On the Theory of Reinforcement Learning with Once-per-Episode Feedback

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce a theory of reinforcement learning (RL) in which the learner receives feedback only once at the end of an episode. While this is an extreme test case for theory, it is also arguably more representative of real-world applications than the traditional requirement in RL practice that the learner receive feedback at every time step. Indeed, in many real-world applications of reinforcement learning, such as self-driving cars and robotics, it is easier to evaluate whether a learner's complete trajectory was either "good" or "bad," but harder to provide a reward signal at each step. To show that learning is possible in this more challenging setting, we study the case where trajectory labels are generated by an unknown parametric model, and provide a statistically and computationally efficient algorithm that achieves sub-linear regret.

1 Introduction

The Reinforcement Learning (RL) paradigm involves a learning agent interacting with an unknown dynamical environment over multiple time steps. The learner receives a reward signal after each step which it uses to improve its performance over time. This formulation of RL has had significant empirical success in the recent past [22, 21, 31, 30].

While this empirical success is encouraging, as RL starts to tackle a more wide-ranging class of consequential real-world problems, such as self-driving cars, supply chains, and medical care, a new set of challenges arise. Foremost among them is the lack of a well-specified reward signal associated with every state-action pair in many real-world settings. For example, consider a robot manipulation task where the robot must fold a pile of clothes. It is not clear how to design a useful reward signal that aids the robot to learn to complete this task. However, it is fairly easy to check whether the task was successfully completed (that is, whether the clothes were properly folded) and provide feedback at the end of the episode.

This is a classical challenge but it is one that is often neglected in theoretical treatments of RL. To address this challenge we introduce a framework for RL that eschews the need for a Markovian reward signal at every step and provides the learner only with binary feedback based on its complete trajectory in an episode. In our framework, the learner interacts with the environment for a fixed number of time steps (H) in each episode to produce a trajectory, τ, which is the collection of all states visited and actions taken in these rounds. At the end of the episode a binary reward $y_\tau \in \{0, 1\}$ is drawn from an unknown distribution $\mathbb{Q}(\cdot|\tau)$ and handed to the learner. This protocol continues for N episodes and the learner's goal is to maximize the number of expected binary "successes."

One approach to deal with the lack of a reward function in the literature is Inverse Reinforcement Learning [23], which uses demonstrations of good trajectories to learn a reward function. However,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

this approach is difficult to use when good demonstrations are either prohibitively expensive or difficult to obtain. Another closely related line of work studies reinforcement learning with preference feedback [2, 13, 3, 5, 35, 24, 36]. Our framework provides the learner with an even weaker form of feedback than that studied in this line of work. Instead of providing preferences between trajectories, we only inform the learner whether the task was completed successfully or not at the end.

To study whether it is possible to learn under such drastically limited feedback we study the case where the conditional rewards, y_τ, are drawn from an unknown logistic model (see Assumption 2.1). Under this assumption we show that learning is possible—we provide an optimism-based algorithm that achieves sub-linear regret (see Theorem 3.2). Technically our theory leverages recent results of Russac et al. [29] for the online estimation of the parameters of the underlying logistic model, and combining them with the UCBVI algorithm [4] to obtain regret bounds. Under an explorability assumption we also show that our algorithm is computationally efficient and we provide a dynamic programming algorithm to solve for the optimistic policy at every episode.

We note that Efroni et al. [10] study a similar problem to ours, such that a reward is revealed only at the end of the episode, but they assume that there exists a underlying Markovian reward function associated with each state-action pair, and reward revealed to the learner is the sum of rewards over the state-action pairs. This assumption allows them to use an online linear bandit algorithm [1] to directly estimate the underlying Markovian reward function. This is not possible in our setting since we do not assume the existence of an underlying Markovian reward function.

The remainder of the paper is organized as follows. In Section 2 we introduce notation and describe our setting. In Section 3 we present our algorithm and main results. Under an explorability assumption we prove that our algorithm is computationally efficient (in Section E). In Section 4 we present a simple experiment where we show that learning is possible with binary feedback once per episode. Section 5 points to other related work and we conclude with a discussion in Section 6. Other technical details and proofs are deferred to the appendix.

2 Preliminaries

This section presents notational conventions and a description of the setting.

2.1 Notation

For any $k \in \mathbb{N}$ we denote the set $\{1, \dots, k\}$ by $[k]$. Given any set $\mathcal{T}$, let $\Delta_{\mathcal{T}}$ denote the simplex over this set. Given a vector $\mathbf{v}$, for any $p \in \mathbb{N}$, let $\|\mathbf{v}\|_p$ denote the ℓ_p norm of the vector. Given a vector $\mathbf{v}$ and positive semi-definite matrix $\mathbf{M}$, define $\|\mathbf{v}\|_{\mathbf{M}} := \sqrt{\mathbf{v}^\top \mathbf{M} \mathbf{v}}$. Given a matrix $\mathbf{M}$ let $\|\mathbf{M}\|_{op}$ denote its operator norm. For any positive semi-definite matrix $\mathbf{M}$ we use $\lambda_{\max}(\mathbf{M})$ and $\lambda_{\min}(\mathbf{M})$ to denote its maximum and minimum eigenvalues respectively. We will use $C_1, C_2, \dots$ to denote absolute constants whose values are fixed throughout the paper, and $c, c', \dots$ to denote "local" constants, which may take different values in different contexts. We use the standard "big Oh notation" [see, e.g., 6].

2.2 The Setting

We study a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, H)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathbb{P}(\cdot|s, a)$ is the law that governs the transition dynamics given a state and action pair (s, a), and $H \in \mathbb{N}$ is the length of an episode. Both the state space $\mathcal{S}$ and action space $\mathcal{A}$ are finite in our paper.

