Imitating Past Successes can be Very Suboptimal

Anonymous Author(s)
Affiliation
Address
email

Abstract

Prior work has proposed a simple strategy for reinforcement learning (RL): label experience with the outcomes achieved in that experience, and then imitate the relabeled experience. These outcome-conditioned imitation learning methods are appealing because of their simplicity, strong performance, and close ties with supervised learning. However, it remains unclear how these methods relate to the standard RL objective, reward maximization. In this paper, we prove that existing outcome-conditioned imitation learning methods do not necessarily improve the policy. However, we show that a simple modification results in a method that does guarantee policy improvement. Our aim is not to develop an entirely new method, but rather to explain how a variant of outcome-conditioned imitation learning can be used to maximize rewards.

1 Introduction

Recent work has proposed methods for reducing reinforcement learning (RL) to a supervised learning problem using hindsight relabeling [1, 10]. These approaches label each state action pair with an *outcome* that happened in the future (e.g., reaching a goal), and then learn a conditional policy by treating that action as optimal for making that outcome happen [4, 8, 25]. While most prior work defines outcomes as reaching goal states [4, 8, 20, 23, 25, 30], some work defines outcomes in terms of rewards [2, 12, 24], language [16], or sets of reward functions [6, 14]. We will refer to these methods as *outcome-conditioned behavioral cloning* (OCBC).

OCBC methods are appealing because of their simplicity and strong performance [2, 5, 8]. Their implementation mirrors standard supervised learning problems. These methods do not require learning a value function and can be implemented without any reference to a reward function. Thus, OCBC methods might be preferred over more standard RL algorithms, which are typically challenging to implement correctly and tune [9].

However, these OCBC methods face a major challenge: it remains unclear whether the learned policy is actually optimizing any control objective. Does OCBC correspond to maximizing some reward function? If not, can it be mended such that it does perform reward maximization? Understanding the theoretical underpinnings of OCBC is important for determining how to correctly apply this seemingly-appealing class of methods. It is important for diagnosing why existing implementations might fail to solve some tasks, and is important for predicting when these methods will work effectively. Relating OCBC to reward maximization may provide guidance on how to choose tasks and relabeling strategies to maximize a desired reward function.

The aim of this paper is to understand how OCBC methods relate to reward maximization. The key idea in our analysis is to decompose OCBC methods into a two-step procedure: the first step corresponds to averaging together experience collected when attempting many tasks, while the second step reweights the combined experience by task-specific reward functions. This averaging step is equivalent to taking a convex combination of the initial task-conditioned policies, and this averaging

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

can *decrease* performance on some tasks. The reweighting step, which is similar to EM-based policy search, corresponds to policy improvement. While EM-based policy search methods are guaranteed to converge, we prove that OCBC methods can fail to converge because they interleave an averaging step. Because this problem has to do with the underlying distributions, it cannot be solved by using more powerful function approximators or training on larger datasets.

The main contribution of our work is an explanation of how outcome-conditioned behavioral cloning relates to reward maximization. We show that outcome-conditioned behavioral cloning does not, in general, maximize performance. In fact, it can result in decreasing performance with successive iterations. However, by appropriately analyzing the conditional probabilities learned via outcome-conditioned behavioral cloning, we show that a simple modification results in a method with guaranteed improvement. Our analysis also provides practical guidance on applying OCBC methods (e.g., should every state be labeled as a success for some task?).

2 Preliminaries

We focus on a multi-task MDP with states s, actions a, initial state distribution $p_0(s_0)$, dynamics $p(s_{t+1} \mid s_t, a_t)$, tasks $e \sim p(e)$, and reward functions $r_e(s_t, a_t) \in [0, 1]$. We discuss the single task setting in Sec. 3.3. We use $\pi(a \mid s, e)$ to denote the policy for achieving outcome e. Let $\tau \triangleq (s_0, a_0, s_1, a_1, \cdots)$ be an infinite length trajectory. We overload notation and use $\pi(\tau \mid e)$ as the probability of sampling trajectory τ from policy $\pi(a \mid s, e)$. The objective and Q-values are:

$$\max_{\pi} \; \mathbb{E}_{p(e)}\left[\mathbb{E}_{\pi(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right]\right], \qquad Q^{\pi}(s, a, e) \triangleq \mathbb{E}_{\pi(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t) \middle| \begin{smallmatrix} s_0 = s \\ a_0 = a \end{smallmatrix}\right]. \tag{1}$$

Given a policy $\pi(a \mid s, e)$ and its corresponding Q-values Q^{π}, *policy improvement* is a procedure that produces a new policy $\pi'(a \mid s, e)$ with higher average Q-values:

$$\mathbb{E}_{\pi'(\tau|e)}\left[Q^{\pi}(s, a, e)\right] > \mathbb{E}_{\pi(\tau|e)}\left[Q^{\pi}(s, a, e)\right] \quad \text{for all states } s.$$

Policy improvement is most often implemented by creating a new policy that acts greedily with respect to the Q-function, but can also be implemented by reweighting the action probabilities of the current policy by (some function of) the Q-values [7, 18, 22]. Policy improvement is guaranteed to yield a policy that gets higher returns than the original policy, provided the original policy is not already optimal [26]. *Policy iteration* alternates between policy improvement and estimating the Q-values for the new policy, and is guaranteed to converge to the reward-maximizing policy [26].

