reward functions from noisy preferences. Generating preferences requires less expertise than generating demonstrations. Thus they can be solicited from vastly available non-expert humans. A critical challenge lies in the noise of preferences, which is overlooked in the literature of PbRL. This challenge is addressed with a novel probabilistic model called DCBT. DCBT collaboratively models the correlation between label reliability and annotators. It also utilizes the estimated reward function to compute preference estimates, which effectively smooths labels reliability. Evaluations on Atari 2600 games show the efficacy of the proposed model in learning reward functions from noisy preferences, followed by an ablation study for annotator collaborating and smoothing. Overall, this paper explores a novel methodology for harvesting human knowledge to learn policies in batch RL setting.

Our ablation study indicates the occurrence of over-fitting, which means the reward model might overly fit states and actions in preferences. How to inject induction bias to overcome such an effect is an interesting future work. Moreover, while there are 50 million states and actions for each game, only 50,000 are covered in queries. Under a fixed labelling budget, how to more effectively generate queries is an important issue.

Acknowledgment

This work was partially supported by JST CREST Grant Number JPMJCR21D1.

Bibliography

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 104–114. PMLR, 2020.

R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 00063444.

X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 193–202, 2013.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4302–4310. Curran Associates, Inc., 2017.

W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 2892–2901, 2018.

S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau. Benchmarking batch deep reinforcement learning algorithms, 2019.

C. Gelada and M. G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 3647–3655, 2019.

B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems 31*, page 8022–8034. Curran Associates Inc., 2018.

A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems 32*, 2019.

S. Lange, T. Gabel, and M. Riedmiller. *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, 2012.

A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Proceedings of the Second Conference on Robot Learning (CoRL)*, pages 879–893, 2018.

A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and F. Li. Scaling robot supervision to hundreds of hours with

roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1048–1055, 2019.

A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021.

B. Pavse, I. Durugkar, J. Hanna, and P. Stone. Reducing sampling error in batch temporal difference learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 7543–7552, 2020.

F. Rodrigues and F. Pereira. Deep learning from crowds. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

F. Sasaki and R. Yamashina. Behavioral cloning from noisy demonstrations. In *Proceeding of the International Conference on Learning Representations (ICLR)*, 2021.

S. Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems 9*, 1996.

J. W. Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(1):7026–7071, 2017. ISSN 1532-4435.

C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama. Imitation learning from imperfect demonstration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6818–6827, 2019.

Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.