The learner's trajectory τ is the concatenation of all states and actions visited during an episode; that is, $\tau := (s_1, a_1, \cdots, s_H, a_H)$. Given any $h \in [H]$ and trajectory τ, a sub-trajectory $\tau_h := (s_1, a_1, \dots, s_h, a_h)$ is all the states and actions taken up to step h. Also set $\tau_0 := \emptyset$. Let $\tau_{h:H} := (s_h, a_h, \dots, s_H, a_H)$ denote the states and action from step h until the end of the episode. Let Γ be the set of all possible trajectories τ. Analogously, for any $h \in [H]$ let Γ_h be the set of all sub-trajectories up to step h. At the start of each episode the initial state s_1 is drawn from a fixed distribution ρ that is known to the learner.

At the end of an episode the trajectory τ gets mapped to a feature map $\phi(\tau) \in \mathbb{R}^d$. We also assume that the learner has access to this feature map ϕ. Here are two examples of feature maps:

1. **Direct parametrization:** Without loss of generality assume that $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ and $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$. The feature map $\phi(\tau) = \sum_{h=1}^{H} \phi_h(s_h, a_h)$, where the per-step maps $\phi_h(s, a) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|H}$ are defined as follows:

$$(\phi_h(s,a))_j = \begin{cases} 1 & \text{if } j = (h-1)|\mathcal{S}||\mathcal{A}| + (s-1)|\mathcal{A}| + a, \\ 0 & \text{otherwise.} \end{cases}$$

   The complete feature map $\phi(\tau) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|H}$ is therefore an encoding of the trajectory τ.

2. **Reduced parametrization:** Any trajectory τ is associated with a feature $\phi(\tau) \in \mathbb{R}^d$, where $d < |\mathcal{S}||\mathcal{A}|H$.

After the completion of an episode the learner is given a random binary reward $y_\tau \in \{0, 1\}$. Let $\mathbf{w}_\star \in \mathbb{R}^d$ be a vector that is unknown to the learner. We study the case where the rewards are drawn from a binary logistic model as described below.

Assumption 2.1 (Logistic model)**.** *Given any trajectory $\tau \in \Gamma$, the rewards are said to be drawn from a logistic model if the law of $y_\tau|\tau$ is*

$$y_\tau|\tau = \begin{cases} 1 & \textit{w.p.} \quad \mu\left(\mathbf{w}_\star^\top \phi(\tau)\right) \\ 0 & \textit{w.p.} \quad 1 - \mu\left(\mathbf{w}_\star^\top \phi(\tau)\right), \end{cases} \tag{1}$$

where for any $z \in \mathbb{R}$, $\mu(z) = \frac{1}{1+\exp(-z)}$ is the logistic function. We shall refer to $\mathbf{w}_\star$ as the "reward parameters."

We make the following boundedness assumptions on the features and reward parameters.

Assumption 2.2 (Bounded features and parameters)**.** *We assume that*

- $\|\mathbf{w}_\star\|_2 \leq B$ *for some known value* $B > 0$ *and*
- *for all* $\tau \in \Gamma$, $\|\phi(\tau)\|_2 \leq 1$.

A policy π is a collection of per-step policies $(\pi_1, \ldots, \pi_H)$ such that

$$\pi_h : \Gamma_{h-1} \times \mathcal{S} \to \Delta_{\mathcal{A}}.$$

If the agent is using the policy π then at round h of the episode the learner plays according to the policy π_h. We let Π_h denote the set of all valid policies at step h and let Π denote the set of valid policies over the trajectory. Let $\mathbb{P}^\pi(\cdot|s_1)$ denote the joint probability distribution over the learner's trajectory τ and the reward y_τ when the learner plays according to the policy π and the initial state is s_1. Often when the initial state is clear from the context we will refer to $\mathbb{P}^\pi(\cdot|s_1)$ by simply writing $\mathbb{P}^\pi$. Also with some abuse of notation we will sometimes let $\mathbb{P}^\pi$ denote the distribution of the trajectory and the reward where the initial state is drawn from the distribution ρ.

Given an initial state $s \in \mathcal{S}$ the value function corresponding to a policy π is

$$V^\pi(s) := \mathbb{E}_{y_\tau, \tau \sim \mathbb{P}^\pi}\left[y_\tau \mid s_1 = s\right] = \mathbb{E}_{\tau \sim \mathbb{P}^\pi}\left[\mu\left(\mathbf{w}_\star^\top \phi(\tau)\right) \;\middle|\; s_1 = s\right],$$

where the second equality follows as the mean of y_τ conditioned on τ is $\mu(\mathbf{w}_\star^\top \phi(\tau))$. With some abuse of notation we denote the average value function as

$$V^\pi := \mathbb{E}_{s_1 \sim \rho}\left[V^\pi(s_1)\right].$$

Also define the optimal policy as

$$\pi_\star \in \operatorname*{arg\,max}_{\pi \in \Pi} V^\pi.$$

It is worth noting that in our setting the optimal policy may not be Markovian. The learner plays for a total of N episodes. The policy played in episode $t \in [N]$ is $\pi^{(t)}$ and its value function is

$V^{(t)} := V^{\pi^{(t)}}$. Also define the value function for the optimal policy to be $V_\star := V^{\pi_\star}$. Our goal shall be to control the regret of the learner, which is defined as

$$\mathcal{R}(N) := \sum_{t=1}^{N} V_\star - V^{(t)}. \tag{2}$$

The trajectories in these N episodes are denoted by $\{\tau^{(t)}\}_{t=1}^N$ and rewards received are denoted by $\{y^{(t)}\}_{t=1}^N$.