2.1 Outcome-Conditioned Behavioral Cloning (OCBC)

Many prior works have used hindsight relabeling to reduce the multi-task RL problem to one of conditional imitation learning. Given a dataset of trajectories, these methods label each trajectory with a task that the trajectory completed, and then perform task-conditioned behavioral cloning [2, 4, 6, 8, 12, 14, 16, 20, 23–25, 30]. We call this method outcome-conditioned behavioral cloning (OCBC).

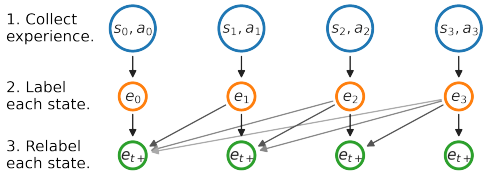

Figure 1: **Labeling and *re*labeling**: *(1)* OCBC collects experience, and then *(2)* labels every state with a task that was completed *at that state*. For example, in goal-reaching problems we have $e_t = s_t$, as state s_t completes the task of reaching goal s_t. *(3)* Finally, each state is *relabeled* with a task that was completed *in the future*. Each state may be relabeled with *multiple* tasks, as indicated by the extra green circles.

The input to OCBC is a behavioral policy $\beta(a \mid s)$, and the output is an outcome-conditioned policy $\pi(a \mid s, e)$. While the behavioral policy is typically left as an arbitrary choice, in our analysis we will construct this behavioral policy out of task-conditioned behavioral policies $\beta(a \mid s, e)$. This construction will allow us to say whether the new policy $\pi(a \mid s, e)$ is better than the old policy $\beta(a \mid s, e)$. We define $\beta(\tau)$ and $\beta(\tau \mid e)$ as the trajectory probabilities for behavioral policies $\beta(a \mid s)$ and $\beta(a \mid s, e)$, respectively.

The key step in OCBC is choosing which experience to use for learning each task, a step we visualize in Fig. 1. After collecting experience with the behavioral policy $\beta(a \mid s)$ (blue circles), *every* state s_t in this experience is labeled with an outcome e_t that is achieved at that state (orange circles). Most commonly, the outcome is equal to the state itself: $e_t = s_t$ [4, 8, 15, 24], though prior work considers

other types of outcomes, such as natural language descriptions [16]. We define $p(e_t \mid s_t)$ as the distribution of outcomes achieved at state s_t. Finally, we define a random variable e_{t+} to represent the future outcomes for state s_t and action a_t. The future outcome is defined as the outcomes for states that occur in the γ-discounted future. Formally, we can define this distribution over future outcomes as:

$$p^{\beta(a|s)}(e_{t+} \mid s_t, a_t) \triangleq (1-\gamma)\mathbb{E}_{\beta(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t p(e_t = e_{t+} \mid s_t)\right]. \tag{2}$$

Outcome-conditioned behavioral cloning then performs conditional behavioral cloning on this relabeled experience. Using $p^{\beta}(s,a)$ to denote the distribution over states and actions visited by the behavioral policy and $p^{\beta}(e_{t+} \mid s, a)$ to denote the distribution over future outcomes (Eq. 2), we can write the conditional behavioral cloning objective as

$$\mathcal{F}(\pi;\beta) = \mathbb{E}_{\substack{p^{\beta}(e_{t+}|s_t,a_t)\\ p^{\beta}(s_t,a_t)}}\left[\log \pi(a_t \mid s_t, e_{t+})\right]. \tag{3}$$

Algorithm 1 Outcome-conditioned behavioral cloning (OCBC)

input behavior policy $\beta(a \mid s)$
$\mathcal{D} \leftarrow \textsc{Relabel}(\tau)$ where $\tau \sim \beta(\tau)$ ▷ See Fig. 1.
$\mathcal{F}(\pi;\beta) \leftarrow \mathbb{E}_{(s_t,a_t,e_{t+})\sim\mathcal{D}}\left[\log \pi(a_t \mid s_t, e_{t+})\right]$
$\pi(a \mid s, e) \leftarrow \arg\max_{\pi} \mathcal{F}(\pi)$
return $\pi(a \mid s, e)$

We define $\pi_O(a \mid s, e) \triangleq \arg\max_{\pi} \mathcal{F}(\pi;\beta)$ as the solution to this optimization problem. We summarize the OCBC training procedure in Alg. 1.

3 Relating OCBC to Reward Maximization

In this section, we show that the conventional implementation of OCBC does not quite perform reward maximization. This analysis identifies why OCBC might perform worse, and provides guidance on how to choose tasks and perform relabeling so that OCBC methods perform better. The analysis also allows us to draw an equivalence with prior EM-based policy search methods.

3.1 What are OCBC methods *trying* to do?

The OCBC objective (Eq. 3) corresponds to a prediction objective, but we would like to solve a control objective: maximize the probability of achieving outcome e. We can write this objective as the likelihood of the desired outcome under the future outcome distribution (Eq. 2) following the task-conditioned policy $\pi(a \mid s, e)$:

$$\mathbb{E}_{e^*\sim p(e)}\left[p^{\pi(\cdot|\cdot,e^*)}(e_{t+} = e^*)\right] = \mathbb{E}_{e^*\sim p(e),\pi(\tau|e=e^*)}\left[\sum_{t=0}^{\infty}\gamma^t \underbrace{(1-\gamma)p(e_t = e^* \mid s_t)}_{\triangleq r_{e^*}(s_t,a_t)}\right]$$

By substituting the definition of the future outcome distribution (Eq. 2) above, we see that this control objective is equivalent to a standard multi-task RL problem (Eq. 1) with the reward function: $r_e(s_t, a_t) \triangleq (1-\gamma)p(e_t = e \mid s_t)$. *Do OCBC methods succeed in maximizing this RL objective?*

3.2 How does OCBC relate to reward maximization?

While it is unclear whether OCBC methods maximize rewards, the OCBC objective is closely related to reward maximizing because the distribution over future outcomes looks like a value function:

Proposition 3.1. *The distribution of future outcomes (Eq. 2) is equivalent to a Q-function for the reward function $r_e(s_t)$:*

$$p^{\beta(a|s)}(e_{t+} = e \mid s_t, a_t) = (1-\gamma)\mathbb{E}_{\beta(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t p(e_t = e_{t+} \mid s_t)\right] = Q^{\beta(a|s)}(s_t, a_t; e).$$

This result hints that OCBC methods are close to performing reward maximization. To clarify this connection, we use Bayes' rule to express the OCBC policy $\pi_O(a \mid s, e)$ in terms of this Q-function:

$$\begin{aligned}\pi_O(a_t \mid s_t, e_{t+}) &= p^{\beta(a|s)}(a_t \mid s_t, e_{t+})\\ &\propto p^{\beta(a|s)}(e_{t+} \mid s_t, a_t)\beta(a_t \mid s_t) = Q^{\beta(a|s)}(s_t, a_t; e = e_{t+})\beta(a_t \mid s_t).\end{aligned} \tag{4}$$

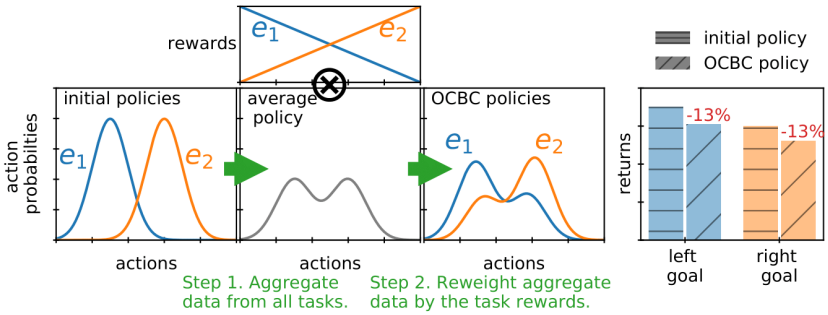

Figure 2: **How does OCBC relate to reward maximization?** OCBC is equivalent to averaging the policies from different tasks *(Step 1)* and then reweighting the action distribution by Q-values *(Step 2)*. The reweighting step is policy improvement, but the averaging step can decrease performance. In this example, OCBC *decreases* the task reward by 13%.

This expression provides a simple explanation for what OCBC methods are doing – they take the behavioral policy's action distribution and reweight it by the Q-values, increasing the likelihood of good actions and decreasing the likelihood of bad actions, as done in prior EM-based policy search methods [18, 21, 22]. Such reweighting is guaranteed to perform policy improvement:

Proposition 3.2. *Let $\pi(a \mid s, e)$ be the Bayes-optimal policy for applying OCBC to behavioral policy $\beta(a \mid s)$. Then $\pi(a \mid s, e)$ achieves higher returns than $\beta(a \mid s)$ under reward function $r_e(s_t, a_t)$:*

$$\mathbb{E}_{\pi(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right] \geq \mathbb{E}_{\beta(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right].$$

The proof is in Appendix B. This initial result is an apples-to-oranges comparison because it compares a task-conditioned policy $\pi(a \mid s, e)$ to a task-agnostic policy, $\beta(a \mid s)$.

To fully understand whether OCBC corresponds to policy improvement, we have to compare successive iterates of OCBC. To do this, assume that data is collected from a task-conditioned behavioral policy $\beta(a \mid s, e)$, where tasks are sampled from the prior $e \sim p(e)$.[1] We can now rewrite the Bayes-optimal OCBC policy (Eq. 4) as the output of a two-step procedure: average together the policies for different tasks, and then perform policy improvement:

$$\underbrace{\beta(a \mid s) \leftarrow \int \beta(a_t \mid s_t, e) p^{\beta}(e \mid s_t) de}_{\text{averaging step}} \quad \longleftrightarrow \quad \underbrace{\pi(a \mid s, e) \leftarrow \frac{Q^{\beta}(s, a, e)}{p^{\beta}(e \mid s)} \beta(a \mid s)}_{\text{improvement step}}$$

The distribution $p^{\beta}(e \mid s_t)$ is the distribution over *commanded* tasks, given that the agent is in state s_t. This two-step decomposition is useful because it allows us to answer an important question: *does OCBC actually learn policies that maximize each of the task rewards?*

To answer this question, we visualize this two-step decomposition of OCBC in Fig. 2 for a setting with two tasks, one state, and a one-dimensional action distribution. We assume the two tasks have equal probability, and other details are included in the supplemental materials. The averaging step takes the initial task-conditioned policies *(left)* and combines them into a single policy *(center-bottom)*. Since there is only one state, the weights $p(e \mid s)$ are equal for both tasks. The improvement step reweights the average distribution by the task rewards *(center-top)*, resulting in the final task-conditioned policies learned by OCBC *(right)*. In this example, the final task-conditioned policies achieve returns that are about 13% *worse* than the initial policies.