3 Optimistic Algorithms that Use Trajectory Labels

We now present an algorithm to learn from labeled trajectories. Throughout this section we assume that both Assumptions 2.1 and 2.2 are in force.

The derivative of the logistic function is $\mu'(z) = \frac{\exp(-z)}{(1+\exp(-z))^2}$, and therefore, μ is $1/4$-Lipschitz. The following quantity will play an important role in our bounds

$$\kappa := \max_{\tau \in \Gamma} \sup_{\mathbf{w}: \|\mathbf{w}\| \le B} \frac{1}{\mu'(\mathbf{w}^\top \phi(\tau))}.$$

A consequence of Assumption 2.2 is that $\kappa \le \exp(B)$. Since the true reward parameter $\mathbf{w}_\star$ is unknown we will estimate it using samples. At any episode $t \in [N]$, a natural way of computing an estimator of $\mathbf{w}_\star$, given past trajectories $\{\tau^{(q)}\}_{q \in [t-1]}$ and labels $\{y^{(q)}\}_{q \in [t-1]}$, is by minimizing the ℓ_2-regularized cross-entropy loss:

$$\mathcal{L}_t(\mathbf{w}) := -\sum_{q=1}^{t-1} y^{(q)} \log\left(\mu\left(\mathbf{w}^\top \phi(\tau^{(q)})\right)\right) - (1 - y^{(q)}) \log\left(1 - \mu\left(\mathbf{w}^\top \phi(\tau^{(q)})\right)\right) + \frac{\|\mathbf{w}\|_2^2}{2}.$$

This function is strictly convex and its minimizer is defined to be

$$\widehat{\mathbf{w}}_t := \arg\min_{\mathbf{w} \in \mathbb{R}^d} \mathcal{L}_t(\mathbf{w}). \tag{3}$$

Define a design matrix at every episode

$$\mathbf{\Sigma}_1 := \kappa \mathbf{I}, \quad \text{and} \quad \mathbf{\Sigma}_t := \kappa \mathbf{I} + \sum_{q=1}^{t-1} \phi(\tau^{(q)}) \phi(\tau^{(q)})^\top, \quad \text{for all } t \ge 1.$$

Further, define the confidence radius $\beta_t(\delta)$ as follows

$$\beta_t(\delta) := \left(1 + B + \rho_t(\delta)\left(\sqrt{1+B} + \rho_t(\delta)\right)\right)^{3/2} \tag{4}$$
$$\text{where,} \quad \rho_t(\delta) := d \log\left(4 + \frac{4t}{d}\right) + 2 \log\left(\frac{N}{\delta}\right) + \frac{1}{2}.$$

We adapt a result due to Russac et al. [29, Proposition 7] who studied the online logistic bandits problem to establish that at every episode and every trajectory the difference between $\mu(\mathbf{w}_\star^\top \phi(\tau))$ and $\mu(\widehat{\mathbf{w}}_t^\top \phi(\tau))$ is small.

Lemma 3.1. *For any $\delta \in (0, 1]$, define the event*

$$\mathcal{E}_\delta := \left\{ \textit{for all } t \in [N], \tau \in \Gamma : \left|\mu(\mathbf{w}_\star^\top \phi(\tau)) - \mu(\widehat{\mathbf{w}}_t^\top \phi(\tau))\right| \le \sqrt{\kappa} \beta_t(\delta) \|\phi(\tau)\|_{\mathbf{\Sigma}_t^{-1}} \right\}. \tag{5}$$

Then $\mathbb{P}(\mathcal{E}_\delta) \ge 1 - \delta$.

We provide a proof in Appendix B.2. The proof follows by simply translating [29, Proposition 7] into our setting. We note that we specifically adapt these recent results by Russac et al. [29] since they directly apply to $\widehat{\mathbf{w}}_t$, the minimizer of the ℓ_2-regularized cross-entropy loss. In contrast, previous work on the logistic bandits problem [see, e.g., 12, 11] established confidence sets for an estimator that was obtained by performing a non-convex (and potentially computationally intractable) projection of $\widehat{\mathbf{w}}_t$ onto the ball of Euclidean radius B.

Our algorithm shall construct an estimate of the transition dynamics $\widehat{\mathbb{P}}_t$. Let $N_t(s,a)$ be the number of times that the state-action pair (s,a) is encountered before the start of episode t, and let $N_t(s';s,a)$ be the number of times the learner encountered the state s' after taking action a at state s before the start of episode t. Define the estimator of the transition dynamics as follows:

$$\widehat{\mathbb{P}}_t(s'|a,s) := \frac{N_t(s';s,a)}{N_t(s,a)}. \tag{6}$$

Also define the state-action bonus at episode t

$$\xi_{s,a}^{(t)} := \min\left\{2, 4\sqrt{\frac{\log\left(\frac{6(|\mathcal{S}||\mathcal{A}|H)^H(8NH^2)^{|\mathcal{S}|}\log(N_t(s,a))}{\delta}\right)}{N_t(s,a)}}\right\}. \tag{7}$$

In this definition whenever $N_t(s,a)=0$, that is, when a state-action pair hasn't been visited yet, we define $\xi_{s,a}^{(t)}$ to be equal to 2. Finally, we define the optimistic reward functions

$$\bar{\mu}_t(\mathbf{w},\tau) := \min\left\{\mu\left(\mathbf{w}^\top\phi(\tau)\right) + \sqrt{\kappa}\beta_t(\delta)\|\phi(\tau)\|_{\mathbf{\Sigma}_t^{-1}}, 1\right\} \quad \text{and} \tag{8a}$$

$$\widetilde{\mu}_t(\mathbf{w},\tau) := \bar{\mu}_t(\mathbf{w},\tau) + \sum_{h=1}^{H-1}\xi_{s_h,a_h}^{(t)}. \tag{8b}$$

The first reward function $\bar{\mu}_t$ is defined as above to account for the uncertainty in the predicted value of $\mathbf{w}_\star$ in light of Lemma 3.1, and the second reward function $\widetilde{\mu}_t$ is designed to account for the error in the estimation of the transition dynamics $\mathbb{P}$. With these additional definitions in place we are ready to present our algorithms and main results.