Formally, this counterexample provides us with a proof of three negative results:

Proposition 3.3. *There exists an environment, task distribution, and task e where the OCBC policy, $\pi(a \mid s, e)$, does not achieve the highest rewards:*

$$\mathbb{E}_{\pi(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right] < \max_{\pi^*} \mathbb{E}_{\pi^*(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right].$$

[1] The state-action distribution $p^{\beta}(s, a)$ and future outcome distribution $p^{\beta}(e_{t+} \mid s, a)$ for this *mixture* of policies is equivalent to those for a Markovian policy, $\beta(a_t \mid s_t) = \int \beta(a_t \mid s_t, e) p^{\beta}(e \mid s_t) de$ [31, Thm. 2.8].

Proposition 3.4. *There exists an environment, policy $\beta(a \mid s, e)$, and task e such that the policy produced by OCBC, $\pi(a \mid s, e)$, achieves lower returns than the behavior policy:*

$$\mathbb{E}_{\pi(\tau|e)}\left[\sum_{t=0}^{\infty}\gamma^t r_e(s_t, a_t)\right] < \mathbb{E}_{\beta(\tau|e)}\left[\sum_{t=0}^{\infty}\gamma^t r_e(s_t, a_t)\right].$$

Not only can OCBC fail to find the optimal policy, it can yield a policy that is worse than the initial behavior policy. Moreover, iterating OCBC can yield policies that get worse and worse:

Proposition 3.5. *There exists an environment, an initial policy $\pi_0(a \mid s, e)$, and task e such that iteratively applying OCBC, $\pi_{t+1} \leftarrow \arg\max_\pi \mathcal{F}(\pi; \beta = \pi_t)$, results in* decreasing *the probability of achieving desired outcomes:*

$$\mathbb{E}_{\pi_{t+1}(\tau|e)}\left[\sum_{t=0}^{\infty}\gamma^t r_e(s_t, a_t)\right] < \mathbb{E}_{\pi_t(\tau|e)}\left[\sum_{t=0}^{\infty}\gamma^t r_e(s_t, a_t)\right] \quad \textit{for all } t = 0, 1, \cdots.$$

While prior work has claimed that OCBC is an effective way to learn from suboptimal data [4, 8], this result suggests that this claim is not always true. Moreover, while prior work has argued that *iterating* OCBC is necessary for good performance [8], our analysis suggests that such iteration can actually lead to *worse* performance than doing no iteration.

In this example, the reason OCBC made the policies worse was because of the averaging step. In general, the averaging step may make the policies better or worse. If all the tasks are similar (e.g., correspond to taking similar actions), then the averaging step can decrease the variance in the estimates of the optimal actions. If the tasks visit disjoint sets of states (e.g., the trajectories do not overlap), then the averaging step will have no effect because the averaging coefficients $p^\beta(e \mid s)$ are zero for all but one task. If the tasks have different optimal actions but do visit similar states, then the averaging step can cause performance to decrease. This case is more likely to occur in settings with wide initial state distributions, wide task distributions, and stochastic dynamics. In Sec. 4, we will use this analysis to propose a variant of OCBC that does guarantee policy improvement.

3.3 Relationship with EM policy search

While OCBC is conventionally presented as a way to solve control problems without explicit RL [4, 8], we can use the analysis in the preceding sections to show that OCBC has a close connection with prior RL algorithms based on expectation maximization [3, 11, 13, 17, 18, 21, 22, 27, 28]. We refer to these methods collectively as *EM policy search*. These methods typically perform some sort of reward-weighted behavioral cloning [3, 22], with each iteration corresponding to a policy update of the following form:

$$\max_\pi \mathbb{E}_{p^\beta(s_t, a_t)}\left[f(Q^\beta(s_t, a_t)) \log \pi(a_t \mid s_t)\right],$$

where $f(\cdot)$ is a positive transformation. While most prior work uses an exponential transformation ($f(Q) = e^Q$) [13, 18, 28], we will use a linear transformation ($f(Q) = Q$) [3, 22] to draw a precise connection with OCBC. Prior methods use a variety of techniques to estimate the Q-values, with Monte-Carlo estimates being a common choice [22].

OCBC is EM policy search. We now show that each step of EM policy search is equivalent to each step of OCBC, for a particular set of tasks $\mathcal{E}$ and a particular distribution over tasks $p(e)$. To reduce EM policy search to OCBC, will use two tasks: task e_1 corresponds to reward maximization while task e_0 corresponds to reward minimization. Given a reward function $r(s) \in (0, 1)$, we annotate each state with a task by sampling

$$p(e \mid s) = \begin{cases} e_1 & \text{with probability } r(s) \\ e_0 & \text{with probability } (1 - r(s)) \end{cases}.$$

With this construction, the relabeling distribution is equal to the Q-function for reward function $r(s)$: $p^\beta(e_{t+} = e_1 \mid s_t, a_t) = Q^\beta(s_t, a_t)$. We assume that data collection is performed by deterministically sampling the reward maximization task, e_1. Under this choice of tasks, the objective for OCBC

(Eq. 3) is exactly equivalent to the objective for reward weighted regression [22]:

$$\begin{aligned}\mathcal{F}(\pi;\beta) &= \mathbb{E}_{p^\beta(s_t,a_t)p^\beta(e_{t+}|s_t,a_t)}\left[\log \pi(a_t \mid s_t, e_{t+})\right] \\ &= \mathbb{E}_{p^\beta(s_t,a_t)}\left[p(e_{t+}=e_1 \mid s_t,a_t)\log\pi(a_t \mid s_t,e_1) + p(e_{t+}=e_0 \mid s_t,a_t)\log\pi(a_t \mid s_t,e_0)\right] \\ &= \mathbb{E}_{p^\beta(s_t,a_t)}\left[Q^\beta(s_t,a_t)\log\pi(a_t \mid s_t,e_1) + \cancel{(1-Q^\beta(s_t,a_t))\log\pi(a_t \mid s_t,e_0)}\right].\end{aligned}$$

On the last line, we have treated the loss for the reward minimization policy (task e_0) as a constant, as EM policy search methods do not learn this policy.