3.1 UCBVI with Trajectory Labels

Our first algorithm is an adaptation of the UCBVI algorithm [4] to our setting with labeled trajectories.

Algorithm 1: UCBVI with trajectory labels.

Input: State and action spaces $\mathcal{S}, \mathcal{A}$.
Initialize $\widehat{\mathbb{P}}_1 = \mathbf{0}$, visitation set $\mathcal{K} = \emptyset$.
for $t = 1, \cdots$ **do**

1. Calculate the $\widehat{\mathbf{w}}_t$ by solving equation (3).
2. If $t > 1$, compute $\pi^{(t)}$

$$\pi^{(t)} \in \arg\max_{\pi\in\Pi} \mathbb{E}_{s_1\sim\rho,\ \tau\sim\widehat{\mathbb{P}}_t^{\pi}(\cdot|s_1)}\left[\widetilde{\mu}_t(\widehat{\mathbf{w}}_t,\tau)\right]. \tag{9}$$

   Else for all $h, s, \tau_{h-1} \in [H]\times\mathcal{S}\times\Gamma_{h-1}$, set $\pi_h^{(1)}(\cdot|s,\tau_{h-1})$ to be the uniform distribution over the action set.
3. Observe the trajectory $\tau^{(t)} \sim \mathbb{P}^{\pi^{(t)}}$ and update the design matrix

$$\mathbf{\Sigma}_{t+1} = \kappa\mathbf{I} + \sum_{q=1}^{t}\phi(\tau^{(q)})\phi(\tau^{(q)})^\top. \tag{10}$$

4. Update the visitation set $\mathcal{K} = \{(s,a)\in\mathcal{S}\times\mathcal{A} : N_t(s,a) > 0\}$.
5. For all $(s,a)\in\mathcal{K}$, update $\widehat{\mathbb{P}}_{t+1}(\cdot|s,a)$ according to equation (6).
6. For all $(s,a)\notin\mathcal{K}$, set $\widehat{\mathbb{P}}_{t+1}(\cdot|s,a)$ to be the uniform distribution over states.

Theorem 3.2. *For any $\bar{\delta}\in(0,1]$, set $\delta = \bar{\delta}/(6N)$ then under Assumptions 2.1 and 2.2 the regret of Algorithm 1 is upper bounded as follows:*

$$\mathcal{R}(N) \le \widetilde{O}\left(\left[H\sqrt{(H+|\mathcal{S}|)|\mathcal{S}||\mathcal{A}|} + H^2 + \sqrt{\kappa d}(d^3 + B^{3/2})\right]\sqrt{N} + (H+|\mathcal{S}|)H|\mathcal{S}||\mathcal{A}|\right),$$

with probability at least $1-\bar{\delta}$.

The regret of our algorithm scales with $\sqrt{N}$ and polynomially with the horizon, number of states, number of actions, κ, dimension of the feature maps and length of the reward parameters (B). The minimax regret in the standard episodic reinforcement learning is $O(\sqrt{H|\mathcal{S}||\mathcal{A}|N})$ [25, 4]. Here we pay for additional factors in H, $|\mathcal{S}|$ and κ since our rewards are non-Markovian and are revealed to the learner only at the end of the episode. We provide a proof of this theorem in Appendix B. For a more detailed bound on the regret with the logarithmic factors and constants specified we point the interested reader to inequality (41) in the appendix.

Proof sketch. First we show that with high probability at each episode the value function of the optimal policy $V_\star$ is upper bounded by $\widetilde{V}^{(t)} := \mathbb{E}_{s_1\sim\rho,\ \tau\sim\widehat{\mathbb{P}}_t^{\pi^{(t)}}(\cdot|s_1)}\left[\widetilde{\mu}_t(\widehat{\mathbf{w}}_t,\tau)\right]$ (the value function of the policy $\pi^{(t)}$ when the rewards are dictated by $\widetilde{\mu}_t$ and the transition dynamics are given by $\widehat{\mathbb{P}}_t$). Then we provide a high probability bound on the difference between the optimistic value function $\widetilde{V}^{(t)}$ and the true value function $V^{(t)}$ to obtain our upper bound on the regret. In both of these steps we need to relate expectations with respect to the true transition dynamics $\mathbb{P}$ to expectations with respect to the empirical estimate of the transition dynamics $\widehat{\mathbb{P}}_t$. We do this by using our concentration results: Lemmas B.1 and B.2 proved in the appendix. While analogs of these lemmas do exist in previous theoretical studies of episodic reinforcement learning, here we had to prove these lemmas in our setting with non-Markovian trajectory-level feedback (which explains why we pay extra factors in H and $|\mathcal{S}|$).

3.2 UCBVI with Added Exploration

Although the regret of Algorithm 1 is sub-linear it is not guaranteed to be computationally efficient since finding the optimistic policy $\pi^{(t)}$ (in equation (9)) at every episode might prove to be difficult. In this section we will show that when the features are sum-decomposable and the MDP satisfies an explorability assumption then it will be possible to find a computationally efficient algorithm with sub-linear regret (albeit with a slightly worse scaling with the number of episodes N).

Assumption 3.3 (Sum-decomposable features)**.** *We assume that the feature maps $\phi \in \mathbb{R}^d$ are sum-decomposable over the different steps of the trajectory, that is, $\phi(\tau) = \sum_{h=1}^{H}\phi_h(s_h,a_h)$.*

Under this assumption, given any $\mathbf{w} \in \mathbb{R}^d$ and any trajectory $\tau \in \Gamma$, $\mathbf{w}^\top\phi(\tau) = \sum_{h=1}^{H}\mathbf{w}^\top\phi_h(s_h,a_h)$. We also make the following explorability assumption.

Assumption 3.4 (Explorability)**.** *For any $s,s'\in\mathcal{S}$, $a,a'\in\mathcal{A}$, and $h\neq h'\in[H]$, suppose that*

$$\phi_h(s,a)^\top\phi_{h'}(s',a') = 0.$$

Further assume that there exists $\omega\in(0,1)$ such that for any unit vector $\mathbf{v}\in\mathbb{R}^d$ and $h\in[H]$ we have that

$$\sup_{\pi\in\Pi}\mathbb{E}_{s_1\sim\rho,\tau\sim\mathbb{P}^\pi}\left[\mathbf{v}^\top\phi_h(s_h,a_h)\right] \geq \omega.$$

In a setting with Markovian rewards an assumption similar to the one above has been made previously by Zanette et al. [38].

In this subsection we will use the sum-decomposable bonus $\sqrt{\kappa}\beta_t(\delta)\sum_{h=1}^{H}\|\phi_h(s_h,a_h)\|_{\mathbf{\Sigma}_t^{-1}}$ instead of the bonus $\sqrt{\kappa}\beta_t(\delta)\|\phi(\tau)\|_{\mathbf{\Sigma}_t^{-1}}$ used in the definition of $\bar{\mu}_t$ above (see equation (8a)). Define the reward functions

$$\bar{\mu}_t^{\mathsf{sd}}(\mathbf{w},\tau) := \min\left\{\mu\left(\mathbf{w}^\top\phi(\tau)\right) + \sqrt{\kappa}\beta_t(\delta)\sum_{h=1}^{H}\|\phi_h(s_h,a_h)\|_{\mathbf{\Sigma}_t^{-1}}, 1\right\} \quad \text{and} \tag{11a}$$

$$\widetilde{\mu}_t^{\mathsf{sd}}(\mathbf{w},\tau) := \bar{\mu}_t^{\mathsf{sd}}(\mathbf{w},\tau) + \sum_{h=1}^{H-1}\xi_{s_h,a_h}^{(t)}. \tag{11b}$$

To prove a regret bound for an algorithm that uses these rewards our first step shall be to prove that the sum-decomposable bonus also leads to an optimistic reward function (that is, the value function

defined by these rewards sufficiently over-estimates the true value function). To this end, we will first use Algorithm 2 to find an exploration mixture policy $\bar{U}$ and play according to it at episode t with probability $1/t^{1/3}$. This policy $\bar{U}$ will be such that the minimum eigenvalue of

$$\mathbb{E}_{s_1\sim\rho,\ \tau\sim\mathbb{P}^{\bar{U}}(\cdot|s_1)}\left[\phi(\tau)\phi(\tau)^\top\right] \tag{12}$$

is lower bounded as a function of d, ω and N (see Lemma 3.5). This property shall allow us to upper bound the condition number of the design matrix $\mathbf{\Sigma}_t$ and subsequently ensure that the rewards $\bar{\mu}_t^{\mathsf{sd}}$ and $\widetilde{\mu}_t^{\mathsf{sd}}$ are optimistic. Given a unit vector $\mathbf{v}$ define a reward function at step h as follows:

$$r_h^{\mathbf{v}}(s,a) := \mathbf{v}^\top \phi_h(s,a). \tag{13}$$

Let $r^{\mathbf{v}} := (r_1^{\mathbf{v}}, \ldots, r_H^{\mathbf{v}})$ be the reward function over the entire trajectory that uses the reward function $r_h^{\mathbf{v}}$ at step h. As a subroutine Algorithm 2 uses the EULER algorithm [37]. (We briefly note that other reinforcement learning algorithms with PAC or regret guarantees [e.g., 4, 17] could also be used here in place of EULER.)

Algorithm 2: Find exploration mixture.