While OCBC can fail to optimize this objective (Sec. 3.2), EM-based policy search methods are guaranteed to optimize expected returns (Dayan and Hinton [3], Toussaint et al. [29, Lemma 1]). This apparent inconsistency is resolved by noting how data are collected. Recall from Sec. 3.2 and Fig. 2 that OCBC is equivalent to an averaging step followed by a policy improvement step. The averaging step emerges because we collect experience for each of the tasks, and then aggregate the experience together. Therein lies the difference: EM policy search methods only collect experience when commanding one task, $\pi(a \mid s, e = e_1)$. Conditioning on a single outcome (e_1) during data collection removes the averaging step, leaving just the policy improvement step.

We can apply a similar trick to any OCBC problem: if there is a single outcome that the user cares about (e^*), alternate between optimizing the OCBC objective and collecting new experience conditioned on outcome e^*. If all updates are computed exactly, such a procedure is guaranteed to yield the optimal policy for outcome e^*, maximizing the reward function $r(s) = p(e = e^* \mid s)$.

This trick only works if the user has a single desired outcome. What if the user wants to learn optimal policies for *multiple tasks*? The following section shows how OCBC can be modified to guarantee policy improvement for learning multiple tasks.

3.4 Should you relabel with failed outcomes?

While our results show that OCBC may or may not yield a better policy, they also suggest that the choice of tasks $\mathcal{E}$ and the distribution of commanded tasks $p(e)$ affect the performance of OCBC. In Appendix A, we discuss whether every state should be labeled as a success for some task, or whether some tasks should be treated as a failure for all tasks. The main result relates this decision to a bias-variance tradeoff: labeling every state as a success for some task decreases variance but potentially incurs bias. Our experiment shows that relabeling some tasks as failures is harmful in the low-data setting, but helpful in settings with large amounts of data.

4 Fixing OCBC

OCBC is not guaranteed to produce optimal (reward-maximizing) policies because of the averaging step. Thus, fixing OCBC requires performing the improvement step without the averaging step. Our approach "undoes" the effect of the averaging step by normalizing the policy learned by OCBC.

4.1 Normalized OCBC

As discussed in Sec. 3.2, OCBC is taking the *task-agnostic* policy and reweighting it by the Q-values. The key issue was that improving the policy for some tasks might make the task-agnostic policy worse for other tasks. So, instead, we propose a *normalized OCBC* that takes the *task-conditioned* policy and reweights it by the Q-values:

$$\pi(a \mid s, e) \propto Q^\beta(s, a; e)\beta(a \mid s, e).$$

This update is simple to implement by applying a small change to OCBC. Let the task-conditioned behavior policy $\beta(a \mid s, e)$ be given. In addition to learning the OCBC policy, $\pi(a \mid s, e)$, normalized OCBC also estimates the behavioral policy: $\pi_N(a \mid s) \approx \beta(a \mid s)$. Doing so will allow us to "cancel out" the averaging step typically done by OCBC.

Normalized OCBC estimates the behavioral policy using task-agnostic behavioral cloning:

$$\max_{\pi_N(a|s)} \mathbb{E}_{e\sim p(e),\tau\sim\beta(\tau|e),(s_t,a_t)\sim\tau}\left[\log \pi_N(a_t \mid s_t)\right].$$

Once we have learned these two policies, we take actions by sampling from the *ratio* of the policies:

$$\tilde{\pi}(a \mid s, e) \leftarrow \frac{\pi(a \mid s, e)\beta(a \mid s, e)}{\pi_N(a \mid s)} \approx \frac{Q^\beta(s, a; e)\cancel{\beta(a \mid s)}}{\int Q^\beta(s, a'; e)\beta(a' \mid s)da'} \frac{\beta(a \mid s, e)}{\cancel{\beta(a \mid s)}} \propto Q^\beta(s, a; e)\beta(a \mid s, e).$$

The approximation above reflects errors in estimating the OCBC policy and the task-agnostic behavior policy.In settings with discrete actions, this ratio can be computed by taking the vector of action probabilities from the three policies ($\tilde{\pi}(\cdot \mid s, e), \beta(\cdot \mid s, e), \pi_N(\cdot \mid s)$), computing an elementwise product, and then normalizing the vector to sum to 1. In settings with continuous actions, the ratio can be estimated using importance sampling, as shown in Alg. 2: sample many actions from the task-conditioned policy, compute an importance weight for each action, and then sample one of these actions. Normalized OCBC performs policy improvement:

Algorithm 2 Normalized OCBC

function RATIOPOLICY($s_t, e_{t+}; \pi(a \mid s, e), \pi_N(a \mid s)$)
  $a^{(1)}, a^{(2)}, \cdots \sim \pi(a_t \mid s_t, e_{t+})$
  $w \leftarrow \left[\frac{1}{\pi_N(a^{(1)}|s_t)}, \frac{1}{\pi_N(a^{(2)}|s_t)}, \cdots\right]$
  $i \sim \text{CATEGORICAL}\left(\frac{w}{\|w\|_1}\right)$
  return $a^{(i)}$
function POLICYIMPROVEMENT($\pi(a \mid s, e)$)
  $\pi(a \mid s, e) \leftarrow \arg\max_\pi \mathbb{E}_{\substack{e\sim p(e), \tau\sim\pi(\tau|e) \\ (s_t,a_t)\sim\tau, k\sim\text{Geom}(1-\gamma)}} [\log \pi(a_t \mid s_t, e_{t+k})]$
  $\pi_N(a \mid s) \leftarrow \arg\max_{\pi_N} \mathbb{E}_{\substack{e\sim p(e), \tau\sim\pi(\tau|e) \\ (s_t,a_t)\sim\tau, k\sim\text{Geom}(1-\gamma)}} [\log \pi_N(a_t \mid s_t)]$
  return RATIOPOLICY($s_t, e_{t+}; \pi, \pi_N$)

Lemma 4.1. *Let a task-conditioned behavioral policy $\beta(a \mid s, e)$ be given, and let $\tilde{\pi}(a \mid s, e)$ be the output of normalized OCBC. If all policies are estimated exactly, then the new policy achieves a greater return than the task-conditioned behavioral policy:*

$$\mathbb{E}_{\tilde{\pi}(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right] \geq \mathbb{E}_{\beta(\tau|e)}\left[\sum_{t=0}^{\infty} \gamma^t r_e(s_t, a_t)\right] \quad \textit{for all tasks } e.$$

4.2 How to embed reward maximization problems into OCBC problems?

We assume that a set of reward functions is given: $\{r_e(s, a) | e \in \mathcal{E}\}$. Without loss of generality, we assume that all reward functions are positive. Let $r_{\text{max}} \triangleq \max_{s,a,e} r_e(s, a)$ be the maximum reward for any task, at any state and action. We then define an additional, "failure" task: $r_{\text{fail}}(s, a) = r_{\text{max}} - \sum_e r_e(s, a)$. We then use these reward functions to determine how to label each state for OCBC: $p(e_t \mid s_t, a_t) = \frac{1}{r_{\text{max}}} r_e(s_t, a_t) \quad$ for all $e \in \mathcal{E} \cup \{\text{fail}\}$.

5 Experiments

Our experiments aim at answering three questions. **First**, does OCBC fail to converge to the optimal policy in practice, as predicted by our theory? We test this claim on both a bandit problem and a simple 2D navigation problem. **Second**, on these same tasks, does our proposed fix to OCBC allow the method to converge to the optimal policy, as our theory predicts? Our aim in starting with simple problems is to show that OCBC can fail to converge, *even on extremely simple problems*. **Third**, do these issues with OCBC persist on a suite of more challenging, goal-conditioned RL tasks [8]? We also use these more challenging tasks to understand when existing OCBC methods work well, and when the normalized version is most important. We emphasize that the aim of this work is primarily to understand when OCBC methods can work and when our proposed normalization can boost performance. Our aim is not to present an entirely new method that achieves state-of-the-art results. Appendix C contains experimental details.

Bandit problem. Our first experiment looks at the learning dynamics of iterated OCBC on a bandit problem. We intentionally choose a simple problem to illustrate that OCBC methods can fail on even the simplest of problems. The bandit setting allows us to visualize the policy for each task as a distribution over actions. We consider a setting with three actions and three tasks, so the policy for each task can be represented as a point $(\pi(a = a_1 \mid e), \pi(a = a_2 \mid e))$, as shown in Fig. 3. We first visualize the averaging and improvement steps of OCBC. The averaging step *(left)* collapses the three task-conditioned policies into a single, task-agnostic policy. The improvement step *(center-left)* then produces task-conditioned policies by weighting the action distribution by the probability that

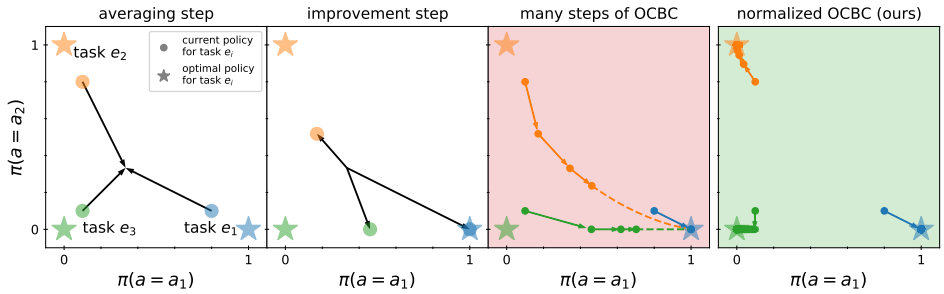

Figure 3: **OCBC = averaging + improvement.** *(Left)* On a bandit problem with three actions and three tasks, we plot the current and optimal policies for each task. OCBC first averages together the action distributions for each task and then *(Left-Center)* learns new policies by reweighting the action distribution by the Q-values for each task. *(Right-Center)* Iterating OCBC (averaging + improvement) produces the optimal policy for task e_1 but suboptimal policies for tasks e_2 and e_3; the final policies for tasks e_2 and e_3 are *worse* than the initial policies. *(Right)* Normalized OCBC does converge to the optimal policies for each task.

each action solves the task. The blue task (e_1) is only solved using action a_1, so the new policy for task e_1 exclusively takes action a_1. The green task (e_3) is solved by both action a_1 (with probability 0.3) and action a_3 (with probability 0.4), so the new policy samples one of these two actions (with higher probability for a_3). Iterating OCBC *(center-right)* produces the optimal policy for task e_1 but suboptimal policies for tasks e_2 and e_3; the final policies for tasks e_2 and e_3 are *worse* than the initial policies. In contrast, normalized OCBC *(right)* does converge to the optimal policies for each task. The fact that OCBC can fail on such an easy task suggests it may fail in other settings.