Input: Initial unit vector $\mathbf{v}_1$, Exploration lower bound ω, number of EULER episodes N_{EUL}, number of evaluation episodes N_{EVAL}.
Initialize: $\mathbf{A}_0 = \frac{\omega^2}{16}\mathbf{I}$, $n = 0$ and $\lambda_{\min} = \inf_{\mathbf{z}\in\mathbb{R}^d} \mathbf{z}^\top \mathbf{A}_0 \mathbf{z}$.
while $\lambda_{\min} < \frac{\omega^2}{8}$ **do**
  Update the counter $n \leftarrow n+1$.
  Set $U_n \leftarrow \mathsf{EULER}(\{r^{\mathbf{v}_n}, N_{\mathsf{EUL}})$ //`run` EULER `for` N_{EUL} `episodes`.
  for *t=1,...,*N_{EVAL} *episodes* **do**
    Sample a trajectory $\tau_n^{(t)} \sim \rho \times \mathbb{P}^{U_n}$.
  Calculate the average feature $\widehat{\mathbf{a}}_n = \sum_{t=1}^{N_{\mathsf{EVAL}}} \phi(\tau_n^{(t)})/N_{\mathsf{EVAL}}$.
  Update the matrix $\mathbf{A}_n \leftarrow \mathbf{A}_{n-1} + \widehat{\mathbf{a}}_n \widehat{\mathbf{a}}_n^\top$.
  Update the minimum eigenvalue: $\lambda_{\min} \leftarrow \inf_{\mathbf{z}\in\mathbb{R}^d} \mathbf{z}^\top \mathbf{A}_n \mathbf{z}$.
  Set $\mathbf{v}_n$ to be the minimum eigenvector of $\mathbf{A}_n$.
Set $n_{\mathsf{loop}} = n$.
Return: (i) $\bar{U} = \mathsf{Unif}(U_1, \cdots, U_{n_{\mathsf{loop}}})$ //`the uniform mixture over the policies`;
(ii) $N_{\mathsf{exp}} = n_{\mathsf{loop}} \times (N_{\mathsf{EUL}} + N_{\mathsf{EVAL}})$ //`total number of episodes`.

Lemma 3.5. *There exist positive absolute constants C_1 and C_2 such that, under Assumptions 2.2, 3.3 and 3.4, if Algorithm 2 is run with $N_{\mathsf{EUL}} = \frac{C_1|\mathcal{S}|^2|\mathcal{A}|H^2\log\left(\frac{|\mathcal{S}||\mathcal{A}|N^2 d}{\delta\omega^2}\right)}{\omega^2}$ and $N_{\mathsf{EVAL}} = \frac{C_2 d^3 \log^3\left(\frac{Nd^2}{\delta\omega^2}\right)}{\omega^4}$, and $N > \frac{d\log\left(1+\frac{16N}{d\omega^2}\right)}{\log(3/2)}(N_{\mathsf{EUL}} + N_{\mathsf{EVAL}}) =: \bar{N}_{\mathsf{exp}}$ then, with probability at least $1-2\delta$, we have $N_{\mathsf{exp}} \le \bar{N}_{\mathsf{exp}}$ and furthermore:*

$$\mathbb{E}_{s_1\sim\rho,\ \tau\sim\mathbb{P}^{\bar{U}}(\cdot|s_1)}\left[\phi(\tau)\phi(\tau)^\top\right] \succeq \frac{\omega^2\log(3/2)}{32d\log\left(d\log\left(1+\frac{16N}{d\omega^2}\right)\right)}\mathbf{I}.$$

This lemma is proved in Appendix C. With this lemma in place we now present our modified algorithm under the explorability assumption. In the first few episodes this algorithm finds the exploration mixture policy $\bar{U}$. In a subsequent episode t this algorithm acts according to the policy $\pi^{(t)}$ which maximizes the value function associated with the rewards $\widetilde{\mu}_t^{\mathsf{sd}}(\widehat{\mathbf{w}}_t, \tau)$ with probability $1 - \frac{1}{t^{1/3}}$. Otherwise it uses the exploration mixture policy $\bar{U}$.

Algorithm 3: UCBVI with trajectory labels and added exploration.

Input: State and action spaces $\mathcal{S}, \mathcal{A}$, Initial unit vector $\mathbf{v}_1$, Exploration lower bound ω, number of EULER episodes N_{EUL}, number of evaluation episodes N_{EVAL}.

Initialize $\widehat{\mathbb{P}}_1 = \mathbf{0}$, visitation set $\mathcal{K} = \emptyset$.

Find exploration mixture policy $\bar{U}$ in N_{exp} episodes by running Algorithm 2.

for $t = N_{\mathsf{exp}} + 1, \cdots, N$ **do**

1. Calculate $\widehat{\mathbf{w}}_t$ by solving equation (3).
2. If $t > N_{\mathsf{exp}} + 1$, compute $\pi^{(t)}$

$$\pi^{(t)} \in \arg\max_{\pi} \mathbb{E}_{s_1 \sim \rho,\ \tau \sim \widehat{\mathbb{P}}_t^{\pi}(\cdot|s_1)} \left[\widetilde{\mu}_t^{\mathsf{sd}}(\widehat{\mathbf{w}}_t, \tau) \right]. \tag{14}$$

   Else for all $h, s, \tau_{h-1} \in [H] \times \mathcal{S} \times \Gamma_{h-1}$, set $\pi_h^{(1)}(\cdot|s, \tau_{h-1})$ to be the uniform distribution over the action set.
3. Sample $b_t = \begin{cases} 0 & \text{w.p. } 1 - \frac{1}{t^{1/3}}, \\ 1 & \text{w.p. } \frac{1}{t^{1/3}}. \end{cases}$
4. If $b_t = 1$ then set $\pi^{(t)} \leftarrow \bar{U}$.
5. Observe the trajectory $\tau^{(t)} \sim \mathbb{P}^{\pi^{(t)}}$ and update the design matrix

$$\mathbf{\Sigma}_{t+1} = \kappa \mathbf{I} + \sum_{q=N_{\mathsf{exp}}+1}^{t} \phi(\tau^{(q)}) \phi(\tau^{(q)})^{\top}. \tag{15}$$

6. Update the visitation set $\mathcal{K} = \{(s,a) \in \mathcal{S} \times \mathcal{A} : N_t(s,a) > 0\}$.
7. For all $(s,a) \in \mathcal{K}$, update $\widehat{\mathbb{P}}_{t+1}(\cdot|s,a)$ according to equation (6).
8. For all $(s,a) \notin \mathcal{K}$, set $\widehat{\mathbb{P}}_{t+1}(\cdot|s,a)$ to be the uniform distribution over states.

The following is our regret bound for Algorithm 3.

Theorem 3.6. *For any $\bar{\delta} \in (0,1]$, set $\delta = \bar{\delta}/(12N)$. Under Assumptions 2.1, 2.2, 3.3 and 3.4, and for all $N > \bar{N}_{\mathsf{exp}}$ (see its definition in Lemma 3.5) if Algorithm 3 is run with the parameters N_{EUL} and N_{EVAL} set as specified in Lemma 3.5 then its regret is upper bounded as follows:*

$$\mathcal{R}(N) \leq \widetilde{O}\left(\frac{\sqrt{\kappa H} d}{\omega} (d^3 + B^{3/2}) N^{2/3} + \left[H \sqrt{(H + |\mathcal{S}|)|\mathcal{S}||\mathcal{A}|} + H^2 \right] \sqrt{N} \right.$$
$$\left. + (H + |\mathcal{S}|) H |\mathcal{S}||\mathcal{A}| + \frac{d^2}{\omega^2} \left(\frac{d^2}{\omega^2} + |\mathcal{S}|^2 |\mathcal{A}| H^2 \right) \right),$$

with probability at least $1 - \bar{\delta}$.

The proof of Theorem 3.6 is in Appendix D. For a more detailed bound on the regret with the logarithmic factors and constants specified we point the interested reader to inequality (58) in the appendix. The bound on the regret of this algorithm scales with $N^{2/3}$ up to poly-logarithmic factors. This is larger than the $\sqrt{N}$ regret bound (again up to poly-logarithmic factors) that we proved above for Algorithm 1 since here the learner plays according to the exploration policy $\bar{U}$ with probability $1/t^{1/3}$ throughout the run of the algorithm. However, the next proposition shows that by using the sum-decomposable reward function $\widetilde{\mu}_t^{\mathsf{sd}}$ the policy $\pi^{(t)}$ defined in equation (14) can be efficiently approximated.

Proposition 3.7. *For any $t \in [N]$ define $\widetilde{V}_t^{\mathsf{sd}}(\pi) := \mathbb{E}_{s_1 \sim \rho,\ \tau \sim \widehat{\mathbb{P}}_t^{\pi}(\cdot|s_1)} \left[\widetilde{\mu}_t^{\mathsf{sd}}(\widehat{\mathbf{w}}_t, \tau) \right]$. Given any $\varepsilon > 0$, under Assumptions 2.2, 3.3 and 3.4 it is possible to find a policy $\widehat{\pi}^{(t)}$ that satisfies*

$$\widetilde{V}_t^{\mathsf{sd}}(\pi^{(t)}) - \widetilde{V}_t^{\mathsf{sd}}(\widehat{\pi}^{(t)}) \leq \varepsilon,$$

using at most $\mathsf{poly}\left(|\mathcal{S}|, |\mathcal{A}|, H, d, B, \|\widehat{\mathbf{w}}_t\|_2, \frac{1}{\varepsilon}, \log\left(\frac{N}{\delta}\right)\right)$ *time and memory.*

We describe the approximate dynamic programming algorithm that can be used to find this policy $\widehat{\pi}^{(t)}$ and present a proof of this proposition in Appendix E. We also note that if we use an

ε-approximate policy $\widehat{\pi}^{(t)}$ instead of $\pi^{(t)}$ in Algorithm 3 then its regret increases by an additive factor of at most εN. (It is possible to easily check this by inspecting the proof of Theorem 3.6.) Thus, for example a choice of $\varepsilon = 1/N^{1/3}$ ensures that the regret of Algorithm 3 is bounded by $O(N^{2/3})$ with high probability if the approximate policy $\widehat{\pi}^{(t)}$ (which can be found efficiently) is used instead.

4 Experiments

In this section we experimentally show that it is possible to learn a good policy in a simple non-Markovian domain with binary rewards—received once per episode—using a policy gradient algorithm. We parameterize each policy π_θ by $\theta \in \mathbb{R}^k$. The gradients of the value function can be computed using the REINFORCE [34] algorithm as follows

$$\nabla_\theta V^{\pi_\theta} = \mathbb{E}_{y_\tau, \tau \sim \mathbb{P}^\pi} \left[y_\tau \left(\sum_{h=1}^{H} \nabla_\theta \log \left(\pi_\theta(a_h | s_h) \right) \right) \right].$$

We approximate this expectation empirically by using 30 sample trajectories, and use the Adam optimizer [19] with a default step size of one to update the policy. We studied the behavior of this algorithm on a custom 10×15 grid environment. The agent is initialized at a random location on the grid denoted by the large blue dot. Then the agent is allowed to take one of the actions $\{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}\}$, and move to an adjacent node (if permitted). During the last three steps of an episode, with $H = 30$, if the agent stays at either the black dot ('goal') or at any adjacent nodes marked by the red dots, then the agent receives a reward of 1, while if the agent is not at one of these nodes during the last three steps then it receives a reward of 0. The location of the 'goal' node is also randomly chosen at each episode. We parametrize the policy using a fully connected neural network with 10 hidden layers and with width 4. The state representation that is fed to this policy is of the form $(x^{\text{current}}, y^{\text{current}}, x^{\text{goal}}, y^{\text{goal}})$, where $(x^{\text{current}}, y^{\text{current}})$ represents the current coordinates of the agent and $(x^{\text{goal}}, y^{\text{goal}})$ denotes the coordinates of the 'goal' node.