The bandit setting is a worst-case scenario for OCBC methods because the averaging step computes an *equally*-weighted average of the three task-conditioned policies. In general, different tasks visit different states, and so the averaging step will only average together policies that visit similar states.

2D navigation. Our next experiment tests whether OCBC can fail to converge on problems beyond bandit settings. As noted above, we expect the averaging step to have a smaller negative effect in non-bandit settings. To study this question, we use a simple 2D navigation task, shown in Appendix Fig. 4. This tabular navigation task has four actions, corresponding to the four adjacent states. The dynamics are stochastic, so the agent transitions to a random state with probability 50%. We collect data by sampling two goals: the blue goal is sampled with probability 90% while the orange goal is sampled with probability 10%.

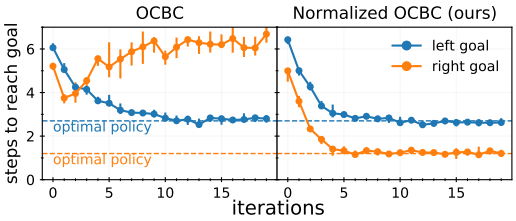

Figure 4: **OCBC $\neq$ policy improvement**: We use a tabular environment with two goals to compare OCBC and normalized OCBC (see Appendix Fig. 7). We plot the number of steps each method requires to reach each goal, when commanded to reach that goal. While both methods succeed at reaching the blue goal in near-optimal time, only our method learns a near-optimal policy for reaching the orange goal.

We compare OCBC to normalized OCBC, measuring the average number of steps to reach the commanded goal and plotting results in Fig. 4. At each iteration, we compute the OCBC update (Eq. 3) exactly. The blue line denotes the number of steps to reach the blue goal, when the agent was commanded to reach the blue goal. Likewise for the orange line. Both methods learn near-optimal policies for reaching the blue goal, but only our method learns a good policy for the orange goal.

OCBC's poor performance at reaching the orange goal can be explained by looking at the averaging step. Because the blue goal is commanded nine times more often than the orange goal, the averaging policy $p(a \mid s)$ is highly skewed towards actions that lead to the blue goal. Removing this averaging step, as done by normalized OCBC, results in learning the optimal policy.

Comparison on benchmark tasks. So far, our experiments have confirmed our theoretical predictions that OCBC *can* fail to converge; our analysis does not suggest that OCBC methods will *always* fail. Our second set of experiments study the performance of OCBC and normalized OCBC in higher-dimensional tasks. We aim to determine whether the theoretical issues with OCBC *ever* occur in these higher-dimensional tasks. We compare to a recent and prototypical implementation of OCBC, GCSL [8]. To give this baseline a strong footing, we use the goal-reaching benchmark proposed in that paper. Following Ghosh et al. [8], we report the *normalized* returns.

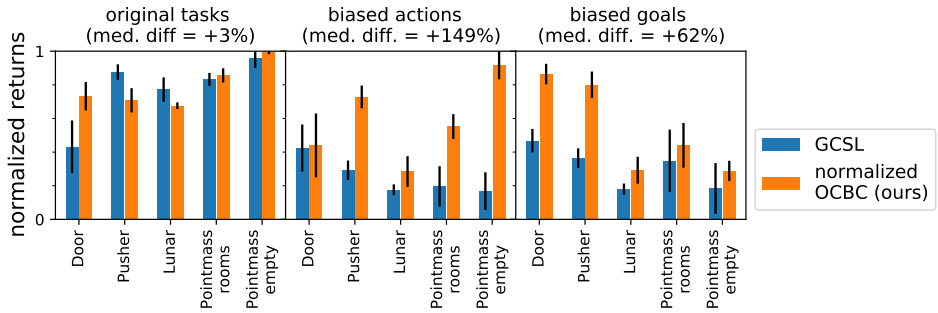

Figure 5: **Goal-conditioned RL benchmark**. We compared normalized OCBC to a prototypical implementation of OCBC (GCSL [8]), using the tasks proposed in that prior work. *(Left)* On the original tasks, both methods perform comparably. On versions of the tasks with *(Center)* skewed action distributions or *(Right)* goal distributions, normalized OCBC performs better, with a median improvement of $\sim 149\%$ and $\sim 62\%$.

We first compare OCBC to normalized OCBC on the original tasks. As shown in Fig. 5 *(Left)*, OCBC and normalized OCBC perform comparably on these tasks. The median improvement from normalized OCBC is a negligible $+3\%$. This result is in line with prior work that finds that OCBC methods can perform well [4, 8, 25], despite our theoretical results.

We hypothesize that OCBC methods will perform worst in settings where the averaging step has the largest effect; we predict that normalized OCBC will be especially important in such settings. We test this hypothesis by modifying the GCSL tasks so that the average policy achieves very low rewards for some tasks. We do this in two ways. For *biased actions*, we skew the initial data collection to only sample the optimal action (as determined by a pretrained policy) 5% of the time, sampling another action uniformly at random for the remaining 95%. This setting mimics the scenario where we want to learn policies from highly-suboptimal behavior. For *biased goals*, we skew the goal distribution so that the goal distribution is no longer centered around the environment's initial state distribution. This setting mimics the scenario where agents must learn to complete both easy and difficult goals, without overfitting to the easy goals.

We show results in Fig. 5. On both sets of imbalanced tasks, our normalized OCBC significantly outperforms OCBC. Normalized OCBC yields a median improvement of $+149\%$ and $+62\%$ for biased settings, supporting our theoretical predictions about the importance of normalization. These results also suggest that existing benchmarks may be accidentally easy for the prior methods because the data distributions are close to uniform, an attribute that may not transfer to real-world problems.

6 Conclusion

The aim of this paper is to analyze how outcome-conditioned behavioral cloning relates to training optimal reward-maximizing policies. While prior work has pitched these methods as an attractive alternative to standard RL algorithms, it has remained unclear whether these methods actually optimize any control objective. Understanding how these methods relate to reward maximization is important for predicting when they will work and for diagnosing their failures. Our main result is that standard OCBC methods do not perform policy improvement, and do not correspond to maximizing the probability of achieving the desired outcome or any well-defined reward in general. In fact, they can decrease the probability of success or the average reward. Based on our analysis, we propose a simple change that makes OCBC methods perform policy iteration. While the aim of this work was not to propose an entirely new method, simple experiments did show that these simple changes are important for guaranteeing convergence on simple problems.

One limitation of our analysis is that it does not address the effects of function approximation error and sampling error. As such, we showed that the problem with OCBC will persist even when using arbitrarily expressive and tabular policies trained on unlimited data. The practical performance of OCBC methods, and the normalized counterparts that we introduce in this paper, will be affected by these sorts of errors, and analyzing these is a promising direction for future work. A second limitation of our analysis is that it does not focus on how the set of tasks should be selected, and what distribution over tasks should be commanded. Despite these limitations, we believe our analysis may be useful for designing simple RL methods that are guaranteed to maximize returns.

References

[1] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058.

[2] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*.

[3] Dayan, P. and Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278.

[4] Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. (2019). Goal-conditioned imitation learning. *Advances in Neural Information Processing Systems*, 32:15324–15335.

[5] Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. (2021). Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*.

[6] Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. (2020). Rewriting history with inverse rl: Hindsight inference for policy improvement. In *Advances in Neural Information Processing Systems*.

[7] Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR.

[8] Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C. M., Eysenbach, B., and Levine, S. (2020). Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*.

[9] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[10] Kaelbling, L. P. (1993). Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer.

[11] Kober, J. and Peters, J. (2009). Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE.

[12] Kumar, A., Peng, X. B., and Levine, S. (2019). Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*.

[13] Levine, S. and Koltun, V. (2013). Variational policy search via trajectory optimization. *Advances in neural information processing systems*, 26:207–215.

[14] Li, A. C., Pinto, L., and Abbeel, P. (2020). Generalized hindsight for reinforcement learning. *arXiv preprint arXiv:2002.11708*.

[15] Lynch, C. and Sermanet, P. (2020). Grounding language in play. *arXiv preprint arXiv:2005.07648*.

[16] Lynch, C. and Sermanet, P. (2021). Language conditioned imitation learning over unstructured data. *Proceedings of Robotics: Science and Systems. doi*, 10.

[17] Neumann, G. et al. (2011). Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 817–824.

[18] Neumann, G., Peters, J., et al. (2009). Fitted q-iteration by advantage weighted regression. In *Advances in Neural Information Processing Systems 21-Proceedings of the 2008 Conference*, pages 1177–1184.

[19] Oh, J., Guo, Y., Singh, S., and Lee, H. (2018). Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR.

[20] Paster, K., McIlraith, S. A., and Ba, J. (2020). Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*.

[21] Peters, J., Mulling, K., and Altun, Y. (2010). Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[22] Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750.

[23] Savinov, N., Dosovitskiy, A., and Koltun, V. (2018). Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*.

[24] Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. (2019). Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*.

[25] Sun, H., Li, Z., Liu, X., Zhou, B., and Lin, D. (2019). Policy continuation with hindsight inverse dynamics. *Advances in Neural Information Processing Systems*, 32:10265–10275.

[26] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[27] Toussaint, M. (2009). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056.

[28] Toussaint, M., Harmeling, S., and Storkey, A. (2006). Probabilistic inference for solving (po) mdps. Technical report, Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh.

[29] Toussaint, M., Storkey, A., and Harmeling, S. (2010). Expectation-maximization methods for solving (po) mdps and optimal control problems. *Inference and Learning in Dynamic Models*.

[30] Yang, R., Fang, M., Han, L., Du, Y., Luo, F., and Li, X. (2021). Mher: Model-based hindsight experience replay. *ArXiv*, abs/2107.00306.

[31] Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The main claim of the paper is that existing outcome conditioned RL methods can learn policies that are suboptimal; they can get progressively worse. We include the proof in Apepndix B
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our contributions are theoretical in nature, and therefore do not have a direct effect on the potential societal applications of RL.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is attached in the supplemental
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix. C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars show the mean and standard deviation across random seeds. The number of random seeds is specified within each section.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The didactic experiments ran in a few minutes on a desktop CPU. The continuous control experiments ran for a few hours on a desktop GPU.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]