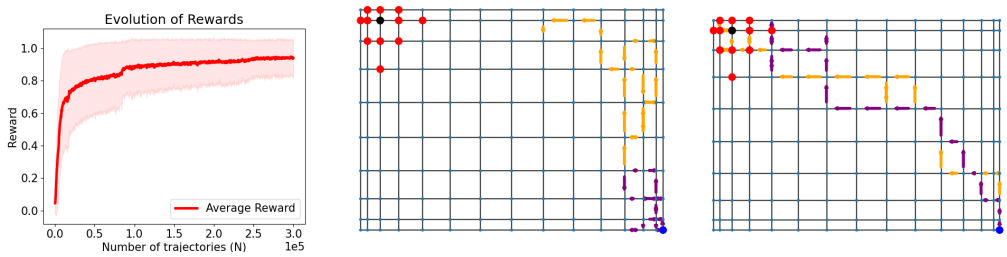

Figure 1. Left: Reward learning curve averaged over 40 independent runs. The shaded region represents a confidence interval which is $\pm$standard deviation. **Middle:** The purple and yellow paths represent two sample paths taken by an initial random policy. **Right:** The purple and yellow paths represent two sample paths taken by a trained policy.

5 Additional Related Work

There have been many theoretical results that analyze regret minimization in standard episodic reinforcement [16, 27, 14, 26, 4, 17, 7, 37, 32, 9, 28]. As stated above to estimate the reward parameter we rely on the recent results by Russac et al. [29] who in term built on earlier work [11, 12] that analyzed the GLM-UCB algorithm. Dong et al. [8] provided and analyzed a Thompson sampling approach for the logistic bandits problem.

6 Discussion

We have shown that efficient learning is possible when the rewards are non-Markovian and delivered to the learner only once per episode. It would be interesting to see if one can establish guarantees under more general reward models than the logistic model that we study here. Another interesting question is if faster rates of learning are possible when the learner obtains ranked trajectories (that is, moving beyond binary labels).

References

[1] Y. Abbasi-Yadkori, D. Pal, and C. Szepesvari. Online-to-confidence-set conversions and application to sparse stochastic bandits. In *Artificial Intelligence and Statistics*, pages 1–9, 2012.

[2] R. Akrour, M. Schoenauer, and M. Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27, 2011.

[3] R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by feedback. In *International Conference on Machine Learning*, pages 1503–1511, 2014.

[4] M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272, 2017.

[5] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, page 4299–4307, 2017.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2009.

[7] C. Dann, T. Lattimore, and E. Brunskill. Unifying PAC and regret: uniform PAC bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5717–5727, 2017.

[8] S. Dong, T. Ma, and B. Van Roy. On the performance of Thompson sampling on logistic bandits. In *Conference on Learning Theory*, pages 1158–1160, 2019.

[9] Y. Efroni, N. Merlis, M. Ghavamzadeh, and S. Mannor. Tight regret bounds for model-based reinforcement learning with greedy policies. In *Advances in Neural Information Processing Systems*, pages 12224–12234, 2019.

[10] Y. Efroni, N. Merlis, and S. Mannor. Reinforcement learning with trajectory feedback. *arXiv preprint arXiv:2008.06036*, 2020.

[11] L. Faury, M. Abeille, C. Calauzènes, and O. Fercoq. Improved optimistic algorithms for logistic bandits. In *International Conference on Machine Learning*, pages 3052–3060, 2020.

[12] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594, 2010.

[13] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2): 123–156, 2012.

[14] A. Gopalan and S. Mannor. Thompson sampling for learning parameterized Markov decision processes. In *Conference on Learning Theory*, pages 861–898, 2015.

[15] S. R. Howard, A. Ramdas, J. McAuliffe, and J. Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.

[16] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

[17] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4868–4878, 2018.

[18] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879, 2020.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[20] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

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

[23] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[24] E. Novoseller, Y. Wei, Y. Sui, Y. Yue, and J. Burdick. Dueling posterior sampling for preference-based reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 1029–1038, 2020.

[25] I. Osband and B. Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

[26] I. Osband and B. Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pages 2701–2710, 2017.

[27] I. Osband, D. Russo, and B. Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, 2013.

[28] A. Pacchiano, P. Ball, J. Parker-Holder, K. Choromanski, and S. Roberts. On optimism in model-based reinforcement learning. *arXiv preprint arXiv:2006.11911*, 2020.

[29] Y. Russac, L. Faury, O. Cappé, and A. Garivier. Self-concordant analysis of generalized linear bandits with forgetting. In *International Conference on Artificial Intelligence and Statistics*, pages 658–666, 2021.

[30] A. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[31] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550 (7676):354–359, 2017.

[32] M. Simchowitz and K. G. Jamieson. Non-asymptotic gap-dependent regret bounds for tabular MDPs. In *Advances in Neural Information Processing Systems*, pages 1153–1162, 2019.

[33] J. Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.

[34] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[35] 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.

[36] Y. Xu, R. Wang, L. Yang, A. Singh, and A. Dubrawski. Preference-based reinforcement learning with finite-time guarantees. In *Advances in Neural Information Processing Systems*, pages 18784–18794, 2020.

[37] A. Zanette and E. Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pages 7304–7312, 2019.

[38] A. Zanette, A. Lazaric, M. J. Kochenderfer, and E. Brunskill. Provably efficient reward-agnostic navigation with linear value iteration. *arXiv preprint arXiv:2008.07737*, 2020.

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]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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] All assumptions are stated clearly with the accompanying theoretical results.

(b) Did you include complete proofs of all theoretical results? [Yes] The complete proofs of all theoretical results are carefully proved in the appendix.

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] See Section 4.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 1.

(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)? [N/A]

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]